# Computational Modeling of a Time-Independent, Heterogeneous Reactor Core Using Simplified Discrete Ordinates Neutron Transport Techniques

Thesis

Kristofer S. Labowski, Capt, USAF

AFIT/GAP/ENP/01S-01

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

***AIR FORCE INSTITUTE OF TECHNOLOGY***

**Wright-Patterson Air Force Base, Ohio**

# Report Documentation Page

| | | |
|---|---|---|
| **Report Date**<br>9 Nov 2001 | **Report Type**<br>Final | **Dates Covered (from... to)**<br>01 Mar 2001 - 01 Sep 2001 |

| | |
|---|---|
| **Title and Subtitle**<br>Computational Modeling of a Time-Independent, Heterogeneous Reactor Core Using Simplified Discrete Ordinate Neutron Transport Techniques | **Contract Number**<br><br>**Grant Number**<br><br>**Program Element Number** |
| **Author(s)**<br>Capt Kristofer S. Labowski, USAF | **Project Number**<br><br>**Task Number**<br><br>**Work Unit Number** |
| **Performing Organization Name(s) and Address(es)**<br>Air Force Institute of Technology AFIT/ENP Graduate School of Engineering & Management (AFIT/EN) 2950 P Street, Bldg. 640 Wright-Patterson AFB, OH 45433-7765 | **Performing Organization Report Number**<br>AFIT/GAP/ENP/01S-01 |
| **Sponsoring/Monitoring Agency Name(s) and Address(es)** | **Sponsor/Monitor's Acronym(s)**<br><br>**Sponsor/Monitor's Report Number(s)** |

**Distribution/Availability Statement**
Approved for public release, distribution unlimited

**Supplementary Notes**

**Abstract**
The Linear Characteristic (LC) method on rectangular boxoid meshes is a discrete ordinate neutron transport technique that uses both zeroth and first moments of the angular neutron flux to construct a relatively accurate representation of neutron particle distributions in a given medium. The significant number of calculations required by LC when compared to more conventional methods such as Diamond Difference vastly increases computational time by a factor of 60 (or minutes to hours). A modified extrapolation, linear acceleration method with iterative shooting capability was adapted along with approximated initial guesses to the scalar flux to effectively reduce LC computational times by as much as 70% to 90%. Unlike conventional methods that contain instabilities and errors that are accentuated and magnified at optically thick boundaries, the LC method calculated stable consistent flux results through boundary regions of varying absorptive materials. The inherently computationally intensive linear algebra used by the LC method was simplified using FORTRAN 90/95 programming commands designed to optimize vector and matrix calculations and reduce and simplify the overall source code.

| | |
|---|---|
| **Subject Terms**<br>Neutron transport, Diamond Difference Method, Step Method, Boxoid Mesh Linear Characteristic Method, Isotropic Fission, Three-Dimensional Reactor Model, Fuel Management, Pressurized Water Reactor, Discrete Ordinates, Reactor Core, Commercial Reactor, Fuel Loading Patterns | |
| **Report Classification**<br>unclassified | **Classification of this page**<br>unclassified |
| **Classification of Abstract**<br>unclassified | **Limitation of Abstract**<br>UU |
| **Number of Pages**<br>155 | |

AFIT/GAP/ENP/01S-01

# Computational Modeling of a Time-Independent, Heterogeneous Reactor Core Using Simplified Discrete Ordinates Neutron Transport Techniques

THESIS

Presented to the Faculty

Department of Engineering Physics

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the

Degree of Master of Science

Kristofer S. Labowski, B.S., M.S., M.B.A.

Captain, USAF

September 2001

AFIT/GAP/ENP/01S-01

# Computational Modeling of a Time-Independent, Heterogeneous Reactor Core Using Simplified Discrete Ordinates Neutron Transport Techniques

Kristofer S. Labowski, B.S., M.S., M.B.A.

Captain, USAF

Approved:

| | |
|---|---|
| James C. Petrosky (Chairman) | date |
| Vincent J. Jodoin (Member) | date |
| William P. Baker (Member) | date |

## Acknowledgements

I would like to thank my advisor, LTC James Petrosky, for his guidance, counsel, and freedom to attack and implement my solutions to this research. A special thanks goes to Capt. Mark Suriano for his assistance in learning three-dimensional neutron transport methods, especially those applied to the more complex Suriano-developed 3D, boxoid Linear Characteristic method.

Finally, I would like to thank my wife for supporting me and helping me through a very difficult time in my academic life. I would not have competed this thesis without her understanding and patience.

# Table of Contents

## List of Figures

**List of Tables**

## Abstract

A three-dimensional neutron transport based, heterogeneous reactor code (3D-TRAN) was developed to model simple uniform isotropic sources and isotropic fission sources. The code was developed using level symmetric angular quadrature sets and three spatial quadratures: the Diamond Difference (DD), Step, and Linear Characteristic (LC) methods. Each method was analyzed and compared for accuracy, rate of convergence, and stability. The LC method was found to be the most accurate method with a broader range of stability through heterogeneous absorbing regions than the DD method. The Step method always remains positive and stable, but converges very slowly to a specified answer in the limit as the mesh is refined.

The isotropic fission source was examined against the parallelepiped and right circular cylinder diffusion theory analytical solutions. The resulting flux profiles closely matched the diffusion analytical solution with eigenvalues for the right circular cylinder matching those produced in a numerical diffusion code written by Harman (2001) to within a relative error of less than one percent. Numerous initialization routines and input files were written to handle reactor type heterogeneous geometries to include the fuel pin, the fuel assembly, the cylindrical reactor core with rings, and the commercial power reactor. All geometries have the ability to utilize axial variations in the core.

The convergence speed of the code was modified using a linear extrapolation/ shooting method designed to reduce the number of iterations to convergence during eigenvalue calculations. Initial results show improvements of at least 60% reductions in overall computation time. The method was optimized over homogenous fission reactor configurations.

# COMPUTATIONAL MODELING OF A TIME-INDEPENDENT, HETEROGENEOUS REACTOR CORE USING SIMPLIFIED DISCRETE ORDINATES NEUTRON TRANSPORT TECHNIQUES

# Chapter I: Introduction

The development of a three dimensional, structured cubic mesh Discrete Ordinates neutron transport based reactor code is presented. This code, 3D-TRAN, solves the linear, time-independent Boltzmann Transport Equation for the angular flux density using the Diamond Difference, Step, and Linear Characteristic spatial quadratures and a level symmetric angular quadrature. The performance and limitations of the code (Chapters 3 and 4) are discussed in detail along with the theoretical development (Chapter 2) of the algorithms and conclusions about the results (Chapter 5). Background and motivation for this project are explained in the proceeding paragraphs along with the problem statement and generalized approach to the problem.

## Background

Reactor physics and kinetic feedback is often simple to understand, but extremely difficult to represent analytically. Analytical representations generally center on idealizing a nuclear reactor core with neutron diffusion theory. Analytical diffusion theory is both easy to understand and simple to implement for homogenous problems. In most cases, more complicated reactor physics problems are modeled using computer-based codes that convert the differential diffusion equations to finite difference, element or nodal algorithms. These equations are then solved through iterative procedures capable of modeling more complex heterogeneous problems where discrete analytical representation is difficult to formulate and solve. The problem with diffusion theory is that it is an approximation of the more physically accurate theory of neutron transport. To derive diffusion theory, approximations are made to the Boltzmann neutron transport equation that remove angular and energy scatter dependence (by angular integration) and idealize all particle interactions as isotropic. The benefit of these approximations is that faster computational speed (if solving via a computer) can be attained at the limitation of reducing the realm of reactor problems that can be physically and correctly modeled. Diffusion theory becomes highly inaccurate when performing calculations near a vacuum boundary, near or inside of highly absorbing materials (such as reactor control rods) and within any other material that is strongly anisotropic in nature (or has a strong directionally dependent flux). Therefore, diffusion theory based computer models may not yield a desired result when trying to physically understand the particle interactions inside a reactor core. The more accurate way to model a reactor core is to use neutron transport theory directly.

Neutron transport theory incorporates a more accurate representation of the particle-to-particle interactions that occur inside a nuclear reactor core. The benefit is that a wider range of reactor problems can be solved that consider highly absorbing regions or vacuum boundary currents. The cost of this analysis is that the neutron transport equation cannot be solved analytically. The equations must be solved using one or more of the many numerical methods available to an analyst. The more popular methods include Monte Carlo (a statistical based stochastic method), which utilizes particle path/collision tracking techniques based on probability distribution counting statistics, and discrete ordinates, which discretizes the angular integration over specified fixed directions. The benefit of Monte Carlo analysis is that it best represents the physics of tracking a particle's path from birth, to collision(s), to final absorption or loss. The cost of the Monte Carlo calculation is that a significant number of particles (on the order of millions) must be represented to reduce error in the particle distribution (if the code is analogue or only capable of tracking one particle at a time). Monte Carlo solutions are generally accurate (to within the distribution error) and are often considered as the benchmark for any neutron transport calculation performed. The discrete ordinates calculation makes an approximation to the angular, spatial, and energy integral in the transport equation. The advantage to this calculation is that it is generally faster than Monte Carlo techniques and will theoretically converge to the correct answer if the angular, spatial, and energy grids are refined in such a way that the discetization of the transport integrals approach the true Riemann integral representation. The disadvantage of using discrete ordinates is finding accurate, stable numerical methods that correctly model the discretizations in the transport equation. There are many methods for handling the spatial and angular integrations such as the diamond difference spatial method (averaging technique) and Gauss-Legendre angular numerical integration. There are also spatial characteristic methods that perform numerical integrations along the streaming path of the particles. These spatial methods are considerably more accurate than the diamond difference method, but are more computationally intensive (and expensive) to model. A more detailed description of these methods will be presented in Chapter 2.

As of the time of this thesis, the Air Force Institute of Technology (AFIT) has developed a working homogenous model of a three-dimensional cylindrical reactor core using two-group diffusion theory (Harman, 2001). This code has many limitations and only allows for highly idealized cylindrical reactor problem studies. The next step is to add the capability of performing heterogeneous or multi-region reactor analysis over more realistic reactor shapes (such as parallelepipeds or rough, jagged edged cylindrical shapes modeled after real commercial power reactors). The approach taken should be accurate and time efficient. For accuracy, this author has chosen to disregard diffusion theory and proceed with neutron transport using discrete ordinates. The discrete ordinates method was chosen because of its numerical advantages in converging to the "correct" solution in the limit of spatial and angular mesh refinement. The benefit is that a wider range of problems can be analyzed more accurately without the limitations of diffusion approximations. The heterogeneous problem becomes relatively simple while adding the ability to model highly absorbing regions and vacuum boundary currents[1]. Of course, care must still be taken in using numerical methods to solve the neutron transport equation, especially when solving problems using absorbing materials.

Due to the limited time of this research, certain idealized assumptions from Harman's two-group diffusion code will still be used. These assumptions include the use of an isotropic fission source and scattering source on the right hand side of the transport equation. This is essentially the same source used by Harman (2001). The rest of the research will focus more on developing an accurate spatial scheme to best handle a heterogeneous medium. The complexity involved in evaluating differential scatter cross section data for an anisotropic scatter source will not be covered in this research and therefore limits the problems modeled to isotropic idealized conditions. Further descriptions of the numerical theory will follow in Chapter 2.

[1] Boundary current is the directional flow of neutrons through a cell or problem boundary. This quantity is a vector relation and is best described as N number of particles passing through a given unit surface area.

## Motivation

The development of this research stems from a desire to present nuclear engineering students with a modeling tool to explore reactor behavior and physical trends that occur during reactor operations. The use of diffusion theory or neutron transport theory as a model basis present both a learning opportunity and a familiarity with the limitations of both theories. The diffusion theory solution to reactor physics is covered extensively in undergraduate texts with explanations for analytical solutions as well as numerical applications with certain omissions from the development (see Harman, 2001). Neutron transport theory is generally discussed in a way that the student can see the relation of diffusion theory as a subset of neutron transport theory. The goal of this effort is to develop a working reactor computer model using neutron transport theory to provide a student with the opportunity to learn how to implement a transport code as well as learn about the numerical limitation imposed on this theory. A student versed in neutron transport will have a better understanding and appreciation for reactor physics from both diffusion and transport viewpoints. An understanding of the limitations of diffusion theory will aid students in designing appropriate reactor models for any nuclear reactor problem of interest; therefore, the neutron transport code developed in this effort will address some of the more relevant issues concerning proper implementation of a neutron transport based code package.

## Problem Statement

The goal of this research is to develop a working three-dimensional computational model of a heterogeneous, time-independent, critical, pressurized water reactor (PWR).

The development of this model has evolved from a study of three dimensional neutron transport in Cartesian coordinates using the discrete ordinates Linear Characteristic, Diamond Difference, and Step spatial approaches on a structured cubic mesh. More specifically, the development of the Linear Characteristic method over cubic cells and the direct coordinate transformations required will be derived and explained. The development of this research includes the equations used and the methodology chosen to solve the time-independent neutron transport equation. The final code includes a two group energy model with a built in expansion capability to model three or more energy groups. The code models heterogeneous unit fuel cells (circles within a square), fuel assemblies (a parallelepiped) and a full reactor core (a rough edged circle made of fixed dimension fuel assemblies). Also, the capability of modeling simplified uniform sources (for neutron shielding problems) as well as isotropic fission sources is included. The transport numerical method is examined for performance based on the criteria of speed, and accuracy in predicting reactor physics behavior in the homogenous and heterogeneous reactor cores. Monte Carlo based neutron transport serves as the benchmark needed to compare overall performance of the discrete ordinates algorithms explored here. The reliability and accuracy of the Monte Carlo transport technique adds credibility and reality to the foundations of this research.

## Approach

The primary goal of this research was to develop a working model of a reactor using neutron transport as the foundation of the numerical model. The code developed (called 3D-TRAN) had the following features:

1. Three-dimensional modeling capability.
2. Time-independence, steady-state analysis.
3. Numerically stable quadratures in angle and space.
4. Capability of performing calculations on coarse mesh grids (for core models) to within a defined stability range determined by analysis.
5. Model heterogeneous material regions inside the reactor.
6. Meshing scheme capable of modeling different reactor geometries to include:
    a. Unit fuel cell (concentric circular rings within a square)
    b. Parallelepiped (used to develop the fuel assembly)
    c. Right circular cylinder (to model fixed dimension assemblies as a rough circular core as in a real power reactor)
    d. Cylinder with multiple concentric rings (multi region core)
    e. Ability to add axial variations to all geometries.
7. Model multiple boundary conditions on each of the geometries listed above to include several combinations of the following:
    a. Vacuum
    b. Symmetric, full reflective
    c. Albedo, reflective.
8. Model simple uniform source emitters and isotropic scatter sources.
9. Model isotropic fission sources to represent a reactor.
10. Calculate eigenvalues for fission based problems.

The primary approach undertaken (to model a nuclear reactor ) in this study involved solving the time-independent Boltzmann Transport Equation (BTE) for neutral particles. This equation, in its most general form, is:

$$[\hat{\Omega}\cdot\vec{\nabla} + \sigma(\vec{r}, E)]\Psi(\vec{r}, \hat{\Omega}, E) = S(\vec{r}, \hat{\Omega}, E). \quad (1.1)$$

The terms on the left hand side (LHS) of the equation represent neutron losses and the term on the right hand side (RHA) represents neutron sources.  The adaptation of this equation to numerical applications is discussed in detail in Chapter 2.

The source on the RHS can consist of a uniform, isotropic emitter, isotropic or anisotropic scatter source or a fission source (either single group or multi-energy group dependent). For our case, three basic sources were chosen.  First, the uniform isotropic emitter and the simplified isotropic scatter source (using a simple scatter fraction of 0.0 to 1.0 representing full absorption to full scatter) were used to test the three dimensional spatial quadrature routines developed in Chapter 2. The simplified source allows for known inputs into the LHS of the BTE to help test for correct implementation of the spatial quadrature. Lastly, an isotropic representation of the fission source similar to Harman (2001) was used as a first try at using neutron transport to model a reactor. "Within group" scattering and "down scatter" (neutrons going to the next lowest energy level or direct coupling) is assumed. Any addition of anisotropic scatter, etc., to the fission source will increase overall computational time and computer modeling complexity.  Due to the limited time available for research, the anisotropic source was disregarded.

After choosing the type of sources to model, the discretizations of the angular and spatial domain on the LHS of the BTE[2] were chosen. For angular discretization, a Level Symmetric numerical integration quadrature formula was used.  There were several quadrature rules to pick from (Atkinson, 1989) (Burden and Faires, 1997) (Lewis and Miller, 1993); however, the Level Symmetric quadrature has traditionally been proven to work well in three-dimensional Cartesian coordinates, especially for symmetry

[2] Otherwise known as the Discrete Ordinates approach.

boundaries (Lewis and Miller, 1993) (Carlson, 1971). A more descriptive explanation of the angular discretization is found in Chapter 2 and Appendix A.

The next step was to pick the neutron energy groups. For this study, a two-group representation consisting of fast and thermal neutrons or a one-group thermal neutron representation was chosen. The two-group model is the traditional separation of the neutron energy spectrum commonly found in reactor applications (Larmash, 1983) (Duderstadt, 1976) (Ott and Bezella, 1985). Sometimes, a four-group representation is used with three fast groups and one thermal group (Duderstadt, 1976). The finer resolution of the energy spectrum can produce more accurate approximations to the reactor physics; however, the total gain in accuracy found with finer resolution is generally minimal when comparing the computational cost of calculating two extra energy groups. With this in mind, the code was written in a generalized form for more than two energy groups (with one fast group up to G total energy groups) for any future analysis that desires a finer resolution in the thermal energy spectrum of the neutrons.

Now that the energy groups are defined, the spatial quadrature was chosen. This quadrature approximates the LHS of the BTE. A proper choice must be made for this quadrature in order to maintain numerical stability for many problems of interest. The traditional spatial quadrature method is Diamond Difference (DD) (Lewis and Miller, 1993) (O'Dell, 1987) (Carlson and Lathrop, 1965). This method uses an averaging technique over inflow and outflow fluxes to calculate the central cell average. It is both easy to implement and simple to understand. The DD method is linear in nature and has second order convergence, both of which are generally desired properties of spatial quadratures (Lathrop, 1969). The third desired property of spatial quadratures is positivity. The method should produce strictly non-negative fluxes because negative fluxes are numerically unrealistic artifacts that sometimes creep into numerical solutions; therefore, negative fluxes produce unreliable results. The DD method does not have strict positivity and suffers from numerical instability if the mesh width is too large or the material is too dense[3]. These instabilities generally result in negative unrealistic flux values that can, and usually do, oscillate about zero - especially near or at a boundary between two materials with a high differential between densities (Mathews 1999). An excellent description of this numerical instability in two dimensions along with several other spatial quadratures is presented in Mathews (1999).

Based on the findings of Mathews (1999), two other methods were chosen to compare against Diamond Difference. The first is the Step method. This method assumes a constant source within the cell; therefore, the exiting flux out of the cell is set equal to the cell average. This method is even easier than DD to implement and understand. The benefit of Step is that it is always non-negative and linear in nature. A disadvantage of Step is its consistent ability to overestimate the flux because of its smearing effects on material meshes (Mathews, 1999). The benefit of Step is that it converges quickly since the number of computations per cell is extremely small and simple to evaluate.

The final spatial quadrature examined is the Linear Characteristic method (LC). The LC method calculates both the zeroth moment (the average) and the first moments of the flux. This advantage allows LC to produce more accurate results over larger mesh cell sizes than the Diamond Difference method (Mathews, 1999). The LC method is also more stable than the DD method; however, it does not have strict positivity. The LC

[3] The combination of the cell width and material cross section forms a relation commonly referred to as optical thickness.

method is a linear method with second order convergence or better depending on the mesh geometry and size (Brennan, 1996) (Miller, 1998) (Mathews, Miller, and Brennan, 2000). In general, the LC method assumes a linear distribution of the source flux across inflow and outflow faces of the cell as well as within the cell itself. The Linear Characteristic method is more complex to develop and requires a substantial amount of computational time compared to DD and Step methods. The biggest advantage to any of the characteristic methods is that they perform integrations along the path of the particles and reduce smearing effects caused by simple averaging across the cell[4]. The vast complexity and intensive modeling of the LC method are explained in further detail in Chapter 2.

Investigation and development of the 3D-TRAN code used several test problems based on the Monte Carlo statistical approach[5]. The use of Monte Carlo as a benchmark to check the 3D-TRAN code provided an alternative method for checking the Discrete Ordinates numerical approach. The Monte Carlo routine used, named the Monte Carlo Analogue $S_n$ code, was developed at AFIT by Suriano (2001). The Analogue $S_n$ Monte Carlo code allowed for direct comparison and validation of spatial quadratures used in the 3D-TRAN code for both continuous angle, energy, and space (Analogue) as well as continues space, energy and discrete angle ($S_n$).

[4] The Exponential Characteristic method (EC) is another characteristic spatial quadrature that produces strictly non-negative fluxes (at second order convergence or better) and works extremely well over large cell widths and absorbing regions. The disadvantage is that it is highly non-linear in nature and requires numerical root solving techniques to solve for the coefficients of the assumed form of the source. The computational complexity will drive calculation times higher than the LC method; therefore, the EC method was disregarded as a viable spatial quadrature for this research at this time. A more detailed analysis of this method is presented in Miller (1998) for 3D tetrahedral meshes.
[5] The Monte Carlo statistical method is considered as the primary benchmark of reactor/particle transport based numerical models. Further detail on Monte Carlo can be found in Lewis and Miller (1993).

Finally, the code is written in FORTRAN 95 for its versatility over FORTRAN 77. Newer intrinsic functions as well as automatic compiler optimization allow for one-line linear algebra operations and matrix manipulations. The input for the code is currently set as a Namelist input file (see Ellis, 1994). This type of input allows for versatility and growth as the code develops. Easy user interface input menus were not developed in this effort. All material values and geometry types are input into the Namelist file along with types of sources and quadrature methods chosen. A preprocessing subroutine handles all initializations of material arrays corresponding with desired geometry and required boundary conditions. Angular ordinates are assigned based on the $S_n$ quadrature set chosen. Reflection boundary arrays are built to link ordinates that are reflections of one another at a reflective or albedo boundary. These arrays link the angular fluxes (a function of the angular ordinate) together for reflective boundaries and allow for the passing of reflected values with appropriate sign change to the correct angular flux mesh point.

The computer code follows a specific order for calculating through the material problem mesh. The mesh walk scheme begins by picking a direction ordinate ($\mu$, $\eta$, and $\xi$ in three dimensions) and walking in that general direction[6] (with the flow of particles) through the entire (x, y, z) mesh, starting by moving across a row in the x direction and proceeding one step at a time in the y direction completing an xy plane. After an xy plane is calculated, the walk moves up one step in z to calculate the next xy plane. Boundary

[6] The walk through the spatial mesh follows the direction assigned by the most recent angular direction ordinate. As directions rotate about the unit directional sphere, the mesh walk must be adjusted accordingly to follow the flow of particles. This involves starting the spatial mesh walk at a new location for each angular ordinate based on its 3D vector direction. Performing the mesh walk this way produces a stable solution that converges more quickly when compared to walking against the directional flow of particles (Lewis and Miller, 1993). The directional flow of particles is solely specified by the $S_n$ quadrature set (angular ordinates) assigned as the approximation to the angular integration of the BTE.

conditions are appropriately treated as they are confronted during the walk through the mesh. This continues until the entire mesh is calculated once for a given angular flux direction. A new ordinate is picked and the walk continues until all ordinates are calculated for the angular flux. The angular flux is converted to a scalar flux via the numerical angular quadrature rule and if isotropic sources are used, convergence is checked for the energy group. All energy groups are calculated and global convergence is checked (either by eigenvalue or scalar flux comparisons). If convergence is achieved, the code exits. Otherwise, calculations continue until convergence criteria are met or the maximum iteration is reached. A generalized FORTRAN pseudo-code summarizing this procedure is provided in figure 1.1. The following chapters will expand upon the numerical theory and testing required to fully develop the 3D-TRAN code.

```
INITILIZE spatial mesh arrays
DO OUTER LOOP until converged
    DO ENERGY LOOP starting with group 1 ending with group g
        DO DIRECTION ORDINATE pick a direction set
            DETERMINE mesh corner starting location
            DO SPATIAL WALK walking in positive (μ, η, ξ)
                WALK  in x, y, then z
                CHECK and update boundaries
            END DO spatial walk
        END DO directional ordinate
    END DO energy group
CALCULATE scalar flux  (and eigenvalue if fission source used)
CHECK CONVERGENCE of scalar flux and/or eigenvalue, EXIT if converged
END DO outer loop, repeat until converged
```

**Figure 1.1: FORTRAN Pseudo-Code for 3D-TRAN**

# Chapter II: General Theory

The following chapter discusses all theoretical development explored and implemented in this research. The discussion begins with a general description of the neutron transport equation and its application in numerical problems. Following the transport equation, a description of the spatial discretization schemes and their limitations is provided for the Diamond Difference, Step and the Linear Characteristic methods.

## General Overview of Neutron Transport Theory

To begin our development of the neutron transport approach to modeling a reactor, one must first examine the neutron transport equation and the simplifying assumptions necessary to treat it numerically. The fundamental equation describing neutron particle transport is the linear Boltzmann transport equation (BTE) (Lewis and Miller, 1993). The generalized form of the time-independent equation is as follows:

$$[\hat{\Omega} \cdot \vec{\nabla} + \sigma(\vec{r}, E)]\Psi(\vec{r}, \hat{\Omega}, E) = S(\vec{r}, \hat{\Omega}, E). \tag{2.1}$$

The angular neutron flux density is represented by $\Psi$ and is a function of its position at point $\vec{r}$, moving with an energy E, in a given direction $\hat{\Omega}$. The streaming term $\hat{\Omega} \cdot \vec{\nabla}\Psi(\vec{r}, \hat{\Omega}, E)$ and the collision term $\sigma(\vec{r}, E)\Psi(\vec{r}, \hat{\Omega}, E)$ account for particles lost out of a given phase-space. The macroscopic cross section $\sigma(\vec{r}, E)$ represents the total probability of interaction via any collision within the volume for energy E[1]. The source term, $S(\vec{r}, \hat{\Omega}, E)$, accounts for all particles born in the phase space $(\vec{r}, \hat{\Omega}, E)$. The BTE describes the balance of neutral particle flow in three spatial dimensions ($\vec{r}$), two angular

[1] In neutron transport, the macroscopic cross section is given the symbol $\sigma$ instead of $\Sigma$ in order to prevent confusion with the summation symbol associated with numerical integration. (Lewis and Miller, 1993)

dimensions ($\hat{\Omega}$), and energy (E). For the case of Cartesian coordinates, the streaming operator term is represented as:

$$\hat{\Omega}\cdot\vec{\nabla}\Psi = \mu\frac{\partial\Psi}{\partial x} + \eta\frac{\partial\Psi}{\partial y} + \xi\frac{\partial\Psi}{\partial z} \tag{2.2}$$

where the direction cosines of the Cartesian axes are $\mu{=}\hat{\Omega}\cdot\hat{e}_x$, $\eta{=}\hat{\Omega}\cdot\hat{e}_y$, and $\xi{=}\hat{\Omega}\cdot\hat{e}_z$. Figure 2.1 shows the orientation of the direction cosines to the particle streaming direction $\hat{\Omega}$ (Lewis and Miller, 1993).

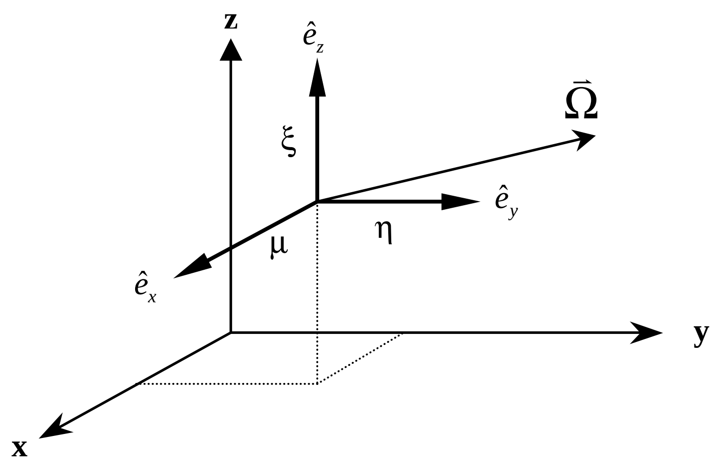

**Figure 2.1: Cartesian Coordinate (x, y, z) Vector Relations**

The solution to equation (2.1) is generally found by integrating over the variables $(\bar{r}, \hat{\Omega}, \text{ E})$. By integrating the BTE over all angles, one gets:

$$\phi(\bar{r}, \text{E}) = \int d\Omega\ \Psi(\bar{r}, \hat{\Omega}, \text{E})\,. \tag{2.3}$$

To approximate this integral numerically, use the following quadrature rule:

$$\phi_g(\bar{r}, \text{E}_g) = \sum_{n=1}^{N} w_n \Psi_{n,g}(\bar{r}, \text{E}_g) \tag{2.4}$$

where the summation is carried out over N ordinates for a given energy group *g*. For reflection boundary conditions, the quadrature equation (2.4) must contain an even number of weights symmetrically distributed about the origin of the unit directional sphere. The symmetry allows angular flux reflections within already chosen angular ordinates that provide an easy method for handling reflection boundary conditions (See Appendix B for boundary conditions treatment). To approximate this quadrature set, level symmetric quadrature weights are used because they meet all necessary conditions for symmetry in Cartesian coordinates (Lewis and Miller 1993) (Carlson, 1971). There are many other quadrature sets available such as the Gauss-Chebyshev quadrature and the Composite Midpoint quadrature (Atkinson, 1989); however, a Level Symmetric quadrature set was chosen because it has been proven to be stable, accurate, and easy to program in multi-dimensional Cartesian coordinates problems (Lewis and Miller 1993) (Carlson, 1971)[2]. Further discussion of the angular quadrature treatment is given in Appendix A.

The integration of the BTE over energy is performed by dividing the neutron energy spectrum into G energy groups (Lewis and Miller, 1993). Due to the linearity of the Boltzmann Transport Equation, one can separate out each energy dependent angular flux distribution and represent it as follows:

$$\Psi_g(\bar{\mathrm{r}}, \hat{\Omega}) = \int_g \mathrm{dE}\Psi(\bar{\mathrm{r}}, \hat{\Omega}, \mathrm{E}) \tag{2.5}$$

or for multi-energy groups:

$$\int_g dE = \int_{E_g}^{E_{g-1}} dE \tag{2.6}$$

In this representation, the lowest energy is $E_G$ and the highest is $E_0$. The total angular flux is found as follows:

[2] The reader is referred to the references for further discussion on why level symmetric quadratures work well in three-dimensional space; especially with reflective boundary conditions.

$$\Psi(\vec{r}, \hat{\Omega}) = \sum_{g=1}^{G} \int_{g} dE \Psi(\vec{r}, \hat{\Omega}, E) \tag{2.7}$$

The code developed in this research uses a multi-energy group representation so that after the angular integration is performed over all energy groups, one has the following relation:

$$\phi(\vec{r}) = \phi_1(\vec{r}) + \phi_2(\vec{r}) + \ldots + \phi_G(\vec{r}) \tag{2.8}$$

where $\phi_1(\vec{r})$ represents the fast energy group neutron flux (greater than 1.0 MeV) and $\phi_2(\vec{r})$ through $\phi_G(\vec{r})$ represent the thermal energy groups (less than 1.0 MeV).

The final integration over all space is performed by discretizing the problem domain into a structured or unstructured grid that divides the spatial regions efficiently. The BTE is calculated over each cell in the grid and point values are found at the center of each cell. For ease of modeling, a structured cubic or parallelepiped cell grid was chosen to represent the three-dimensional phase space. The spatial quadrature sets used in this research are explained in the next section. This finishes our discussion of the left hand side of the BTE.

The right hand side of the BTE consists of the source. The source can be represented in many ways: uniform, isotropic scatter, anisotropic scatter, fission or any combination of these as appropriate to the problem of interest. In general, the time-independent source with fission term has the following form:

$$\begin{aligned} S(\vec{r}, \hat{\Omega}, E) = {} & q_{ext}(\vec{r}, \hat{\Omega}, E) \\ & + \int dE' \int d\hat{\Omega}' \sigma_s(\vec{r}, E' \to E, \hat{\Omega}' \cdot \hat{\Omega}) \Psi(\vec{r}, \hat{\Omega}', E') \\ & + \frac{\chi(E)}{k} \int dE' \nu\sigma_f(\vec{r}, E') \int d\hat{\Omega}' \Psi(\vec{r}, \hat{\Omega}', E') \end{aligned} \tag{2.9}$$

where the first term represents all external sources, the second term represents the scattering source using the differential scatter cross section, and the third term represents the fission source without delayed neutrons (see Lewis and Miller, 1993 for a more detailed discussion). The fission term contains the fission energy distribution function $\chi(E)$ and the eigenvalue k for reactor criticality. For this study, the RHS of the BTE was simplified from its general form assuming isotropic scatter and fission sources. The simplification involves integrating out the angular dependence, reducing the right hand side to:

$$\begin{aligned} S(\bar{r}, E) = q_{ext}(\bar{r}, E) + \sigma_s(\bar{r}, E)\phi(\bar{r}, E) + \\ \frac{\chi(E)}{k}\nu\sigma_f(\bar{r}, E)\phi(\bar{r}, E) \end{aligned} \tag{2.10}$$

where $\sigma_s$ is the isotropic scatter cross section (consisting of all scatters into energy E and any "within group" scattering) and $\sigma_f$ is the isotropic fission cross section. Equation (2.10) represents the source used for modeling a simple isotropic reactor. Further simplifications were made to develop a simplistic source with no fission to aid in testing the spatial quadratures used in the 3D-TRAN code. The form of the source is as follows:

$$S(\bar{r}, E) = c\sigma_t\phi(\bar{r}, E) + S_{ext}(\bar{r}, E) \tag{2.11}$$

where $S_{ext}$ is a uniform constant source for a given energy, c is the scatter fraction (ranging from 0 to 1 or full absorption to full scatter) and $\sigma_t$ is the total cross section. This form of the source reduces the iteration process to a simple calculation with convergence on the scalar flux, eliminating the need for any eigenvalue search calculations.

The iterative approach to solving the BTE numerically is called the Von Neumann iteration on the source (Lewis ands Miller, 1993) (Lathrop, 1965). This method assumes a form of the source as the first guess for the RHS of the BTE and then calculates the LHS of the BTE. The source is updated and the LHS is calculated. This continues until convergence is achieved; therefore, the BTE can be represented as an iterative equation as follows:

$$[\hat{\Omega}\cdot\bar{\nabla}+\sigma(\bar{r},\,E)]\Psi^{m+1}(\bar{r},\,\hat{\Omega},\;E)=S^{m}(\bar{r},\,\hat{\Omega},\;E) \tag{2.12}$$

where m represents the iteration number. The convergence of the solution to the BTE can be performed either on the scalar flux (the traditional way) or the eigenvalue if calculating on a fission source. For eigenvalue convergence, the Power Iteration Method is generally the accepted procedure for generating and reaching convergence to a solution of the eigenvalue (Lewis and Miller, 1993) (Harman, 2001) (Duderstadt, 1976). This method is well known and will not be explained any further.

## Spatial Quadratures

The spatial quadratures used in the 3D-TRAN code are the Diamond Difference method, Step, and the Linear Characteristic Method. Each of these methods is developed from a basic understanding of the neutron balance equation for a given unit cell. For this research, a structured cubic or parallelepiped mesh was used as the shape of the basic computational cell. The following sections discuss the derivations of the balance equation and the subsequent spatial quadrature methods.

Neutron Balance for the cubic cell

The neutron balance equations are derived for the smallest computational unit in the problem spatial mesh, a cubic cell with sides of length Δx, Δy, and Δz. Figure 2.2 below shows the orientation of the single mesh cell.

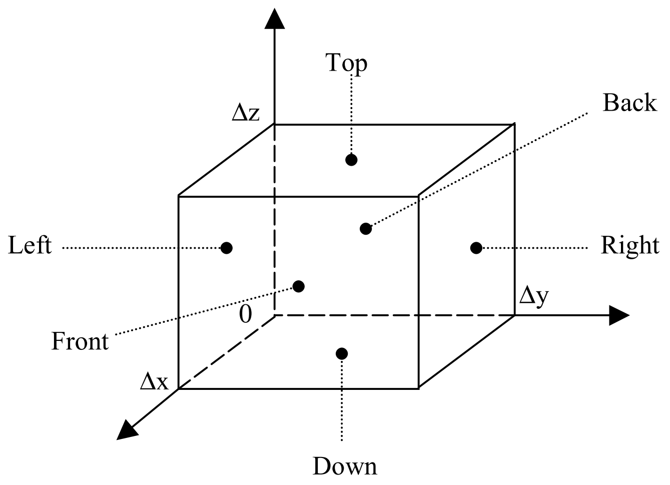

**Figure 2.2: Computational Mesh Cell**

Each face is labeled in order to reference entering and exiting fluxes along the faces of the mesh cell. The zeroth balance equation[3] is found by integrating the BTE, equation (2.1) with the equation (2.2) applied, over the unit mesh volume or $\iiint dxdydz$ . The following notation is used to simplify and reference the fluxes along the face:

$$\begin{aligned}\text{Back Face} &\Rightarrow \Psi_{\mathrm{B}} = \int_0^{\Delta y} \frac{dy}{\Delta y} \int_0^{\Delta z} \frac{dz}{\Delta z} \Psi(0, y, z) \\ \text{Front Face} &\Rightarrow \Psi_{\mathrm{F}} = \int_0^{\Delta y} \frac{dy}{\Delta y} \int_0^{\Delta z} \frac{dz}{\Delta z} \Psi(\Delta x, y, z)\end{aligned} \tag{2.13}$$

$$\begin{aligned}\text{Down Face} &\Rightarrow \Psi_{\mathrm{D}} = \int_0^{\Delta x} \frac{dx}{\Delta x} \int_0^{\Delta y} \frac{dy}{\Delta y} \Psi(x, y, 0) \\ \text{Top Face} &\Rightarrow \Psi_{\mathrm{T}} = \int_0^{\Delta x} \frac{dx}{\Delta x} \int_0^{\Delta y} \frac{dy}{\Delta y} \Psi(x, y, \Delta z)\end{aligned} \tag{2.14}$$

[3] The zeroth balance equation represents the neutron conservation relationship between incoming and exiting average flux values over the unit cell based on the geometry chosen.

$$\begin{aligned}&\text{Left Face } \Rightarrow \ \Psi_{\mathrm{L}} = \int_0^{\Delta x} \frac{dx}{\Delta x} \int_0^{\Delta z} \frac{dz}{\Delta z} \Psi(x,0,z) \\ &\text{Right Face } \Rightarrow \ \Psi_{\mathrm{R}} = \int_0^{\Delta x} \frac{dx}{\Delta x} \int_0^{\Delta z} \frac{dz}{\Delta z} \Psi(x,\Delta y,z)\end{aligned} \tag{2.15}$$

The division by Δx, Δy, and Δz in each of the above equations transforms the arbitrary (x, y, z) mesh to a unit cell with sides of length one with corresponding transformed axes (u, v, w). The fluxes represent the average flux over the faces and can be numerically approximated. The simplified zeroth balance equation then becomes:

$$\alpha_{yx}(\Psi_{\mathrm{F}} - \Psi_{\mathrm{B}}) + \Psi_{\mathrm{R}} - \Psi_{\mathrm{L}} + \alpha_{yz}(\Psi_{\mathrm{T}} - \Psi_{\mathrm{D}}) + \varepsilon_y \Psi_{\mathrm{A}} = \frac{\Delta y}{\eta} S_{\mathrm{A}} \tag{2.16}$$

where,

$$\varepsilon_x = \frac{\sigma_t \Delta x}{\mu}; \quad \varepsilon_y = \frac{\sigma_t \Delta y}{\eta}; \quad \varepsilon_z = \frac{\sigma_t \Delta z}{\xi} \tag{2.17}$$

$$\alpha_{yx} = \frac{\varepsilon_y}{\varepsilon_x}; \quad \alpha_{yz} = \frac{\varepsilon_y}{\varepsilon_z} \tag{2.18}$$

$$\Psi_{\mathrm{A}} = \int_0^{\Delta x} \frac{dx}{\Delta x} \int_0^{\Delta y} \frac{dy}{\Delta y} \int_0^{\Delta x} \frac{dz}{\Delta z} \Psi(x,y,z) \tag{2.19}$$

$$S_{\mathrm{A}} = \int_0^{\Delta x} \frac{dx}{\Delta x} \int_0^{\Delta y} \frac{dy}{\Delta y} \int_0^{\Delta x} \frac{dz}{\Delta z} S(x,y,z) \tag{2.20}$$

The source has been simplified as a function called S(x,y,z) for derivation purposes with both $S_A$ representing the average source and $\Psi_A$ representing the average flux or zeroth moment within the cell volume. The energy dependence is ignored for now because each of these equations applies to all energy groups of interest (assuming proper energy dependent material properties are substituted appropriately in each equation). Equation (2.16) serves as the basis for numerically finding the average flux in a mesh of unit computational cells. For particle flow into and out of the cell, the flux is assumed to enter through the back, down, and left faces and exit through the front, top, and right faces (see figure 2.3). This fundamental orientation of fluxes across the unit mesh cell faces is maintained for all transport calculations. The cell itself is allowed to rotate and reorient itself to the direction of particle flow determined by each angular ordinate in the $S_n$ set. The final set of balance equations needed correspond to the first moments across the unit cell.

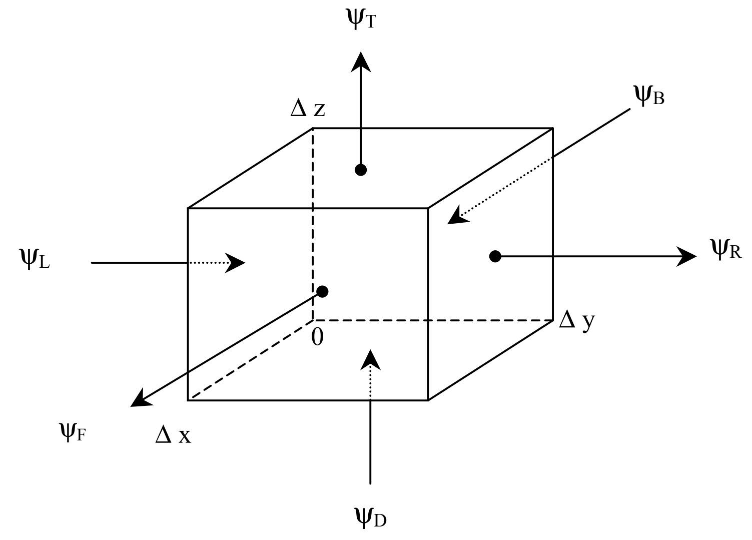

**Figure 2.3: Inflow and Outflow Fluxes of the Unit Mesh Cell**

The first moment of the flux is found by multiplying the BTE by the first Legendre polynomial then integrating over the volume of the unit cell[4]. The $P_1(x)$ Legendre polynomial is scaled and centralized about the center of the cell volume or the center of the cell face. This representation allows for easy interpretation of the first moments in a cell and follows the traditional derivations found in the literature (Lewis and Miller, 1993) (Carlson and Lathrop, 1965). The advantage to central based first

[4] The first moment balance equations are only used by the Linear Characteristic method. Since this method assumes a linear distribution, one uses the modified, centralized first Legendre polynomial as an approximation to the assumed form of the neutron distribution within the unit cell and along the cell faces.

moments is that they correspond with the location of the zeroth cell average within the unit cell volume and along the cell face. The first moments are represented as follows:

$$\Psi_{\mathrm{x}} = \int_0^{\Delta x} \frac{dx}{\Delta x} \int_0^{\Delta y} \frac{dy}{\Delta y} \int_0^{\Delta x} \frac{dz}{\Delta z}\, 3\mathrm{P}_1(\mathrm{x})\Psi(x,y,z) \tag{2.21}$$

$$\Psi_{\mathrm{y}} = \int_0^{\Delta x} \frac{dx}{\Delta x} \int_0^{\Delta y} \frac{dy}{\Delta y} \int_0^{\Delta x} \frac{dz}{\Delta z}\, 3\mathrm{P}_1(\mathrm{y})\Psi(x,y,z) \tag{2.22}$$

$$\Psi_{\mathrm{z}} = \int_0^{\Delta x} \frac{dx}{\Delta x} \int_0^{\Delta y} \frac{dy}{\Delta y} \int_0^{\Delta x} \frac{dz}{\Delta z}\, 3\mathrm{P}_1(\mathrm{z})\Psi(x,y,z) \tag{2.23}$$

where the factor 3 is a constant designed to normalize the Legendre polynomial. Now, the flux moments about the back face on the unit cell are defined as follows:

$$\begin{aligned} \theta_{\mathrm{By}} &= \int_0^{\Delta y} \frac{dy}{\Delta y} \int_0^{\Delta x} \frac{dz}{\Delta z}\, 3\mathrm{P}_1(\mathrm{y})\Psi(0,y,z) \\ \theta_{\mathrm{Bz}} &= \int_0^{\Delta y} \frac{dy}{\Delta y} \int_0^{\Delta x} \frac{dz}{\Delta z}\, 3\mathrm{P}_1(\mathrm{z})\Psi(0,y,z) \end{aligned} \tag{2.24}$$

and the front face moments are:

$$\begin{aligned} \theta_{\mathrm{Fy}} &= \int_0^{\Delta y} \frac{dy}{\Delta y} \int_0^{\Delta x} \frac{dz}{\Delta z}\, 3\mathrm{P}_1(\mathrm{y})\Psi(\Delta x,y,z) \\ \theta_{\mathrm{Fz}} &= \int_0^{\Delta y} \frac{dy}{\Delta y} \int_0^{\Delta x} \frac{dz}{\Delta z}\, 3\mathrm{P}_1(\mathrm{z})\Psi(\Delta x,y,z) \end{aligned} \tag{2.25}$$

The same technique is applied to the down, top, left, and right faces with the corresponding x, y, or z axis swap performed as necessary[5]. The source also has first moments associated with it of the form:

[5] First moments about an inflow or outflow face are commonly represented by the symbol $\theta$ rather than $\Psi$. This notation helps avoid some confusion between face and volumetric flux moment values (Mathews, 1999)

$$S_{\mathrm{x}} = \int_0^{\Delta x} \frac{dx}{\Delta x} \int_0^{\Delta y} \frac{dy}{\Delta y} \int_0^{\Delta x} \frac{dz}{\Delta z}\, 3\mathrm{P}_1(x)\, S(x,y,z)$$
$$S_{\mathrm{y}} = \int_0^{\Delta x} \frac{dx}{\Delta x} \int_0^{\Delta y} \frac{dy}{\Delta y} \int_0^{\Delta x} \frac{dz}{\Delta z}\, 3\mathrm{P}_1(y)\, S(x,y,z) \qquad (2.26)$$
$$S_{\mathrm{z}} = \int_0^{\Delta x} \frac{dx}{\Delta x} \int_0^{\Delta y} \frac{dy}{\Delta y} \int_0^{\Delta x} \frac{dz}{\Delta z}\, 3\mathrm{P}_1(z)\, S(x,y,z)$$

The first volumetric moments within the cell are similar to the face moments, but defined as follows:

$$\Psi_{\mathrm{x}} = \int_0^{\Delta x} \frac{dx}{\Delta x} \int_0^{\Delta y} \frac{dy}{\Delta y} \int_0^{\Delta x} \frac{dz}{\Delta z}\, 3\mathrm{P}_1(x)\, \Psi(x,y,z)$$
$$\Psi_{\mathrm{y}} = \int_0^{\Delta x} \frac{dx}{\Delta x} \int_0^{\Delta y} \frac{dy}{\Delta y} \int_0^{\Delta x} \frac{dz}{\Delta z}\, 3\mathrm{P}_1(y)\, \Psi(x,y,z) \qquad (2.27)$$
$$\Psi_{\mathrm{z}} = \int_0^{\Delta x} \frac{dx}{\Delta x} \int_0^{\Delta y} \frac{dy}{\Delta y} \int_0^{\Delta x} \frac{dz}{\Delta z}\, 3\mathrm{P}_1(z)\, \Psi(x,y,z)$$

After performing the necessary integrations and substitutions over the BTE, the first moment balance equations for x, y, and z for the unit cell are as follows:

$$3\alpha_{yx}(\Psi_{\mathrm{F}} + \Psi_{\mathrm{B}} - 2\Psi_{\mathrm{A}}) + \theta_{\mathrm{Rx}} - \theta_{\mathrm{Lx}} + \alpha_{yz}(\theta_{\mathrm{Tx}} - \theta_{\mathrm{Dx}}) + \varepsilon_y \Psi_x = S_x \frac{\Delta y}{\eta} \qquad (2.28)$$

$$\alpha_{yx}(\theta_{\mathrm{Fy}} + \theta_{\mathrm{By}}) + 3(\Psi_{\mathrm{R}} + \Psi_{\mathrm{L}} - 2\Psi_{\mathrm{A}}) + \alpha_{yz}(\theta_{\mathrm{Ty}} - \theta_{\mathrm{Dy}}) + \varepsilon_y \Psi_y = S_y \frac{\Delta y}{\eta} \qquad (2.29)$$

$$\alpha_{yx}(\theta_{\mathrm{Fz}} + \theta_{\mathrm{Bz}}) + \theta_{\mathrm{Rz}} - \theta_{\mathrm{Lz}} + 3\alpha_{yz}(\Psi_{\mathrm{T}} - \Psi_{\mathrm{D}} - 2\Psi_{\mathrm{A}}) + \varepsilon_y \Psi_z = S_{\mathrm{z}} \frac{\Delta y}{\eta}\ . \qquad (2.30)$$

Now we have all the equations necessary to check for neutron balance in a three dimensional, cubic computational cell.

The balance equations are used along with required auxiliary equations in order to solve for all unknown fluxes in the unit cell. The auxiliary equations are based on assumptions that constitute a type of spatial quadrature. The spatial quadratures of interest in this project are the Diamond Difference Method and the Step method, both of which require the zeroth balance equation only. The final spatial quadrature, the Linear Characteristic Method, requires the first moment balance equations as well as the zeroth balance equation to check for neutron balance over the unit cell. These methods are described in more detail in the following pages.

## Diamond Difference Method

The Diamond Difference (DD) technique is the most common method used in discrete ordinates because of its simplicity and easy implementation. It assumes a set of auxiliary equations that perform an average to the flux across the inflow and outflow faces of a unit cell (Lewis and Miller, 1993) (O'Dell, 1987). These equations reduce the number of unknowns in the zeroth balance equation and allow for generation of a simple solution to the average flux inside the cell. For three dimensions (referencing the configuration of figure 2), the auxiliary equations are:

$$\Psi_A = \frac{\Psi_L + \Psi_R}{2} = \frac{\Psi_F + \Psi_B}{2} = \frac{\Psi_T + \Psi_D}{2} \tag{2.31}$$

where the known values (inflow fluxes) are $\psi_L$, $\psi_B$, and $\psi_D$. The inflow fluxes are known either from boundary conditions or from the outflow fluxes of an adjoining cell; therefore, the iteration on this method (as well as the other discrete ordinates methods presented in this research) must begin at a corner of the problem mesh where the three inflow values to the unit cell are specified and walk into the interior of the problem. The unknown of interest for the discrete ordinates methods is usually the average flux within the cell. After substituting equation (2.31) into the balance equation (2.16), the DD equation for the average flux in the cell becomes:

$$\Psi_{\mathrm{A}} = \frac{S_{\mathrm{A}}\dfrac{\Delta y}{\eta} + 2\alpha_{yx}\Psi_{\mathrm{B}} + 2\alpha_{yz}\Psi_{\mathrm{D}} + 2\Psi_{\mathrm{L}}}{2(\alpha_{yx} + \alpha_{yz} + 1) + \varepsilon_{y}}. \tag{2.32}$$

The DD method has second order convergence and is a linear-continuous method (O'Dell, 1987), but can produce negative fluxes. The negative fluxes are caused by the averaging technique used to generate the method. As cell widths become large and or material cross sections become large, the calculation of the average flux in equation (2.32) can produce a positive result; however, when the outflow fluxes are calculated from equations (2.31), the result can yield negative fluxes on the outflow faces[6]. The next cell in line for computation will then start with negative inflow fluxes. The end result is that the average flux will become negative, or it will oscillate about zero if the flux is small (Mathews, 1999). Therefore, DD works well for small mesh sizes and moderate to low density materials, but the mesh must be carefully examined for negative fluxes and adjusted finer as material density increases. This makes large-scale reactor problems difficult to model on desktop computers because of the large material mesh sizes required and the relatively limited memory (RAM) space available for calculations to run efficiently.

## Step Method

The Step method uses an assumption that the source remains constant within a computational cell. This method avoids the negative fluxes commonly found with the Diamond Difference method, but only has linear convergence. The Step method is very

[6] Negative fluxes are produced when the average flux calculated in the cell is less than half the value of the incoming flux.

easy to implement because the auxiliary equations approximate the outgoing fluxes equal to the cell average flux. These equations are as follows:

$$\Psi_{\mathrm{A}} = \Psi_{\mathrm{F}} = \Psi_{\mathrm{R}} = \Psi_{\mathrm{T}} . \tag{2.33}$$

When these assumptions are substituted into the zeroth balance equation (2.16), the average cell flux is calculated as:

$$\Psi_{\mathrm{A}} = \frac{S_{\mathrm{A}} \frac{\Delta y}{\eta} + \alpha_{yx} \Psi_{\mathrm{B}} + \alpha_{yz} \Psi_{\mathrm{D}} + \Psi_{\mathrm{L}}}{\alpha_{yx} + \alpha_{yz} + 1 + \varepsilon_{y}} . \tag{2.34}$$

The outgoing fluxes are then calculated by using equations (2.33). The Step method works very well for most applications, but has a tendency to overestimate flux profiles in problems of interest (Mathews, 1999). The use of the auxiliary equations tends to smear the flux over the computational cell and can often reduce or dampen flux variations in larger computational cells, especially near material boundaries. The Step method is very fast in implementation because it uses only four simple equations to calculate flux in each unit cell. The positivity associated with the method also makes it a very useful technique for producing rough, first-look estimates of general flux shapes in more complicated problems. This method also makes a good initial guess generator for more complicated methods such as the Linear Characteristic method discussed in the next section.

## Linear Characteristic Method

The Linear Characteristic (LC) spatial quadrature method, like all characteristic methods, assumes a generalized form of the source distribution within a computational unit cell (Mathews, Miller, and Brennan, 2000). The LC functional form of the source within the cell is:

$$S(u,v,w) = \mathrm{A} + \mathrm{B_u}u + \mathrm{B_v}v + \mathrm{B_w}w \qquad (2.35)$$

where the coordinates (*u, v, w*) represent any right-handed orthogonal based three-dimensional coordinate system. The coefficients A, $\mathrm{B_u}$, $\mathrm{B_v}$, and $\mathrm{B_w}$ must be determined for each cell to define the linear source distribution function. These coefficients are found by using the known zeroth and first moments of the source within the cell as well as the assumed linear distribution of the flux across an entering cell face. These values are known either by initial conditions or are defined from a previous iteration. With the source distribution defined, the zeroth and first moments of the fluxes can be calculated within the cell as well as along the exiting cell faces. This process is more complicated than the Diamond Difference and Step methods presented earlier and will require a closer examination of the orientation and subdivision of the computational cell.

In general, the characteristic methods take into account the direction of travel of neutrons within the computational cell. The directional ordinate of the angular integration defines the direction of travel of the neutrons. Integration along the particle directional path is performed to sum up all contributions of particles born from the within cell source as well as summing the particle contribution entering the cell from surrounding cells in the mesh. Since the direction of travel is important in characteristic methods, the orientation of the unit cell to the particle directional flow is critical to the successfully implementation of this method. For this research, a cubic or parallelepiped structured mesh[7] is assumed as in figure 2.2 with fluxes entering and leaving the cell as in figure 2.3. For a given direction ordinate, the unit cell is broken down into as many as eleven sub cells defined by a given directional ordinate vector (Suriano, 2001). Figure 2.4 below

[7] In this research, a well-structured mesh implies no overlap of cell faces between adjoining cells. Or in other words, the exiting faces of the cell directly line up with the entering faces of adjoining cells.

illustrates the subdivision of the unit cell. For any off-diagonal directional ordinate, the unit cell is broken down into four prisms, six tetrahedrons and one parallelepiped. If a direction ordinate falls along the cube diagonal, the unit cell is only broken down into six tetrahedrons. Therefore, our three-dimensional problem is reduced to as many as eleven one-dimensional calculations along the streaming direction. At first glance, there appears to be eleven different sets of equations to derive for each ordinate in the $S_n$ set; however, one can use a simple Jacobian transformation technique[8] that converts our eleven sub volumes into three basic unit volumes, the tetrahedron, the prism, and the parallelepiped. The Linear Characteristic equations are defined for each of the three sub volumes. For brevity, the equations for each volume are derived in Appendix C.

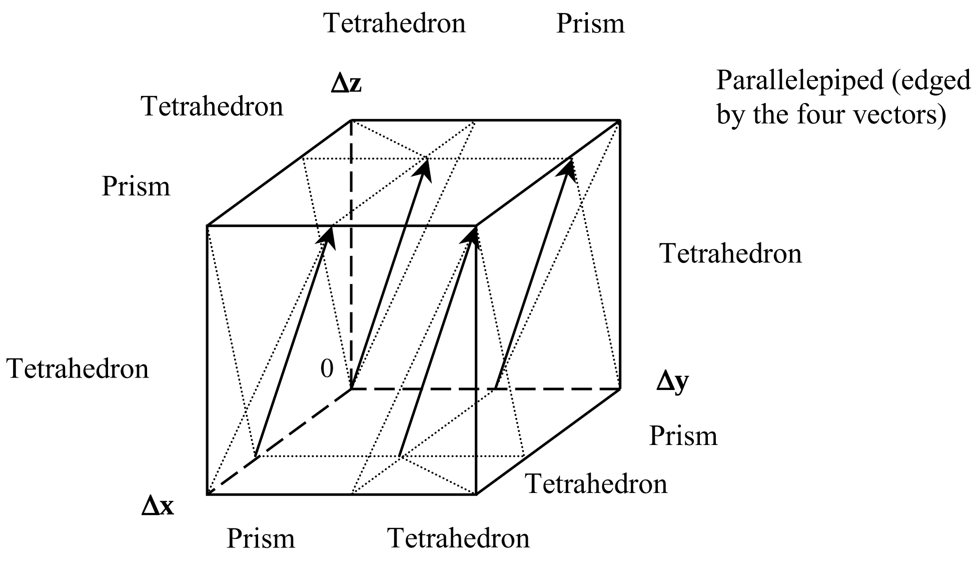

**Figure 2.4: Direction Ordinate Cell Splitting Technique**

After the unit cell is broken down into at most eleven sub volumes, the average zeroth and first moment fluxes are computed in each sub cell as well as across the exiting

[8] The LC tetrahedron paper by Mathews, Miller, and Brennan (2000) outlines the Jacobian transformation technique in detail.

face of the sub cell volume. The fluxes are transformed back to the orientation of the global unit cell and recombined using volumetric and area weighted averages based on the global unit cell volume and outflow face areas. The process continues with the next unit cell in the array until all cells are calculated in the spatial mesh. A more descriptive overview of the Jacobian transformation process required for the LC method is provided in Appendix D.

The LC method is a linear iterative method with at least second order convergence or higher in three dimensions (Brennan, 1996). The LC method is considered to be more stable than Diamond Difference and does not suffer from synthetic numerical dispersion when propagating particles along given directional paths (Mathews, 1999); however, LC does not have strict positivity. It can and will produce negative fluxes in numerical situations when the optical thickness of the cell is too great. The advantage LC has over Diamond Difference is that the characteristic integrations performed carry both zeroth and first moment information about the neutron flux. The addition of first moment information allows LC to be calculated over larger optical thicknesses compared to the same Diamond Difference stable meshing requirements. Therefore, larger computational cells can be used in the mesh thereby reducing the total mesh size and computer memory requirements needed to perform the calculations (Mathews, 1999).

The reduction of mesh sizes comes at a great cost. The number of LC computations required per unit cell are far greater than those of Diamond Difference or Step and dramatically increase code run times from milliseconds to minutes. This undesired effect is nullified by the fact that Diamond Difference requires a finer mesh array for stability thereby increasing computer memory requirements exponentially. A point is reached in the computations of a finer mesh where the computer uses hard disk space as virtual memory and begins writing data back and forth from RAM to disk. The communication process required for this to happen drastically slows down the computer and can cause an increase in code run time from seconds to hours. Therefore, the ability to run the same problem over a coarser mesh in a few minutes using the Linear Characteristic method suddenly becomes very desirable. An analysis of this effect is given in Chapter 3 by comparing all spatial methods used in this research.

# Chapter III: Performance and Evaluation

The following chapter summarizes the results of the performance tests used to evaluate the 3D-TRAN code for proper implementation of the theory discussed in Chapter 2. These tests include convergence testing over refined meshes, angular quadrature comparisons, heterogeneous mesh testing, and timing studies. All results are summarized and used to describe key features of the transport techniques used in the 3D-TRAN code.

## Convergence Testing

The success of numerical methods depends on their ability to approach the correct answer represented by the equations being modeled. A properly implemented numerical method should converge to a specific answer as problem parameters such as mesh size are refined. This fact stems from the ability of a good numerical method to converge to the true solution in the limit as the discretized parameters become small and approximate more closely a continuous medium. For the transport methods of Chapter 2, this property should also hold true. Therefore, the first series of tests involves testing the refinement of the spatial mesh grid and its effects on convergence.

The Analogue Monte Carlo statistical method is considered the best estimate of the solution to the transport equation for all comparisons of the discrete ordinates techniques since it assumes a continuous angle, space, and energy flux distribution (Lewis and Miller, 1993). In order to test the implementation of the discrete ordinates spatial quadrature, a more direct comparison is needed. The development of a Monte Carlo $S_n$ code by Suriano (2001) allowed for testing of the error in the spatial quadrature while essentially ignoring the error in the angular quadrature. This is accomplished by forcing Monte Carlo angle draws along the discrete ordinates rays defined by the angular quadrature set. Essentially, the Monte Carlo is modified to be continuous in space but discrete in angle thereby duplicating the discretization of the angular integral performed in discrete ordinates. This capability makes MCSN a suitable benchmark[1] and it allows for checking the order of convergence of the spatial quadrature to see if the implemented techniques meet or exceed their documented convergence criteria.

The first test performed comes from the literature (Miller, 1998) (Brennan, 1996). The test problem consists of a 1.0 cm edged cube with vacuum boundary conditions on all sides. The cube contains a uniform source of 1.0 neutron/cm$^3$/sec and a total cross section of 1.0 cm$^{-1}$ with a scatter fraction of 0.5. The Monte Carlo (MC) and Monte Carlo $S_n$ (MCSN) techniques were run on the cube with no spatial discretization to form the

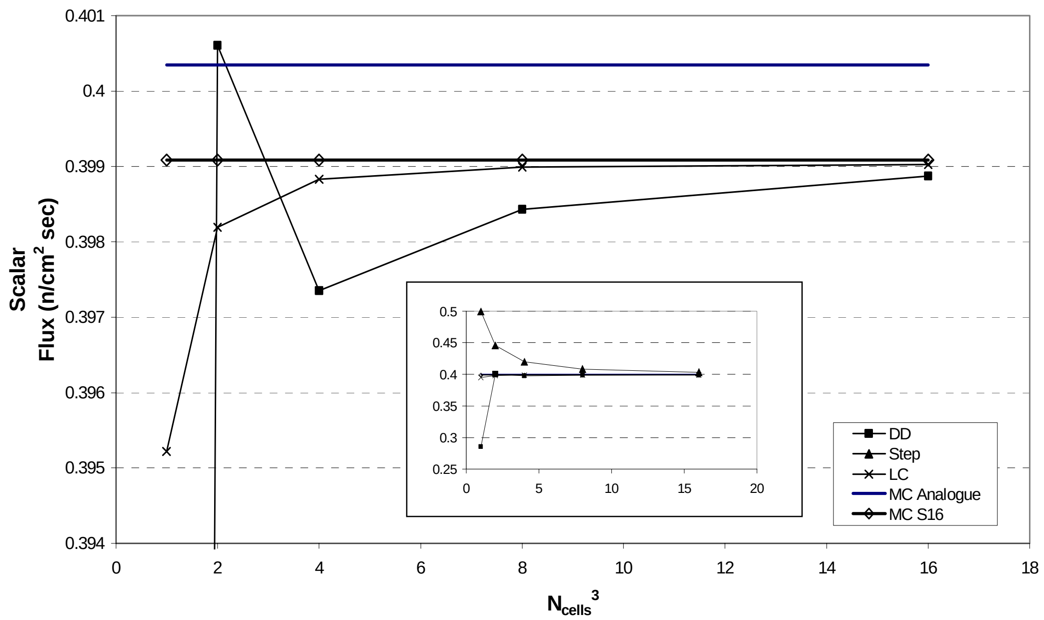

**Figure 3.1: Spatial Quadrature Convergence Test for 1.0 cm Edge Cube**

[1] A high number of particle draws were required to reduce the statistical error associated with Monte Carlo to a standard deviation of less than $10^{-5}$. This criteria was used for all Monte Carlo test runs.

benchmark. The spatial quadratures consisting of Diamond Difference (DD), Step, and Linear Characteristic (LC) were run starting at one spatial mesh cell. The spatial mesh was refined by a factor of two for each subsequent run. The results using the most refined angular quadrature set, $S_{16}$, are given in figure 3.1.

The flux convergence profile[2] for each spatial quadrature shows that as the mesh is refined, convergence to the MCSN answer is achieved as expected. At the dimensions of a 1.0 cm cube, the DD and LC methods converge rapidly to the MCSN benchmark as the mesh is refined. The Step method still has not converged as well as the other methods as indicated in the enlarged graph plotted over a reduced scalar flux range for clarity. The LC method performs the best as it quickly approaches the benchmark answer within one spatial refinement of the mesh.

The convergence of the spatial quadratures provided in figure 3.1 shows proper convergence behavior in the transport solution to the scalar flux, but shows nothing about the order of convergence in the spatial quadratures. The order of convergence can be examined by plotting the absolute relative error (as compared to $MCS_{16}$) against the mesh refinement factor on a logarithmic plot. The order of convergence is found by comparing the slope of the line made by the data for each method. This behavior is shown in Figure 3.2. This graph shows that all methods are following their prescribed order of convergence as stated in Chapter 2. The linear step and second order LC methods both converge linearly (on a Log-Log plot) as the errors are reduced with each refinement in the mesh cells. The convergence of the DD method varies in coarser meshes, but

[2] The flux represented in Figure 3.1 is the value of the flux over the entire cube. This value is found by calculating the volume weighted average flux over the entire mesh. This way, as the mesh is refined and many flux values are calculated, a global flux value is generated for the cube for comparison with the Monte Carlo continuous space answer and proper convergence behavior can be observed.

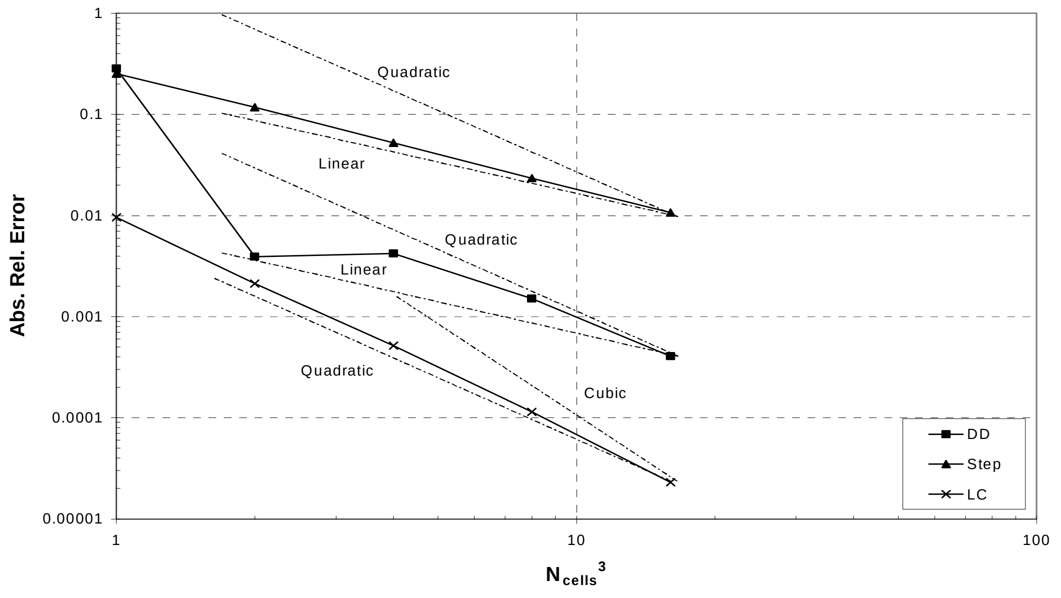

**Figure 3.2: Order of Convergence of Spatial Quadratures**

eventually settles in on quadratic convergence in the refined spatial meshes. This erratic convergence behavior is inherent in DD and becomes increasingly unstable as the spatial mesh coarsens and optical thickness increases. Therefore, DD only performs well under finer mesh conditions.

A comparison of the relative error in figure 3.2 suggests that LC out performs both DD and step in converging to the benchmark. The error in the LC method is more than a factor of 10 lower than DD and more than a factor of 100 lower than Step. In the test case above, a comparison of the LC method's relative error over a given mesh suggests that LC performs about the same as DD at a refinement factor of four more cells per LC cell and 16 more cells per LC cell for Step. Although these results are problem specific, the magnitude of the mesh refinement comparisons between LC and DD/Step appears to hold for all test cases run; therefore, these results suggest that LC should be used over a less refined spatial grid when compared to DD and Step to achieve lower relative errors in the scalar flux. The use of the LC method will also keep problem material mesh sizes lower compared to DD and Step which becomes very important when running problems on memory limited computers.

The convergence testing was repeated for all angular quadratures ($S_2$, $S_4$, $S_6$, $S_8$, $S_{16}$) with the same results and behavior as those presented in figure 3.1 and 3.2. The test problem was also modified to a 10 cm cube and a 100 cm cube to check convergence over large optically thick cells. Again, the LC method continued to out perform Step and DD in calculating scalar flux in reduced mesh sizes.  The DD method also showed increased instability in convergence in the coarser meshes proving that DD is relatively unreliable for meshes with large cells. Step performed very well over large meshes with stable convergence and relative errors out performing those of DD. The results of these tests are not shown because they essentially duplicate the behaviors already displayed in figures 3.1 and 3.2 with the exception of increased convergence instability in the DD method over increasingly larger cells.

## Angular Quadrature Comparisons

The convergence of the transport equation is dependent on the type of spatial quadrature, its spatial mesh refinement and the type of angular quadrature used. The spatial quadrature was discussed in the previous section by eliminating the error present in the angular discretization using the MCSN benchmark code. In this section, the angular quadrature is discussed referencing the shape of the flux profile and the maximum and minimum errors present. The $S_{16}$ angular quadrature set was used as the benchmark for each spatial quadrature method because it represents the most refined quadrature set used in this research. At 288 angular ordinates, the use of $S_{16}$ in large spatial mesh problems will add to computer memory requirements needed to execute the code thereby slowing calculation time. The goal of the angular quadrature tests was to see how well lower order angular quadrature sets predicted the shape of a flux profile by examining the maximum and minimum relative errors compared to $S_{16}$.

The parameters for this test problem are listed in table 3.1. The angular quadrature was varied using $S_2$, $S_4$, $S_6$, $S_8$, and $S_{16}$. The flux profile of interest for each angular quadrature was taken from the center channel of cells running from one boundary, through the center of the cube to the other boundary. The one dimensional flux profile starting at the center of the cube is plotted in figure 3.3 using the LC spatial quadrature method. Figure 3.3 suggests that the flux profile is best approximated using any quadrature set higher than $S_4$. The $S_2$ quadrature set did not perform as well as the other quadratures because of the low number of directional ordinates (totaling eight) present in the set. The $S_4$ set triples the number of discretized angles thereby reducing the overall error significantly, but not as well as $S_6$ and $S_8$. These results show that $S_2$ should

| Problem Parameter | Value |
|---|---|
| Cube width | 2.75 cm |
| Cells per side | 11 |
| Source | 1.0 n/cm$^3$/sec |
| $\sigma_t$ | 1.0 cm$^{-1}$ |
| Scatter Fraction, c | 0.5 |
| Boundary Conditions | Vacuum |

**Table 3.1: Angular Quadrature Test Problem Parameters**

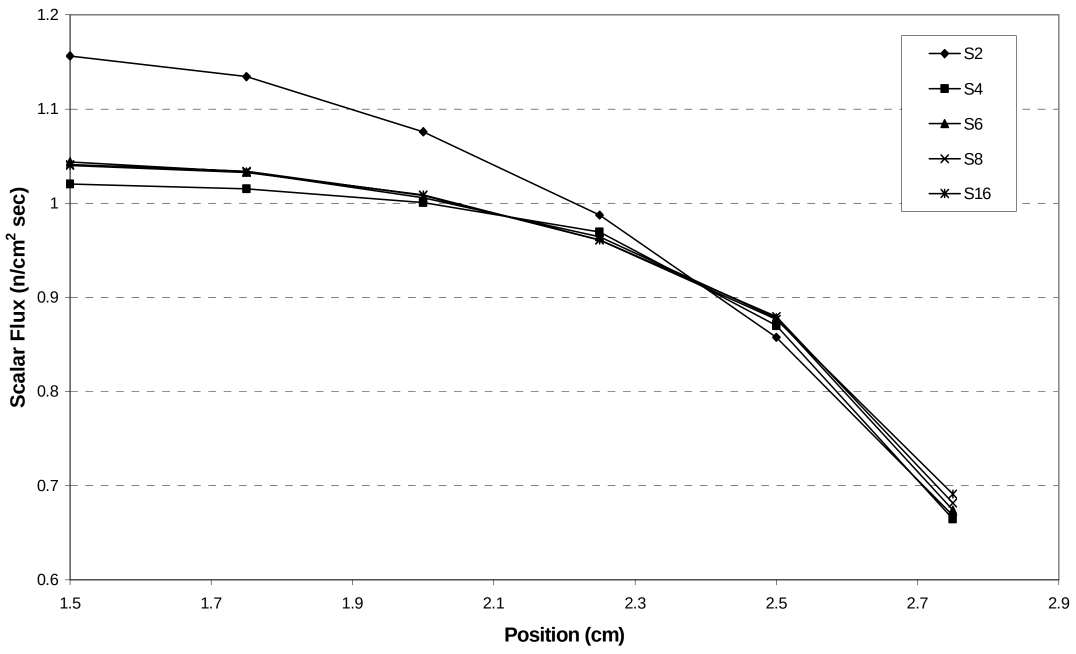

**Figure 3.3: Angular Quadrature Comparison of Scalar Flux using LC**

not be used for more realistic transport calculations where flux profile shapes are very important. The Step and DD spatial quadratures produced almost identical results to those in figure 3.3 showing that the $S_6$ quadrature seems to predict a flux shape that is fairly consistent with that produced by the $S_{16}$ quadrature.

The maximum and minimum relative error for the angular quadratures compared to $S_{16}$ is shown in Figure 3.4 grouped by spatial quadratures. This figure supports the analysis made on figure 3.3 by showing the relatively large errors for the $S_2$ quadrature for all three spatial methods. In most cases, an increase in the angular quadrature causes a decrease in the relative error. A comparison of figure 3.3 and figure 3.4 shows that the use of the $S_6$ quadrature will guarantee a good flux profile as well as a maximum error compared to $S_{16}$ of less than 3% or better. The $S_4$ quadrature can be used; however, it tends to under predict the maximum value of the flux compared to higher order angular quadrature sets. In most reactor calculations, the maximum flux value is very important

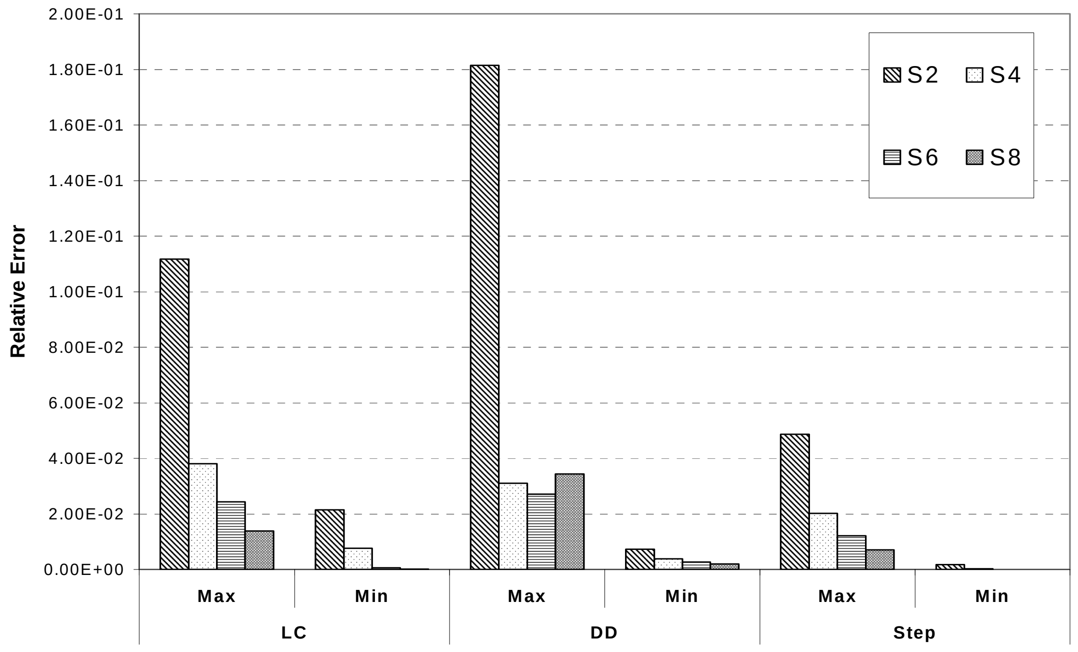

**Figure 3.4: Angular Quadrature Relative Error Comparison**

in identifying regional power spikes; therefore, at least the $S_6$ quadrature should be used. The extra computation time required by the $S_8$ (80 ordinates) and $S_{16}$ (288 ordinates) quadratures is not necessary since figures 3.3 and 3.4 show that $S_6$ (48 ordinates) produces flux profiles consistent with both $S_8$ and $S_{16}$.

## Heterogeneous Testing

The goal of this research was to develop a transport code capable of performing calculations on multiple region problems. This requires an initialization routine designed to properly fill numerical arrays that contain the correct material parameters corresponding to its spatial position in the problem. These arrays are then accessed in the correct order during the spatial walk calculation. In this section, the heterogeneous capability of the 3D-TRAN code is explored by performing a simple test, the traditional source/shield problem, and analyzing each spatial quadratures ability to handle optically thick computational cells. The test problem consists of two distinct slab regions with the following parameters:

1. Source region with S = 100 n/cm$^3$/sec
    a. Length = 3.0 cm in x direction, 0.75 cm in the y and z directions
    b. $\sigma_t$ = 1.0 cm$^{-1}$ and scatter fraction c = 0.5
    c. cell widths of 0.25 cm on each side
    d. Fixed region material for all tests
2. Absorbing region, S = 0 n/cm$^3$/sec
    a. Length = 1.5 cm in x direction, 0.75 cm in the y and z directions
    b. $\sigma_t$ = 1.0 cm$^{-1}$ and scatter fraction c = 0, pure absorption
    c. Cell widths of 0.25 cm on each side
    d. Variable region material – increase optical thickness by increasing $\sigma_t$
3. Boundary conditions[3]
    a. $X_{min}$ = Symmetry, $X_{max}$ = Vacuum
    b. $Y_{min} = Y_{max} = Z_{min} = Z_{max}$ = Symmetry.

[3] The use of symmetry boundary conditions in the y and z directions numerically forces constant flux values along these directions. The only variation in the flux profile exists in the x direction; therefore, a one dimensional "slab" calculation is simulated using a transport code designed for three dimensions.

The problem simulates infinite slab geometry in the y and z directions and semi-infinite slab geometry in the x direction. The flux profile was examined to check for the proper exponential decay behavior in the absorbing region. The results for all spatial quadratures using the $S_6$ angular quadrature are given in figure 3.5. Also, the Monte Carlo analogue and Monte Carlo $S_n$ runs are plotted for comparison of discrete ordinates to Monte Carlo methods.

The flux profile in figure 3.5 shows that all spatial quadratures compare well with the Monte Carlo predicted fluxes. The overall relative shape of the flux shows that the transport code is properly attenuating the neutrons in the absorber region characterized by the exponential decay towards the vacuum boundary on the right. The error in the spatial quadratures can be examined by looking at the MC $S_6$ flux profile[4]. The error analysis

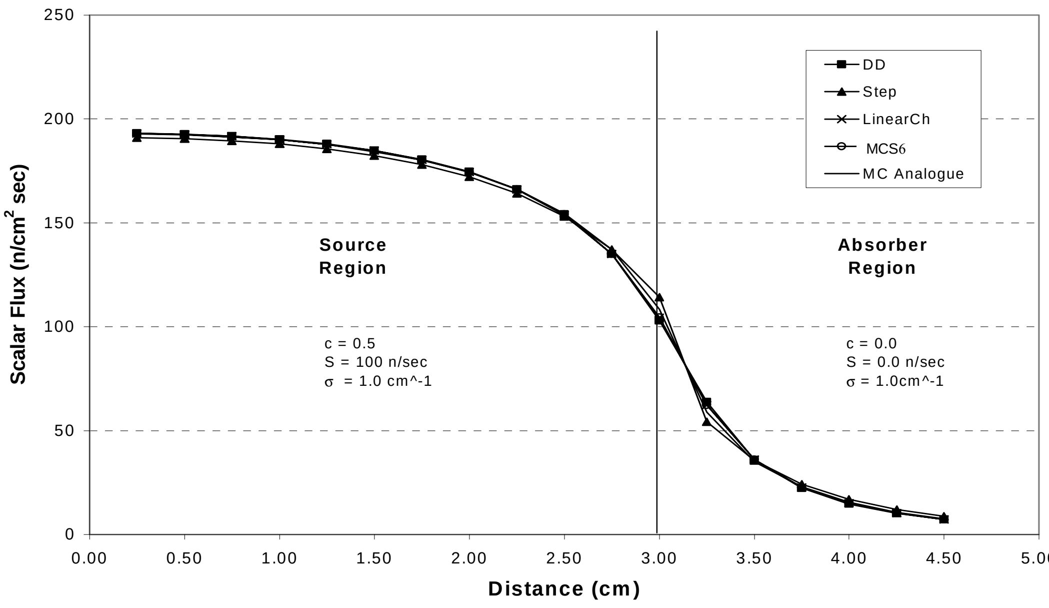

**Figure 3.5: "Slab" Two Region Test Problem**

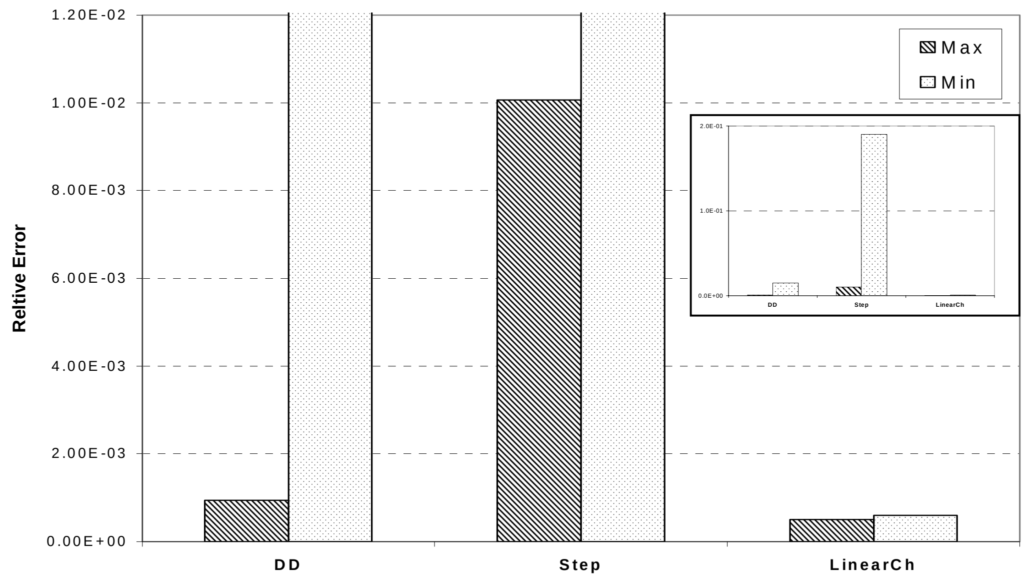

**Figure 3.6: "Slab" Max/Min Value Relative Error Comparison**

that follows does not account for any other form of variance reduction in the Monte Carlo $S_n$; therefore, the MC $S_6$ is taken as the best estimate to the real answer.

Figure 3.6 summarizes the relative error of each spatial quadrature compared against the MC $S_6$ flux profile. The chart shows the error present in the maximum and minimum flux value of each spatial quadrature compared to the maximum and minimum MC $S_6$ flux value. For clarity, the graph shows a scale enlargement with the magnitude of all errors shown in the smaller graph insert. As proven in the convergence tests, the LC method continues to predict flux values that are closer to the benchmark. The low relative error suggests that LC is more accurate in predicting flux values close to the benchmark. The Step method was the least accurate method in predicting flux values. This result is highlighted in figure 3.5 when looking at the flux transition between regions. The DD and LC methods show a relatively smooth transition between regions while the Step shows a more abrupt change in the flux. These results are caused by the assumptions in

[4] The error in the MC $S_6$ predicted flux was minimized to within a standard deviation of less than $10^{-5}$. The discrete ordinate flux profiles were converged to a relative convergence tolerance of $10^{-6}$.

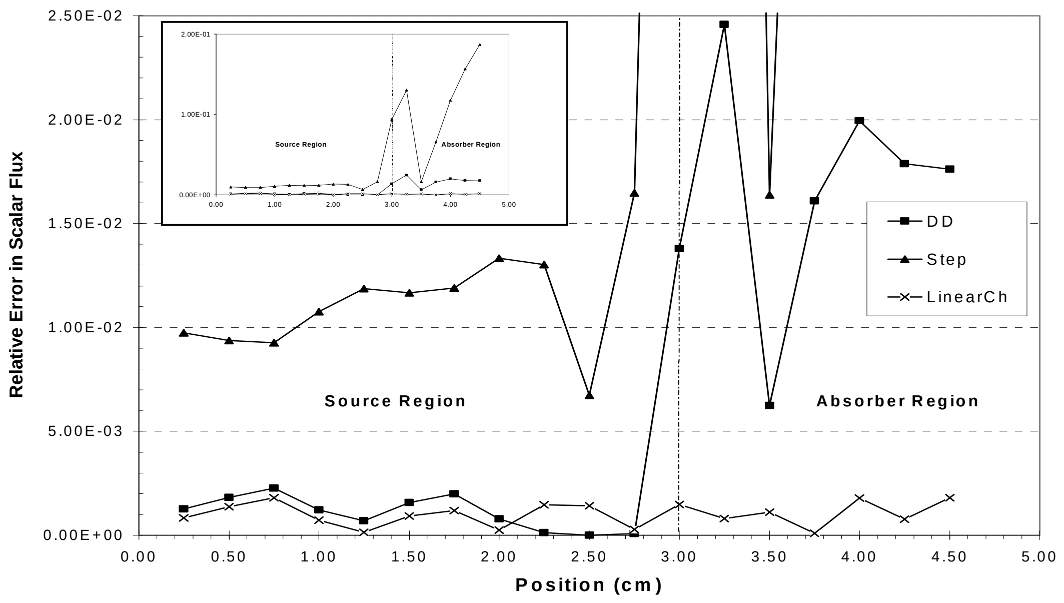

**Figure 3.7: Relative Error in Flux as a Function of Position**

the Step method of a constant source within the cell. Figure 3.6 shows the error relationship for the maximum and minimum flux values only. This graph does not show the relative change in error across the entire problem spatial domain.

In figure 3.7, the relative error as a function of spatial position for each spatial quadrature is plotted compared to the $MCS_6$ flux benchmark value. The scale is increased to accentuate the smaller less visible details shown by the full scale insert graph. The graph shows that both Step and DD method suffer an increase in error at the boundary transition. This error propagates to the absorber region. The maximum relative error occurs at the boundary interface of the two regions for DD while Step has a large increase in error at the region boundary with the maximum error located at the vacuum boundary. The LC method maintains a low relative error through the region boundary transition all the way to the vacuum boundary. The LC maximum error is located at the vacuum

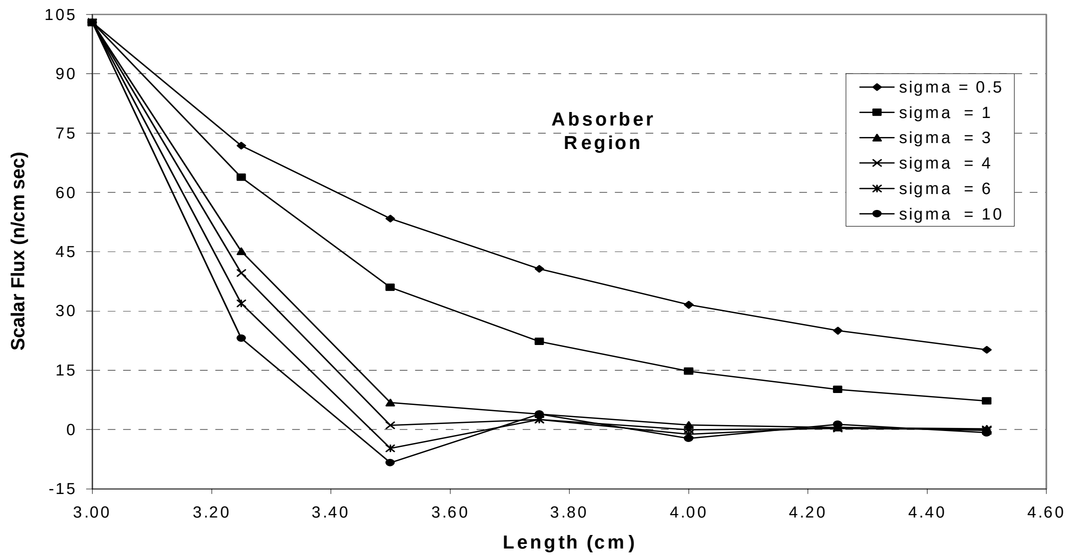

**Figure 3.8: Variation in Optical Thickness in Absorber Region Using DD**

boundary and is relatively small compared to DD and Step; therefore, the LC method predicts the flux shape better than DD or Step spatial quadratures over this mesh size[5].

The LC and DD methods can become unstable if the optical thickness of the cell becomes too large. The next test uses the two-region "slab" problem, but varies the total cross section in the absorber region while maintaining a constant mesh size. The goal of this test was to force negative fluxes in the DD and LC spatial quadratures and define a limiting optical thickness for maintaining realistic flux values.

Figure 3.8 shows the variation in the flux profile of the absorbing region for the DD method only. As the total cross section is increased, the flux in the absorbing region decreases. Eventually, the optical thickness in the absorber region cells becomes too thick causing the flux to become negative. This occurs at a cross section of approximately 3.5cm$^{-1}$. This test was repeated for both Step and LC. Step remained positive as the

[5] The error present in the DD and Step quadratures can be reduced in the absorber region if the mesh size is reduced. This significantly adds to computer memory requirements and may take some "tweaking" in the mesh grid size to reduce errors to within those produced by LC.

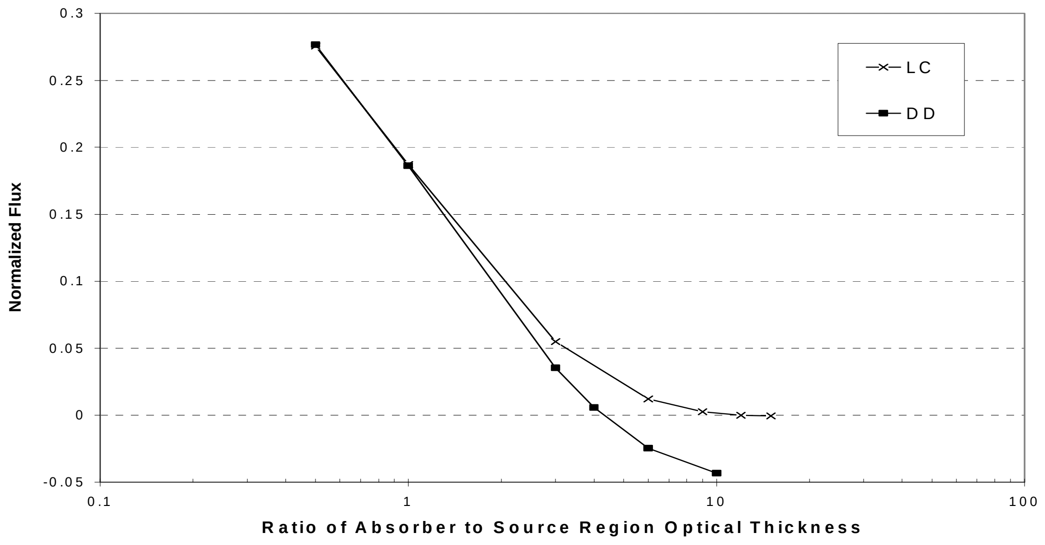

**Figure 3.9: Negative Flux Optical Thickness Limit**

literature states while LC generated negative fluxes after reaching a total cross section of about 11.5 $cm^{-1}$. The DD and LC negative flux behavior is summarized in figure 3.9.

The DD and LC methods were examined at the location of the first negative flux occurrence. For both methods, this occurred at a cell position of 3.5 cm. The flux in this cell was plotted in figure 3.9 as a function of the optical thickness ratio defined by dividing the absorber region optical thickness by the source region optical thickness. In essence, this value is a ratio of the total cross section of each region. The scalar flux in the cell was normalized based on the peak flux value over the entire problem domain. The graph shows that the flux tends to become negative for both DD and LC as the optical thickness ratio increases. As stated earlier, DD becomes negative much quicker than LC and begins oscillating between positive and negative values in the absorber region; therefore, the LC method performs better than DD over more optically thick cells. In fact, the LC method can calculate positive flux values over cells that are up to three times thicker than the optical thickness limit of the DD method.

Capability to model fewer as well as larger mesh sizes is possible by using the LC method. Accuracy can still be maintained without the costs associated with memory consumption caused by large numerical arrays. The reduced memory size comes at the price of increased computation time.  These costs are explored in the next section.

## Timing Tests

The cost of running a computer code falls into two basic categories, memory consumption, and speed. The amount of memory used by a code depends on the types of calculations performed and the storage requirements required for data arrays and their inherent data structure. The configuration of the data will effect how a computer accesses the information.  If data is not structured consistently and concisely, computer access time is wasted sifting through arrays for the right pieces of data required in a calculation; therefore, the speed at which a computer code calculates depends primarily on the data configuration and its storage location in memory.

For transport techniques, the discrete ordinates approach generally runs into lack of memory problems when spatial mesh grids become more refined and array sizes grow almost exponentially. On smaller spatial mesh arrays, the execution of discrete ordinates is relatively quick compared to the time required to implement a Monte Carlo technique. Monte Carlo suffers from a lower order of convergence in the error, approximately on the order of ½. This low order in error reduction forces large run times for achieving reasonable accuracy in the results. Therefore, the discrete ordinates approaches should be used when run times are more cost efficient than Monte Carlo techniques[6]. The following few paragraphs discuss the performance of the 3D-TRAN code in terms of total run time

[6] Timing results for Monte Carlo vs. discrete ordinates is given in table 3.2 later in this chapter.

and memory consumption. All results are based on running the code on a Hewlett-Packard Pentium 800 MHz desktop computer with 384M of RAM (100MHz access) and 37G of hard drive space. The processor used contained an AMD Athalon chip with a 200 MHz front side BUS. The FORTRAN code was run using COMPAQ visual Fortran 90/95 on a Microsoft Windows 98 operating environment.

The timing for the computer code can be broken down into several pieces to include initialization of input data, array allocation and assignment, calculations, and output generation. In most cases, the dominant piece that drives the overall run time is the calculation section of the code. This holds true for 3D-TRAN. Therefore, a fundamental unit of time that encapsulates the heart of the calculations was determined. This unit of measure is called time per phase space cell. The calculations in 3D-TRAN are heavily dependent on the type of spatial and angular quadrature used. The higher the order of angular quadrature or the size of the spatial grid, the larger the variable array sizes become and the longer it takes to run the code. The time per phase space cell reduces the time measure down to a fundamental calculation over one cell and one angle ordinate. This value is calculated by taking the total run time and dividing by the number of iterations, the number of energy groups, the number of angular ordinates, and the number of mesh cells. The timing results per phase space cell for each spatial quadrature are very consistent and relatively problem independent. The results for each spatial quadrature are given in table 3.2.

The timing results calculated in table 3.2 represent the average time per phase space cell for each method. These values were calculated using all the timing results recorded in each of the various test problems performed earlier in this chapter. The DD

|  | Time/Phase space cell (sec) |  |  |
|---|---|---|---|
|  | DD | Step | LC |
| **Average** | 2.7E-06 | 2.7E-06 | 1.6E-04 |
| **Variance** | 3.0E-14 | 3.6E-15 | 1.3E-08 |
| **Standard Deviation** | 1.7E-07 | 6.0E-08 | 1.1E-04 |
|  |  |  |  |
| **Range** | 4.4E-07 | 1.6E-07 | 2.9E-04 |
| **Coefficient of Variation** | 6.5% | 2.2% | 71.8% |

**Table 3.2: Time per Phase Space Cell for Each Spatial Quadrature**

and Step spatial quadratures possess fairly constant timing results. The standard deviation, data range, and coefficient of variation support this conclusion. These results are expected because both the Step and DD method use only a handful of equations to generate the average flux in the cell. These methods are not calculation intensive and thereby require only simple memory calls to array data.

The LC method is quite different with regard to time required. The calculation time is about 60 times higher than DD and Step. The extensive number of calculations performed in the LC method contributes greatly to the increase in time. Also, memory access speed slows down because the timing data showed an increase in the time per phase space cell as the angular quadrature and mesh array size increased. With larger arrays, it takes the computer code longer to search through all the arrays and find the required data while performing numerous calculations for one phase space cell. This result is reflected in the standard deviation of the average time per phase space cell as well as the coefficient of variation indicating a large degree of scatter in the data about the mean. Therefore, the time per phase space cell calculation for LC can be used to generate an approximate guess at the run time for a given problem assuming the number of iterations to convergence is known.

The data listed in table 3.2 is very useful in determining the time it takes to perform a calculation. The first piece of information needed is the time it takes to perform one iteration through the code. This is accomplished by multiplying the average time per phase space cell for the specified spatial quadrature by the number of spatial mesh cells and the number of angular ordinates. This number is then multiplied by the number of iterations required for convergence. Before calculating the final run time estimate, the memory limit of the computer must be examined.

For the computer architecture used in this research, the discrete ordinates methods had a limit to how many phase space cell calculations were allowable within computer memory. For DD and Step, about $8x10^6$ calculations per iteration were possible while LC was only capable of $3x10^6$ calculations. At these values, the computer starts using the hard drive as virtual memory, greatly slowing down calculation time as data is transferred between memory and disk. A problem run time can jump from taking only few minutes to hours or even days with virtual hard drive memory. Eventually the code fails because there is not enough computer memory. The calculation memory limit numbers help the user to define problem parameters to keep computer run times down and memory requirements within acceptable RAM limits. Although the memory limits are much lower for LC compared to DD and Step, the LC method requires less total phase space cell calculations (but performs more calculations per phase space cell) to achieve the same accuracy of the DD and step methods.

After the time per iteration is determined and the memory limit evaluated, the number of iterations is required to generate a final run time for the code. The number of iterations is very difficult to predict since it is highly problem dependent. Therefore, if the

LC method is the desired spatial quadrature, the user should run the Step or DD method to see approximately how many iterations it takes for either method to converge. This number is generally consistent with the number of iterations required by LC to converge (at least this holds for the range of problems examined thus far in this research). The inherent positivity of Step and its speed make Step a good first guess to discovering the number of iterations required for a problem. The DD method is not always a good predictor since certain heterogeneous problems may take longer for DD to iterate to convergence, especially if negative fluxes are present. This uncertainty in convergence is seen in table 3.3.

| **Method** | **No. iterations** | **Total Time (sec)** |
|---|---|---|
| **DD** | 133.00 | 2.37 |
| **Step** | 41.00 | 0.77 |
| **LC** | 41.00 | 31.75 |
| **MCS$_6$** | - | 429.57 |
| **MC Analogue** | - | 413.26 |

**Table 3.3: Timing results from Two-Region "Slab" Problem**

The table summarizes the timing results obtained from the two-region three-dimensional "slab" problem shown in figure 3.5. The Step and LC methods converged in 41 iterations while the DD method converged in 133 iterations. The number of iterations required for DD was greater because the method suffered from instability in the more optically thick cells in the absorbing region. Since this region was a pure absorber with no source, the DD method initially calculated negative fluxes in the earlier iterations. This result is due to the initial guess of zero scalar flux for the first iteration. DD took extra iterations to stabilize and build the flux iteratively to a positive value. The Step and

LC method are much more stable over optically thick cells and therefore avoided the instability of the DD method. As the optical thickness increases for the test problem, the DD method remains unstable in iterations required for convergence as the total number of iterations changes from a minimum of 130 to a maximum of 189. The LC and Step methods remain relatively stable at 41 total iterations required for convergence for all optical thicknesses tested. Therefore, Step serves as a good initial guess for the number of iterations to convergence over larger optically thick cells allowing for a good timing estimate to be calculated for the more accurate LC method.

For comparison purposes, the Monte Carlo techniques are included in table 3.3 for a direct comparison of time with discrete ordinates.  These results show that Monte Carlo takes approximately 13 times longer to reach an answer (within a standard deviation of no greater than $10^{-4}$) than the LC method for the two region "slab" problem. A substantial increase in run time is required for Monte Carlo to reduce the overall statistical error in the scalar flux. Therefore, Monte Carlo is an expensive method for determining flux shapes over various problems while discrete ordinates is relatively inexpensive if problem sizes are small.  The deciding factor of determining which method, Monte Carlo or discrete ordinates, to use depends on accuracy required, computer memory available, and speed of calculations. These determinations are purely problem specific, but both methods can provide good direct comparisons to help verify the validity of data.

## Initial Guess and Its Effects on Code Run Time

A final issue that deals with calculation time is the initial flux guess. The discrete ordinates approach requires an initial guess to the scalar flux so the right hand side of the transport equation can be calculated. For simple transport problems, zero is generally used because the problems run quickly. For more complex problems, a zero flux guess increases the run time of the code because it takes longer to iterate to the final answer from zero. This also may introduce negative fluxes early on in the iterative process that will greatly affect the total number of iterations required to convergence. This result was already seen with the DD method over the three-dimensional, two-region "slab" problem. Therefore, run times can be reduced if a good initial guess is used for the scalar flux.

After examining the run times for all three spatial quadratures, the LC method takes the longest time with an average run time of 60 times higher than Step and DD. Therefore, a good first guess generator for LC is to use DD or Step to generate an initial guess to the flux profile. The scalar flux for the initial guess generator is converged to a larger tolerance than the final problem eliminating wasteful iterations. In general, the Step and DD methods may only contain the first one or two digits of accuracy compared to LC with accuracy increasing or decreasing based on mesh cell sizes. For more refined meshes where the optical thickness of the cell is much larger than the mesh cell size, the DD method predicts a good initial guess because the smaller mesh allows for stable convergence to answers closer to the LC method. As the cell size increases closer toward the optical thickness limit, the DD method becomes progressively worse. The Step method then becomes the viable option for producing an initial guess since it performs better than DD over large optically thick cells.

The results for these behaviors on convergence are very problem dependent. If the answer is not initially well defined, the Step method will always provide a good guess to the behavior of the answer since it is always positive. The DD method may also be used as long as the optical thickness of the spatial cells is examined against the limit seen in figure 3.9. The average reduction in computation time seems to be only about 20%. This number is based on an initial guess convergence tolerance of three digits using either Step or DD as the guess generator. For more refined mesh problems such as the two-region slab problem of figure 3.5, the reduction in time can be as much as 60%. These results are summarized in table 3.4.

The first set of data listed in table 3.4 shows the initial run times for all three method using zero as the initial guess. The low run times for DD and Step suggest that an initial guess using either method will be generated quickly without any adverse effects on the LC run time. The LC method was run again with either DD or Step used as the initial guess generator. The tolerance on the initial guess will generate three digits to start the LC method iterative process. The Step method took 22 iterations to generate its initial guess, but reduced the LC run time by about 23 %. The DD method took 48 iterations to generate its initial guess, but reduced overall run time by about 58%. This is a substantial reduction in overall run time. The final times for LC using the initial generator include the time required to generate the initial guess. The DD method produced a better estimate of the initial guess than the Step. In fact, the first three digits of the DD guess for every

| Method | Iteration | Tolerance | Total Run Time (Sec) |
|---|---|---|---|
| **LC** | 41 | 1.00E-06 | 31.75 |
| **DD** | 133 | 1.00E-06 | 2.37 |
| **Step** | 41 | 1.00E-06 | 0.77 |
| **Guess Generator and Effects on LC Method** | | | |
| **Step** | 22 | 1.00E-03 | 24.33 |
| **DD** | 48 | 1.00E-03 | 13.4 |

**Table 3.4: Two Region "Slab" Timing Data and Effects of initial Guess on LC**

flux value matched those of the LC method. Only the first one or two digits of the Step matched the LC method. Therefore, a lower number of iterations can be achieved if more accurate digits are generated in the initial guess. For larger optically thick cell problems, the roles of DD and Step reverse with Step producing the better initial guess.

Although numerous time reduction test problems were run duplicating previous tests performed in this chapter, the results indicate only the general behavior of when DD or Step are more applicable as an initial guess generator. The reduction in time for the LC method achieved by using the initial guess varied from as little as 10% to as much as 60%. These results show that the LC method run time can be reduced if a good initial guess is generated. More research should be performed on either generating a better initial guess that works well over most problems or finding a suite of methods that work well for particular problems. The overall benefit will be achievable reductions in the LC run time that will play an important role in transport problems that run for many minutes or even hours.  For now, the DD and Step methods are used in the 3D-TRAN code for initial guesses to the LC method with an achievable decrease in the overall run time by at least 20%. This number assumes the initial guess generator will have at least the first one or two accurate digits in the guess that match those in the LC answer.

# Chapter IV: Reactor Physics and Eigenvalue Calculations

The following chapter summarizes the performance of the 3D-TRAN code against the analytical diffusion equations for a homogenous fission reactor in the shapes of a parallelepiped and a finite right circular cylinder. The normalized flux profiles and eigenvalue calculations are compared against diffusion to check for correct modeling of the transport isotropic fission source term. After diffusion comparisons are presented, the 3D-TRAN code is run for a few reactor based problems to demonstrate its overall versatility as a teaching tool for reactor physics.

## Diffusion vs. Neutron Transport Revisited

The following sections of this chapter present a direct comparison of analytical diffusion theory to numerical neutron transport theory. It is important at this time to re-establish the importance of this comparison in order to better understand the purpose of this research.

Diffusion theory is generally chosen as the predominate theory for analyzing nuclear reactor type problems. Analytical solutions are available for simplified reactors and conversion of diffusion theory to numerical applications is straightforward. The numerical applications are generally fast in smaller dimension problems, but suffer from the same memory consumption issues of neutron transport when reactor size dimensions are utilized. Large spatial mesh arrays established under finite difference techniques are subject to relatively small spatial steps in order to produce stable results.

Heterogeneity adds another complication. Highly absorbing materials (such as control rods, burnable poisons, etc.) are very difficult to model accurately with diffusion theory. The angular dependence of the flux near such materials often makes diffusion theory inadequate within a few mean free paths away from the absorber boundary because of its assumed isotropic form of the flux.

Neutron transport theory does not suffer from these effects.  Heterogeneity is easily modeled under numerical transport and angular dependence in the flux can be solved for explicitly. This fact makes neutron transport the optimal choice for studying heterogeneous reactor problems where control rod behavior (or other neutron absorbers) in fuel assemblies is of interest. Neutron transport does suffer from computer memory consumption, and numerically solving for eigenvalue type problems can also take a long time compared to diffusion theory. The end result is that neutron transport can produce a more realistic answer to a given problem because the transport behavior is modeled more directly than with diffusion theory where assumptions have removed angular dependence of the flux and assumed diffusive particle type behaviors.  Although, diffusion theory is very applicable in homogenous and relatively mild heterogeneous reactors, the neutron transport theory will yield the same results over the diffusion problem domains as well as produce more accurate answers than diffusion theory in highly heterogeneous problems. This final fact is why a neutron transport model of a reactor core is being explored in this research.

## Homogenous Reactor Testing: Part 1, the Parallelepiped

The diffusion equation can be used to generate analytical solutions to simplified geometry reactors. The general form of the one speed diffusion is:

$$-D\nabla^2\phi(\bar{r}) + \Sigma_a\phi(\bar{r}) = S(\bar{r}) \tag{4.1}$$

where $D$ is the diffusion coefficient and $\Sigma_a$ is the absorption cross section of the material. For a homogenous medium, the source term $S$ is set to zero. The result is a simple differential equation that can be solved by determining the form of the del operator. For three-dimensional space in Cartesian coordinates, the solution to the homogenous, one speed diffusion equation becomes:

$$A\cos\left(\frac{\pi x}{a}\right)\cos\left(\frac{\pi y}{b}\right)\cos\left(\frac{\pi z}{c}\right). \tag{4.2}$$

The dimensions of the parallelepiped consist of sides of length a, b, and c and $A$ represents a normalization constant based on the problem of interest. Equation (4.2) can be directly used to predict the normalized flux shape inside a homogeneous parallelepiped reactor. Only one modification is required to the diffusion equation, the use of extrapolated boundaries. These extrapolated boundaries correct for the physical behavior of the neutron flux not going to zero at the actual problem boundary[1]. The extrapolation relation is based on a correction predicted by the neutron transport equation and has the form of:

$$d = 0.71\lambda_{tr} = 0.71(3D) = 2.13D \tag{4.3}$$

where $d$ is called the extrapolation distance and $\lambda_{tr}$ is the transport mean free path (Lamarsh, 1983). This distance is added to the lengths of the sides of the reactor to

[1] The extrapolation distance corrects the diffusion equation near regions such as vacuum boundaries where Fick's Law, a key assumption used to derive diffusion theory for neutron transport theory, is no longer valid.

correct the neutron flux near vacuum boundaries. With these tools in hand, a comparison of the predicted flux shape can be made between diffusion and neutron transport.

The first problem of interest was designed based on the work performed by Harman (2001). His work used a set of two energy group cross sections defined in table 7.2 of Duderstadt and Hamilton (1976). The cross sections are summarized in table 4.1. The neutron transport analysis requires the total cross section and the "within-group" scattering term to correctly model the isotropic fission source.  The total cross section was calculated based on the diffusion coefficient and assumes that the transport cross-section is approximately the same as the total cross section. This assumption is valid only for an isotropic medium (Duderstadt and Hamilton, 1976) because the average scattering cosine, $\mu_0$, is relatively small (meaning scatter is likely to occur in any direction) and the diffusion coefficient equation becomes:

$$D(\vec{r},E)=\frac{1}{3}\left[\Sigma_t(\vec{r},E)-\bar{\mu}_0\Sigma_s(\vec{r},E)\right]^{-1}\approx\frac{1}{3}\left[\Sigma_t(\vec{r},E)\right]^{-1}. \quad (4.4)$$

The "within group" cross sections can then be calculated by subtracting the absorption cross-section and the group $g$ to $g'$ scatter cross section from the total cross section. As with Harman's work, the fission distribution assumes all fission neutrons are born in the upper energy group.

| Group Constants | 1 of 2 | 2 of 2 |
|---|---|---|
| $\nu\Sigma_f\,(cm^{-1})$ | 0.008476 | 0.185140 |
| $\Sigma_f(cm^{-1})$ | 0.003320 | 0.075370 |
| $\Sigma_a(cm^{-1})$ | 0.012070 | 0.121000 |
| D(cm) | 1.262700 | 0.354300 |
| $\Sigma_R(cm^{-1})$ | 0.026190 | 0.121000 |
| $\Sigma_{s12}(cm^{-1})$ | 0.014120 | 0.000000 |
| $\Sigma_t(cm^{-1})$ | 0.263984 | 0.940822 |
| $\Sigma_{sgg}(cm^{-1})$ | 0.237794 | 0.819822 |
| $\chi(E)$ | 1.0 | 0.0 |

**Table 4.1: Two Group Cross-Sections for typical PWR reactor**

The parallelepiped reactor modeled using the 3D-TRAN code used the two group cross sections of table 4.1, the $S_6$ angular quadrature, and consisted of a cube surrounded by vacuum with sides of length 300 cm. The length was designed to yield an eigenvalue better than one. The mesh spacing inside the reactor was refined using the techniques of Chapter 3 to determine the convergence of the average flux value across the reactor. This test was performed to verify that the scalar average flux did indeed appear to converge to a specific value and to verify the stability of the eigenvalue iteration procedure in accurately reproducing consistent results over more refined meshes. The results are summarized in figure 4.1.

The graph shows that as the mesh is refined, the average flux does appear to converge to the analytical diffusion average flux value. This figure shows the same consistent behavior of figure 3.1[2]. The analytical diffusion actually lies slightly beneath the converged transport solutions. In reality, the diffusion average flux value has an error associated with it. The extrapolation distance used to model the boundaries of the diffusion based reactor is an approximated transport correction factor. The approximation

[2] A relative tolerance of $10^{-6}$ was used for the scalar flux and $10^{-6}$ was used for the eigenvalue.

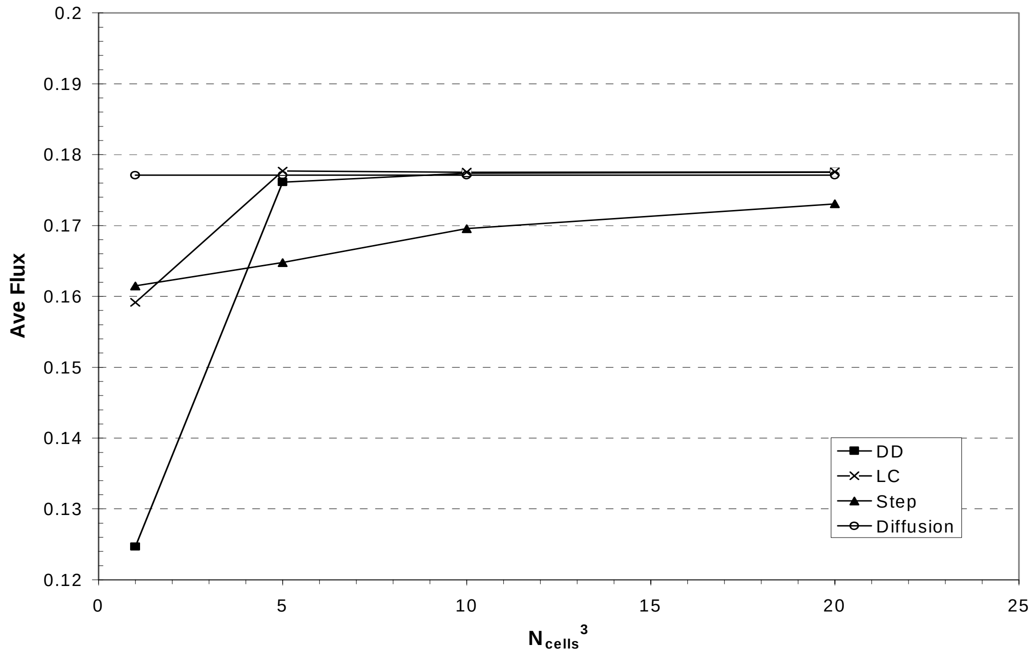

**Figure 4.1: Convergence on Average Scalar Flux Fission Test**

contains some magnitude of error. Therefore, the analytically solved average flux value may lie slightly above or slightly below its current position. The magnitude of the error is relatively small and not considered important in this test problem. The important point of figure 4.1 is to show that the transport solution and the analytical diffusion solution are in relative agreement with each other; therefore, the overall flux shape predicted by 3D-TRAN for our remaining homogenous analysis will be compared against the analytical diffusion equations[3].

[3] Note that the magnitude of the flux units of figure 4.1 is purely dependent on the initial guess used in the 3D-TRAN code. Unlike the results of chapter 3 where a source was specified with an actual value, the fission source on the right hand side of the transport equation will drive the magnitude of the flux generated in the solution. In other words, there can be multiple solutions to the flux profile; however, the overall shape of the profile will remain the same. This effect is seen in diffusion theory when solving the Helmholtz equation with zero sources or sources containing flux (Duderstadt and Hamilton, 1976) where the magnitude of the flux (for example, $10^{10}$) is easily divided out of the equation.

Figure 4.2 shows the two-dimensional parallelepiped, xy flux profile at a z value of 150cm predicted by the Linear Characteristic (LC) method. The plot has been normalized to the maximum flux value. This plot shows that the LC model appears to represent the correct flux shape predicted by equation (4.2). The Diamond Difference (DD) and Step models predict the same behavior and are omitted for brevity. A more direct comparison can be made by looking at a one-dimensional slice through the core at the centerline with z = 150cm and x = 150cm moving along the y direction. These results are seen in figure 4.3.

The graph shows that the numerical transport solution and the analytical diffusion solution are relatively close. For graphing purposes, a square cell size of 15 cm (a side) was chosen to examine the relative error using analytical diffusion as the benchmark.

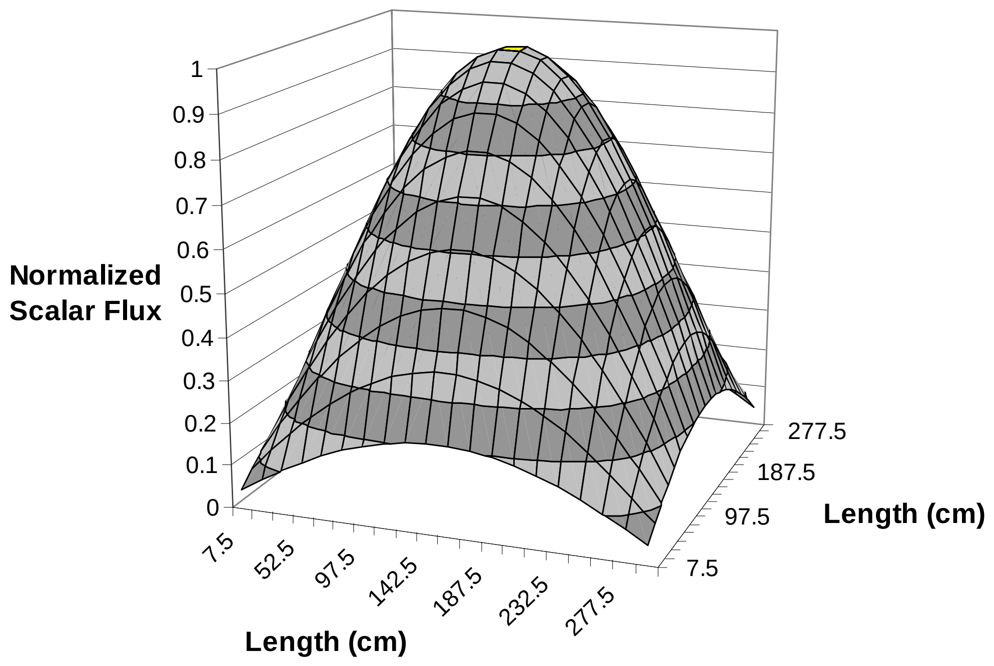

**Figure 4.2: Normalized Scalar Flux Profile for PP Fission Reactor Using LC**

This represents a refinement of 20 cells on each side of the reactor. The DD and LC methods tended to predict the same flux profile while Step over predicted the profile near the edge of the reactor. The refinement of the spatial grid in the transport solution did force the transport solution to more closely approximate the analytical diffusion solution. There was a consistent tendency of the transport and the analytical diffusion solutions to differ near the boundaries of the reactor core. This result is caused by the difference in how each method handles the boundary conditions. The diffusion solution assumes a fixed boundary that is adjusted using the transport correction approximation of equation (4.3). The transport solution assumes that the incoming flux along a vacuum boundary is fixed to zero, but it does not fix the outgoing flux. The flux will adjust along the edge according to the spatial and angular mesh refinement as the outgoing flux is better approximated numerically. The relative error graph comparison of the flux profile shown

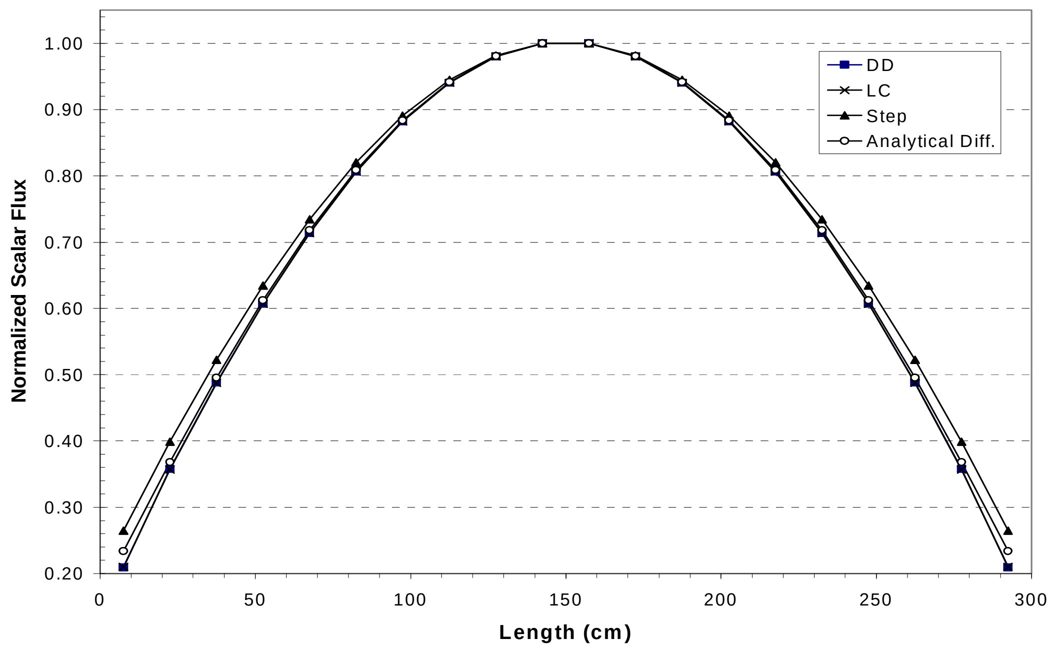

**Figure 4.3: One Dimensional Scalar Flux, Transport vs. Diffusion**

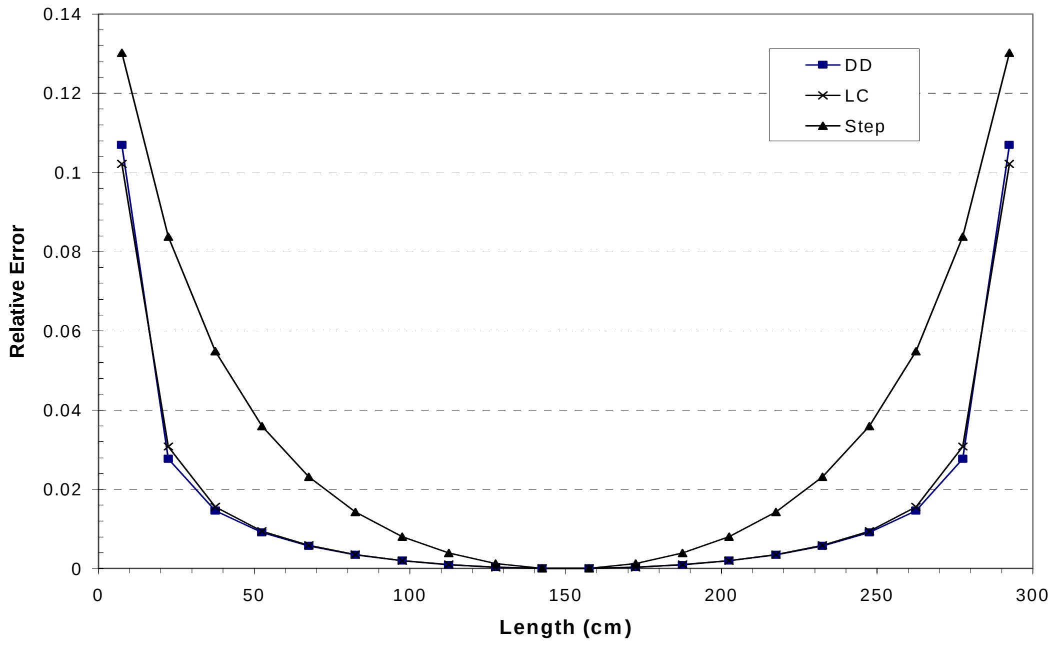

**Figure 4.4: Relative Error Benchmarking Transport to Analytical Diffusion**

in figure 4.3 is given in figure 4.4.

Figure 4.4 shows that both DD and LC are within 3% of the predicted flux of analytical diffusion except near the boundary where the error jumps to a little more than 10%. The Step method has over predicted the analytical diffusion flux profile as seen in figure 4.3 with a larger error of about 13% maximum error near the edge of the core. These errors will decrease as the mesh is refined; however, for the homogenous isotropic fission reactor, the DD and LC methods always tend to better predict the flux profile while Step tends to over estimate the shape. Of course these results assume that the optical thickness is relatively small and well within the bounds of stability defined by figure 3.9 (Optical thickness comparison between LC and DD).

For this test problem, the optical thickness fell at 24.4 for group two and 6.2 for group one. This result suggests that the large optical thickness would introduce instabilities into both DD and LC for group two. Fortunately, the homogenous medium of

| Method | K | Total Iterations | Time (sec) | Convergence Acceleration | Flux Tolerance | K Tolerance | Rel. % Error |
|---|---|---|---|---|---|---|---|
| **DD** | 1.13017 | 1893 | 569.10 | No | 1.00E-04 | 1.00E-04 | 2 |
| **DD** | 1.13017 | 813 | 247.33 | Yes | 1.00E-04 | 1.00E-04 | 2 |
| **Step** | 1.04095 | 209 | 62.06 | Yes | 1.00E-04 | 1.00E-04 | 25 |
| **LC** | 1.12924 | 565 | 7789.70 | Yes/initial guess | 1.00E-04 | 1.00E-04 | 2 |

**Table 4.2: Parallelepiped Reactor Eigenvalue Test Results**

the parallelepiped reactor allowed for a stable solution to be achieved by the transport method since each cell essentially had no change in the optical thickness along each direction of the reactor (an optical thickness ratio of 1.0, which suggests from figure 3.9 stable results). A problem will and does arise if the reactor becomes heterogeneous. The calculation of the flux across the boundary will become unstable when the optical thickness ratio reaches the threshold suggested in figure 3.9. Further discussion on this subject is presented later in this chapter.

The final analysis of the parallelepiped involves a discussion of convergence on the criticality eigenvalue K. The test problem run used the same reactor configuration and cross sections as previously mentioned, but used a fixed spatial mesh consisting of 30cm cubic cells (1000 total mesh cells). The maximum relative error of the flux profile for each method compared to analytical diffusion is listed in table 4.2. The low relative error shown for DD and LC actually represents the maximum error along the interior mesh points of the reactor (neglecting the edge values). The outer most mesh point had a relative error of about 8% maximum on the edge. The cell meshing for this problem was picked based on these low relative errors for DD and LC. Any further refinement in the mesh will increase overall computational time.

The eigenvalue results of table 4.2 show that the reactor is supercritical. This is good because the criticality can always be lowered by adding neutron absorbers to the core to help suppress the flux profile. The larger error in the Step method accounts for the lower K value calculated. The spatial quadratures were run under a less refined tolerance ($10^{-4}$) to speed overall computation time. For example, a more refined tolerance on the eigenvalue and flux of $10^{-6}$ extended the total number of iterations to convergence out to 3809 iterations and produced an eigenvalue of 1.12117. In reactor calculations, the two digits following the decimal point in the K value are generally the most important values because they determine the reactors relative sub or super criticality at any given time. Therefore, to help speed calculations, a less refined tolerance was used with hopes of capturing three good digits in the K value.

After looking at table 4.2, one will instantly see why run time is so important. The test problem was initially run with an initial guess of one for the flux and a small tolerance. These values are listed as the first entry in table 4.2. The code took 1893 iterations to reach convergence to the specified relative tolerance. The number of iterations to convergence for this problem suggests that the LC method will take on the order of hours to compute an answer (approximately 8 seconds an iteration for a total run time of about 4.25 hours using table 3.1 information). This simple test shows that the 3D-TRAN code requires some form of convergence acceleration other than the initial guess generator discussed in Chapter 3. A simple linear extrapolation/shooting method was implemented to help speed convergence. The remaining results in table 4.2 use the convergence acceleration routine developed for the 3D-TRAN code. Further discussion of the convergence acceleration method will be provided later in this chapter. For now, it is only necessary to show the requirement for some form of convergence acceleration routine to make the code more practical for use as a teaching tool.

## Homogenous Reactor Testing: Part 2, the Finite Right Circular Cylinder

To continue our discussion of transport vs. diffusion theory, we examine the finite right circular cylinder reactor analyzed by Harman (2001). This test problem will serve three purposes. First, the 3D-TRAN cylinder mesh initialization routine can be tested for accuracy in modeling a cylinder in a Cartesian mesh. Secondly, the results from Harman provide a direct comparison for transport and diffusion theory predicted eigenvalues. Finally, the test problems will serve as a direct test to see if the isotropic fission source of the 3D-TRAN code is modeling the simple isotropic source developed by Harman (2001) in his diffusion code.

The solution to the analytical diffusion equation (4.1) for a homogenous finite right circular cylinder is:

$$AJ_0\left(\frac{2.405r}{R}\right)\cos\left(\frac{\pi z}{H}\right) \tag{4.5}$$

where *A* is the normalization constant, $J_0$ is the Bessel function of order zero, *R* is the outer reactor radius and *H* is the core total height. This equation uses the extrapolation distance (as calculated in equation (4.3)) to correct the core radius and the core height to more closely approximate the transport vacuum boundary conditions.

The transport code was modified to include a cylindrical initialization routine. Since the code was written in Cartesian coordinates, the cylinder model routine will only approximate the curved edge of the cylinder. This is accomplished by using the equation of a circle with origin fixed at some location in the mesh to determine if the center of a computation mesh cell lies within the radius of the circle. If it does, the mesh cell is initialized as a reactor core cell, and appropriate cross sections and material properties are assigned. Otherwise, the mesh is left blank and initialized to zero. An example of a Cartesian mesh approximation to a circular curve is provided in figure B.1, appendix B. Also, a discussion of how to properly handle the boundaries in the transport code is presented. The end result is that as the Cartesian mesh is refined, the curved edge becomes more defined in the Cartesian mesh and will eventually approach the circular curve in the limit as the cell spacing becomes small. Of course, the onset of rounding errors in small numbers will introduce errors into the transport solution long before the limit is reached. Therefore, the circular mesh generation routine is a zeroth order approximation to a circle and does not conserve the mass and volume of a true right circular cylinder. This fact will introduce larger errors into the transport solution along the curved edge if the mesh cells are too large. These errors will be examined in the following test problem.

The cylindrical reactor analyzed by Harman used the two group cross sections of table 4.1 with reactor dimensions referenced from Duderstadt and Hamilton (1976). The dimensions of the reactor are a height of 360cm and an outer radius of 120 cm surrounded on all sides by vacuum. As Harman refined his spatial mesh, his numerical diffusion solution approached the analytical diffusion solution, equation (4.5) (Harman, 2001). These results were expected; therefore, the analytical solution will be used for a comparison of transport and diffusion. This reactor configuration was modeled in the 3D-TRAN code with the added criteria of using the $S_6$ angular quadrature set to approximate the angular integration. To conserve computer memory, the cylindrical reactor was modeled using quarter core geometry with full symmetry boundaries at $X_{min}$, $Y_{min}$ and $Z_{min}$. This allowed for refinement of the mesh to better approximate the cylindrical curve without exceeding computer memory or sacrificing speed.

Figure 4.5 shows the two-dimensional, right cylinder, xy flux profile at a reactor core z value of 180cm predicted by the Linear Characteristic (LC) method. The plot has been normalized to the maximum flux value. The LC model appears to represent the correct flux shape predicted by equation (4.5). The Diamond Difference (DD) and Step models predict the same behavior and are omitted for brevity. The jagged edge around the bottom of the flux profile indicates the jagged approximation to the curved cylindrical surface[4]. This profile contains mesh cells that are 12cm x 12cm in x and y and 9 cm in z

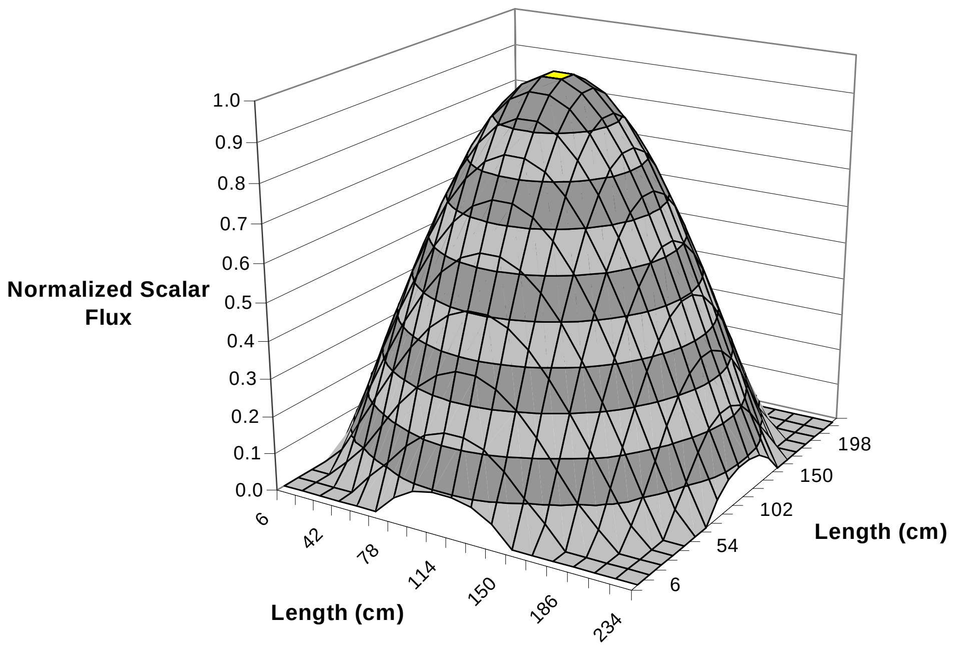

**Figure 4.5: Normalized Scalar Flux Profile for Cylinder Reactor using LC**

[4] The graphing program used to plot the two dimensional flux profile artificially set the flux along the edges to zero because the 3D-TRAN code sets all values outside the approximated cylinder to zero.

(or meshing that is 10 x 10 x 20). As with the parallelepiped, a more direct comparison can be made by looking at a one-dimensional slice through the core starting at the axial centerline at a z location of 180 cm. These results are summarized in figure 4.6.

The graph shows that the numerical transport solution and the analytical diffusion solution are relatively close in shape. The cell size for this shape is the same as for figure 4.5 above. As seen with the parallelepiped, the Step method over predicts the flux profile near the edge of the core. The DD and LC methods have predicted virtually the same flux profile with both coming very close to reproducing the analytical diffusion solution. A less refined mesh caused all the transport methods to over predict the diffusion solution along the edge of the reactor. At the mesh refinement of figure 4.6, the Step method still appears to require an even more refined mesh to better approximate the cylindrical diffusion solution. The radial relative error for figure 4.6 is shown in figure 4.7.

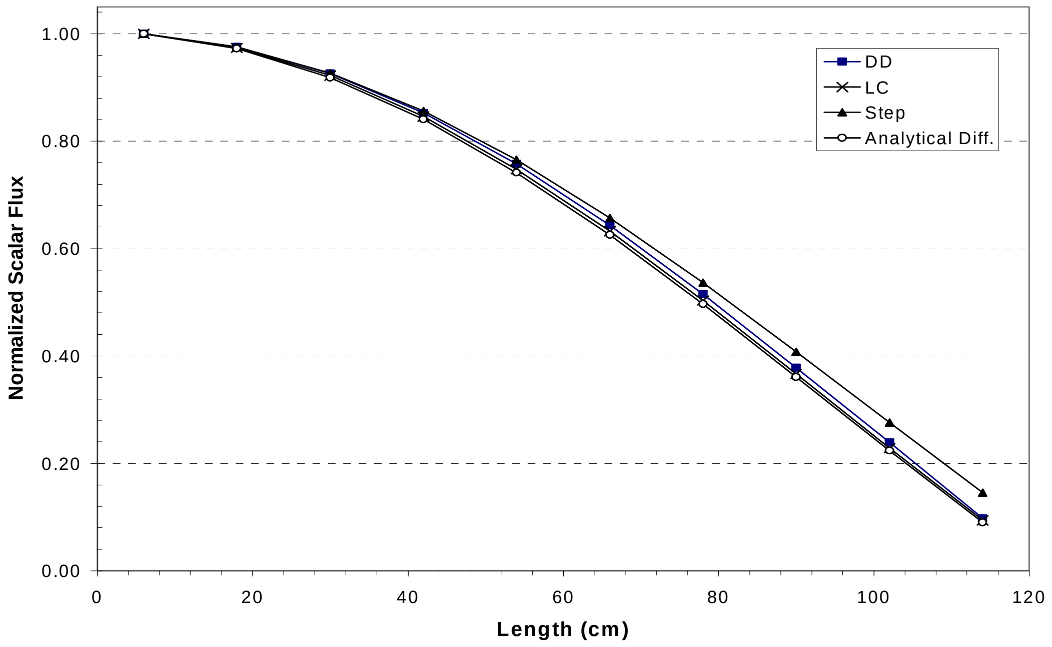

**Figure 4.6: Radial Flux Profile Comparison for Cylindrical Reactor**

This graph (figure 4.7) shows that the DD and LC solution are within 10% of the diffusion solution. As expected, the LC method is within 5% of predicting the analytical solution and out performs both DD and Step. The relative error for all spatial quadratures increases over the last two mesh points starting around 102 cm radial.  This increase in error is caused by the approximation of the curved cylindrical edge in the Cartesian mesh. During testing, as the mesh was refined, the error along the boundary decreased as expected. The other possible error is the calculation of the extrapolation distance used to correct the boundary of the diffusion solution. The extrapolation is a transport approximation and is therefore subject to errors. The important feature of figure 4.7 is to note that the transport solution does approach the corrected analytical diffusion solution as the spatial mesh is refined and that the greatest error will lie along the approximated curved edge of the reactor. These results show that the cylindrical mesh generating routine of 3D-TRAN does work and will predict the correct flux profile for a cylinder.

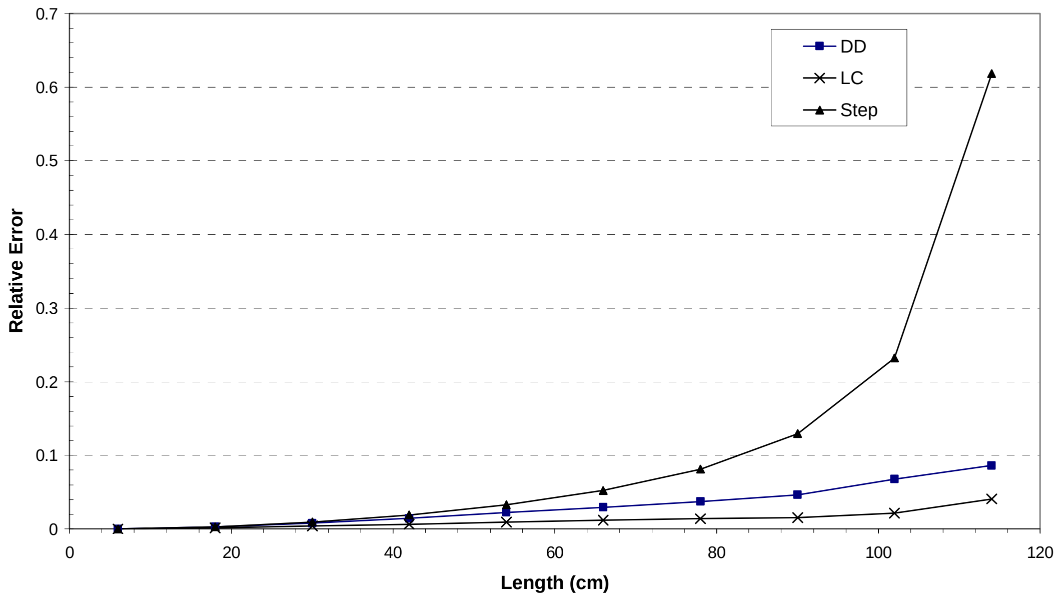

**Figure 4.7: Radial Relative Error Comparison for Cylindrical Reactor**

The only problem is that a more refined mesh may be required to predict the flux more accurately near the curved reactor boundary.

The solution to the axial flux profile is also very good. Figure 4.8 shows the axial flux profile at the same refined mesh size as figures 4.5 – 4.7. As in figure 4.6, the DD and LC methods predict the flux profile very well while the Step method over predicts the flux along the reactor edge. As stated earlier, the refinement of the mesh will bring the Step method flux profile closer to the flux predicted by the analytical diffusion solution. Figure 4.9 shows the axial variation in the relative error using the analytical diffusion as the benchmark.

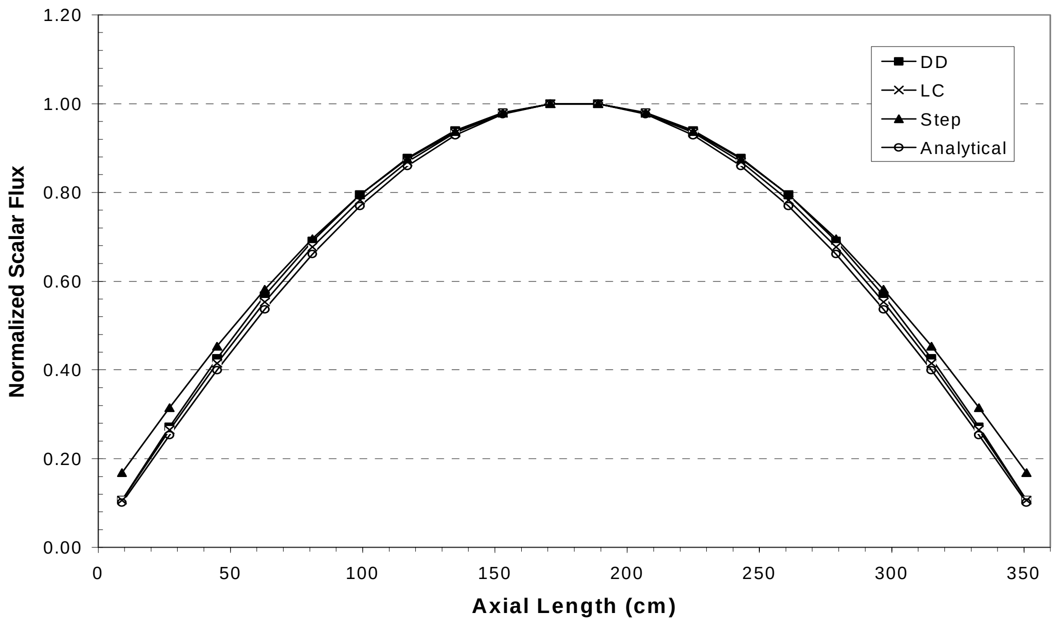

**Figure 4.8: Axial Flux Profile Comparison for Cylindrical Reactor**

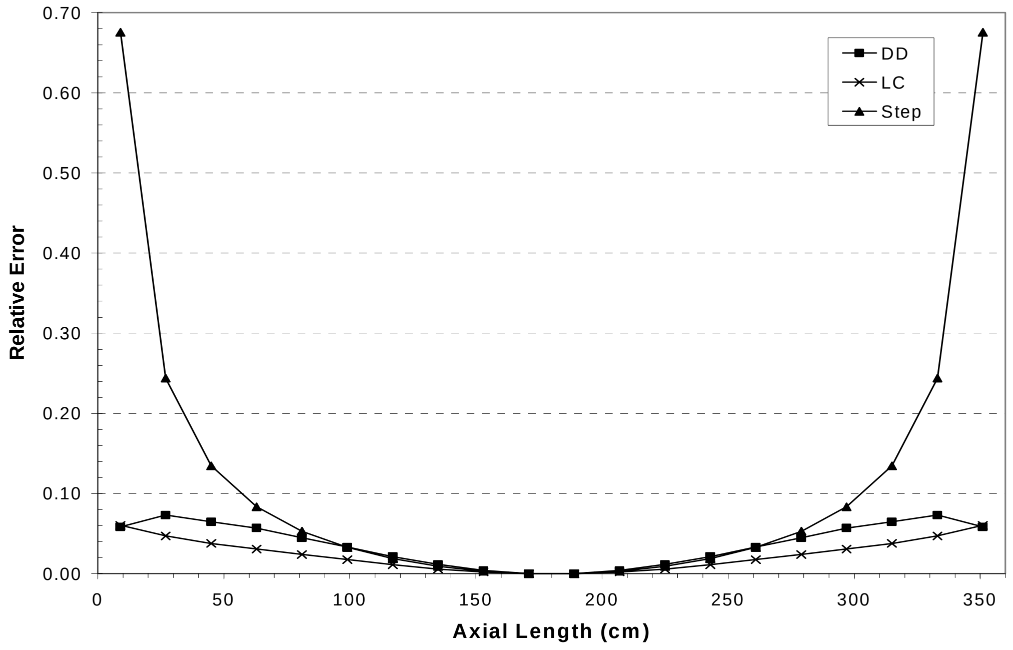

**Figure 4.9: Axial Relative Error Comparison for Cylindrical Reactor**

The graph shows the DD and LC methods are both within about 7% of the analytical diffusion solution. The LC also continues to perform well with relative errors consistently lower than DD and Step. Unlike the radial flux profile errors, the axial errors are more uniform and do not show signs of a sudden increase in the relative error near the edges of the core. Since the axial direction is modeled explicitly, these results are expected. The errors can be reduced by refining the mesh; however, the issue of computer memory limitations and loss of speed effect the final overall decision of the user to continue refining the mesh grid. For teaching purposes, the refined grids of figures 4.5 through 4.9 show the correct reactor physics behaviors and show that the 3D-TRAN code can model a right circular cylindrical reactor.

The final discussion of the cylindrical reactor involves the eigenvalue calculation. As mentioned in the discussion of the parallelepiped, the convergence on the eigenvalue for the cylindrical reactor took a few thousand iterations. The linear extrapolation acceleration method mentioned earlier was used to speed calculations. The results are listed in table 4.3. This table shows the three spatial quadratures with DD repeated twice, once with no acceleration for a timing comparison. The diffusion solution comes directly from Harman (2001) with the eigenvalue calculated numerically. The nodal spacing on the diffusion equations was 15 cm, which is close to the spacing chosen for the transport calculation. The relative error was taken for each spatial quadrature using the diffusion calculated K value as the comparison value. The results show that DD and LC quadratures very accurately match the predicted diffusion eigenvalue to within 0.07% with LC performing slightly better at 0.05%. The Step method is within about 6% with errors caused by the coarseness of the spatial grid.

These final results show that the isotropic fission source in the transport code is correctly modeling the fission source in the diffusion code. Also, the results show that the eigenvalue power iteration technique utilized in 3D-TRAN is producing accurate results. As for convergence acceleration, the technique does show substantial reductions in overall code iterations, especially when combined with the initial guess generator. This

| Method | K | Total Iterations | Time (sec) | Convergence Acceleration | K & Flux Tolerance | Rel % Error in K |
|---|---|---|---|---|---|---|
| **Diffusion** | 1.12109 | 83 | NA | NA | 1.00E-05 | NA |
| **DD** | 1.12187 | 1449 | 351.74 | No | 1.00E-04 | 0.070 |
| **DD** | 1.12187 | 601 | 145.17 | Yes | 1.00E-04 | 0.070 |
| **Step** | 1.05909 | 208 | 52.45 | Yes | 1.00E-04 | 5.530 |
| **LC** | 1.12161 | 121 | 1501.6 | Yes/initial guess | 1.00E-04 | 0.047 |

**Table 4.3: Cylinder Reactor Eigenvalue Test Results**

was seen in the LC results where convergence was achieved in 121 iterations verses the 1449 iterations DD required for a non-accelerated run. Assuming that LC takes about 1449 iterations with no acceleration or initial guess, this is an improvement of 91%; however, the LC method still took 25 minutes to calculate the eigenvalue (far better than a projected 6 hours un-accelerated). These results suggest acceleration of the transport is definitely possible and will not have any adverse effects on the final answer if implemented correctly. The follow section will discuss the use of the linear extrapolation acceleration method and its overall stability in terms of homogenous problems.

## Convergence Acceleration of the Scalar Flux

Before discussing the heterogeneous fission reactor, it is necessary to introduce the concept of convergence acceleration applied to the scalar flux profile in neutron transport. As briefly seen in the homogeneous reactor results, neutron transport solutions to the criticality eigenvalue problem run over many iteration due to the slow convergence of the scalar flux and the eigenvalue. As seen in table 4.3, the homogenous cylinder reactor required 1449 iterations to converge to a relative tolerance of $10^{-4}$. In more refined tolerances, this iteration number will at least double or triple as a minimum. The use of the LC method alone will require many hours or even days to reach a final converged solution. This result is unacceptable and needs improvement. The need for acceleration of the solution to its final tolerance is necessary to decrease overall run times and make the 3D-TRAN code more viable as a teaching tool.

Convergence acceleration provides a means for reducing overall run time and achieving a converged solution in less iterations. For reactor eigenvalue problems, the non-accelerated convergence behavior is slow and very asymptotic in nature. To show this behavior, a test problem was run using the same geometry and material cross sections as the right circular cylinder reactor demonstrated in figures 4.5 – 4.9. The behaviors of the center most cell of the reactor and an edge cell were graphed to examine the slow convergence behavior over number of iterations. The results are graphed in Figures 4.10 and 4.11.

Figure 4.10 shows the center cell of the cylindrical reactor where the flux is maximized. The initial iteration begins with a flux value of 1.0, which continues to build with each iteration until reaching a point where marginal improvement in the flux value is achieved. This asymptotic convergence behavior is very slow as seen in the graph. The second type of convergence behavior seen in the cylindrical homogenous reactor calculation is shown in figure 4.11. This graph shows the scalar flux in an edge cell of the

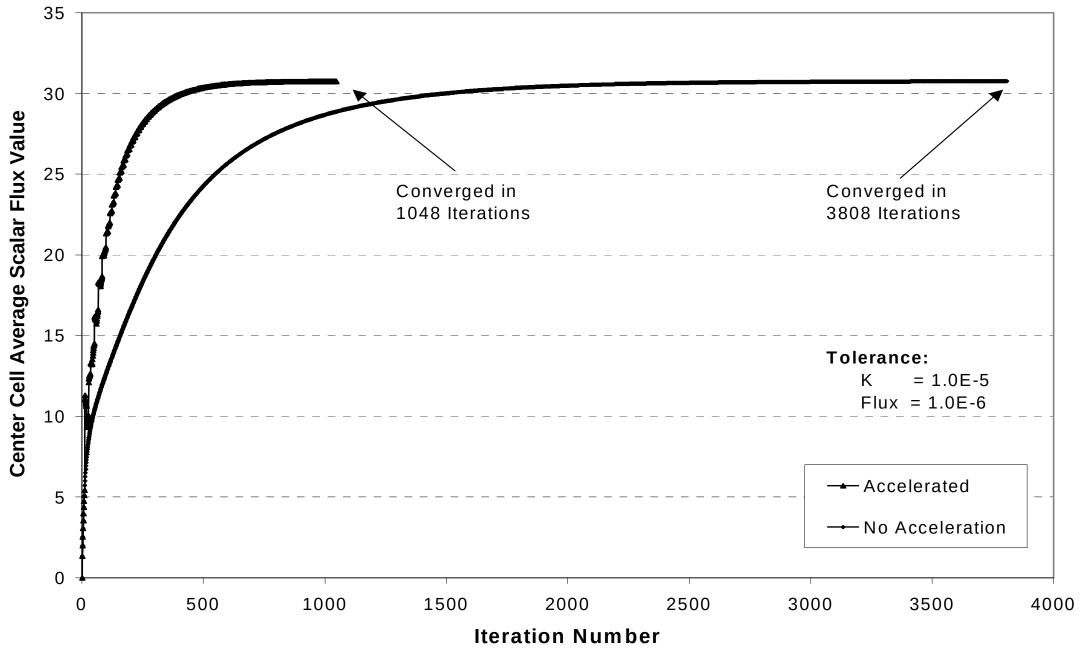

**Figure 4.10: Convergence Behavior of Scalar Flux for Center Reactor Cell**

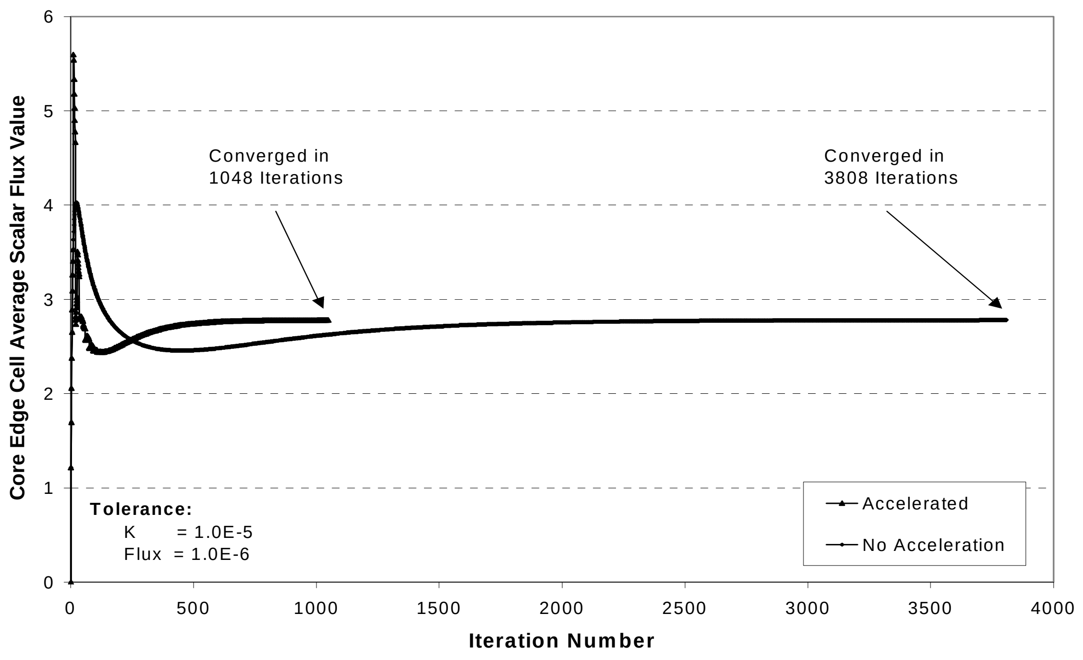

**Figure 4.11: Convergence Behavior of Scalar Flux for Reactor Edge Cell**

reactor where the flux is considered at a minimum. The flux in this cell is also initialized to 1.0 and builds very quickly then diminishes below the converged solution before slowly and asymptotically rising to the final converged answer. The end result is that both cells, the center cell and the edge cell, each behave asymptotically in the limit as the number of iterations becomes large. These behaviors suggest that convergence acceleration by some form of linear extrapolation may be achieved.

Lewis and Miller (1993) suggest that a simple extrapolation technique applied to the scalar flux profile at the end of each iteration will improve overall convergence speed by reducing the total number of iterations required. The suggested method utilizes information from the old or previous iterate as well as the new iterate. The equation for acceleration of the flux is as follows:

$$\phi_{i+1} = \alpha\left(\tilde{\phi}_i - \phi_i\right) + \phi_i \tag{4.6}$$

where $\alpha$ is an over-relaxation factor, $\tilde{\phi}_i$ is the unaccelerated new flux iterate, $\phi_i$ is the previous flux iterate, and $\phi_{i+1}$ is the accelerated new flux iterate. The over-relaxation factor is suggested to fall between one and two for improved convergence (Lewis and Miller, 1993). The implementation of this convergence acceleration technique yielded marginal improvements in the convergence on the scalar flux. The number of iterations required to achieve the converged answer was reduced by only about 30% using an over-relaxation factor of two. This result was good, but there is a better technique that still utilizes equation (4.6), an iterative shooting method.

The iterative shooting method applies the linear extrapolation method of equation (4.6), but does not implement the acceleration every iteration. Instead, the shooting method applies a larger over-relaxation value (greater than two), but performs the extrapolation method only every couple of iterations. The shooting technique takes advantage of the asymptotic convergence behavior seen in figures 4.10 and 4.11 by using the slope of the previous iterate and the unaccelerated iterate to project a new iterate value further along the iteration curve then the unaccelerated value. The only problem is determining the optimum over-relaxation factor as well as the optimum number of iterations to perform before shooting an accelerated iterate. This initially proved to be problematic. The solution to finding optimum performance stemmed from an analysis of the cylindrical reactor test problem used to generate figures 4.10 and 4.11.

The cylindrical test problem was run, keeping all spatial grids, material parameters, and relative tolerances constant. The only values changed were the over-relaxation constant and the number of iterations to perform before shooting a new flux iterate. The tests began by fixing the number of iterations to perform before shooting and then varying the over-relaxation parameter, $\alpha$. The key in this analysis was to identify the over-relaxation parameter that provided the minimum number of iterations to convergence of the solution. The analysis continued by increasing the shooting iteration number and finding the new over-relaxation constant. The results are summarized in figure 4.12.

The graph provides the optimum over-relaxation parameter for any given number of iterations to perform before shooting a new accelerated iterate (for this test problem). The method is stable for any over-relaxation value that is below the stability line in figure 4.12 for any fixed shooting iteration number. The method quickly becomes unstable if alpha exceeds the optimum over-relaxation parameter for any given shooting iteration number. This instability is caused by over shooting the flux iterate where the convergence

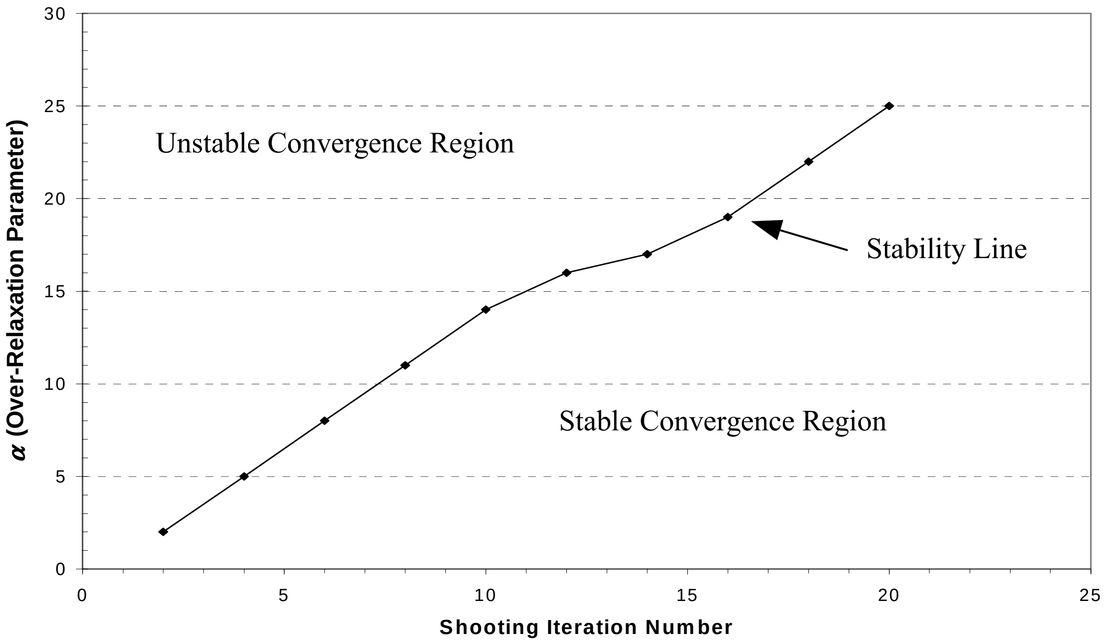

**Figure 4.12: Optimum $\alpha$ per Shooting Iteration Number.**

behavior in the computational cell is disrupted beyond recovery. The shooting method linear extrapolation method utilizes the fact that as the accelerated flux iterate is shot, the code performs a few unaccelerated iterations to allow the solution to settle down into its normal convergence pattern. If shooting occurs before this happens, the convergence behavior in the cell has not settled down and the new accelerated iterate will be based on a slope that may shoot the new answer in the wrong direction. As this continues, the behavior begins oscillating until convergence to a stable solution is completely destroyed.

Now, referring back to the test problem used to define figure 4.12, the optimum alpha values along the stability line produced a close distribution of final iterations to problem convergence. In other words, as the alpha parameter is varied along the stability line, the number of iterations required to converge to the final solution was within 1.5 standard deviations (or 25 iterations) of the average.  The average iteration to convergence was 433 at a relative tolerance of $10^{-4}$ for both the flux and the eigenvalue. The unaccelerated converged solution at this tolerance was 1449 iterations. Therefore, the linear extrapolation shooting method improved overall convergence speed by about 70%. This result is better than achieved before with the linear extrapolation method and better meets the goals of this project to provide a usable teaching tool for students to learn reactor/neutron particle physics.

As a final test, the relative tolerance was set back to $10^{-6}$ for the flux and $10^{-5}$ for the eigenvalue for the cylindrical reactor problem. The accelerated cell convergence behaviors are graphed in figures 4.10 and 4.11 as a direct comparison to the unaccelerated convergence behavior.  Here, the shooting extrapolation acceleration method achieved a 72% increase in convergence speed.

The results presented in this section show the value of having a convergence accelerator on the transport solution when eigenvalue calculations are desired. Although the convergence technique employed here was analyzed for homogenous reactors, the technique can be applied to heterogeneous reactors. The only questions that remain are how well will the accelerator perform in a heterogeneous problem and what is the shape, if definable, of the optimum over-relaxation parameter curve. These questions are not answered in this research and provide a starting point for future analysis on convergence acceleration applied to neutron transport criticality calculations. The convergence accelerator developed in this research is used on the remaining heterogeneous reactor problems in this chapter to provide some initial data for its performance with these types of reactor problems.

## The Heterogeneous Reactor

In the previous section, the performance of the neutron transport homogenous reactor was compared to the analytical homogenous reactor diffusion solution.  The neutron transport code performed very well in reproducing flux profiles that agreed with the analytical diffusion solution. In this section, the heterogeneous reactor is explored. The use of an analytical diffusion comparison is omitted due to complexity in defining a heterogeneous, analytical or numerical diffusion solution. Instead, the neutron transport based 3D-TRAN is used to generate sample heterogeneous problems the student may find interesting in terms of demonstrating reactor physics principles and practices. These problems are designed to show how the 3D-TRAN code can be utilized as a teaching tool to help students see and understand the neutron flux behavior inside a multi-region reactor core.

The problems of interest consist of modeling a reactor unit fuel cell, a two-region fuel assembly problem, and two full size reactor core fuel loading configurations. The unit fuel cell serves as the method for determining the region dependent homogenized cross sections used in the remaining reactor test problems. The two-region assembly problem will demonstrate the source-sink coupled relationship between two fuel assemblies of different weight percent uranium-235. The final core loading configuration problems will demonstrate some basic principles of nuclear fuel management. All of these tests are designed based on the properties of a pressurized water reactor (PWR) core configuration used in most of the present day, commercial nuclear power utilities.

## The Unit Fuel Cell

The commercial nuclear reactor is made of many components that are engineered to fit together in such a way to optimize the fission process of uranium-based fuel. The fuel is bundled together in rods that are held together by many support structures called fuel assemblies. These assemblies consist of a reproducible matrix of fuel rods that are arranged in a square lattice structure at equal distances from each other. These fuel rods contain the uranium fuel pellets that drive the nuclear fission reaction. Therefore, the fuel rod itself is the fundamental fuel element in a nuclear reactor.

The basic physics of the fuel rod can be examined by assuming that the fuel rod dimensions are small compared to the reactor as a whole. In many cases, this is true since fuel rods are only about 0.4 inches in diameter while the core may be about 240 feet in diameter. The core is generally about 12 feet tall with fuel rods being equally as long. If we examine a specific point on a fuel rod in the center of the core about half way up the core height, one can assume that the physical behavior of the neutron flux at this location is relatively constant in the axial direction. The fuel rod sits in a lattice of fuel rods that are, for the most part, identical in composition and dimensions. The mean free path of the neutrons is generally on the order of the distance from one fuel rod to the next adjacent fuel rod. Therefore, the fuel rod at this point (half way up the height) in the core can be reduced to a unit cell calculation in which the height of the cell is one (for three-dimensional calculations) and the sides of the cell consist of a square with dimensions greater than the diameter of the fuel rod plus some of the surrounding water or moderator. Figure 4.13 shows a typical unit fuel cell.

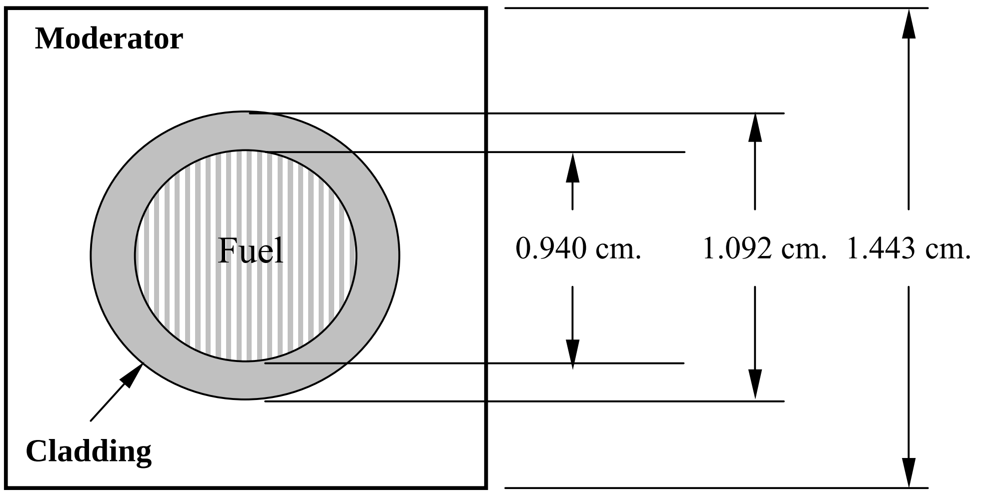

**Figure 4.13: Unit Fuel Cell using TMI dimensions**

For this problem, the Three Mile Island (TMI) initial fuel loading dimensions were used to develop the test problem dimensions of the fuel rod (Feltus, 1995). These dimensions are also listed in figure 4.13 to show the relative locations on the fuel cell. The fuel cell pitch is equal to one side of the square unit cell. The diagram was simplified from its original form by removing the gap that usually exists between the cladding and the fuel pellet. This can be done because one can assume that little neutron interaction occurs in the gap void space between the cladding and the fuel pellet. The problem therefore reduces to a three region heterogeneous problem. Symmetry boundary conditions are used to simulate the existence of the unit fuel cell in a lattice of equal fuel cells in the reactor core. These ideal conditions now allow for simple examination of the neutron flux inside the fuel rod in the radial direction starting at the center of the fuel cell.

The cross sections chosen for this test problem consist of a set of microscopic four-group cross sections referenced from table 7-1 of Duderstadt and Hamilton (1976)[5]. These cross sections allowed for development of macroscopic cross sections based on the weight percent (w/o) of uranium 235. The fuel chosen consisted of 2 w/o, 3 w/o, and 4 w/o fuel. The input data is summarized in Appendix E for brevity. The cylinder estimation routine discussed in the homogenous cylinder reactor section was used to approximate the curved boundary of the cladding and fuel pellet. Quarter pellet symmetry was used to model the fuel cell by taking advantage of symmetry and to better refine the curved boundary. A relative tolerance of $10^{-4}$ was chosen for the flux and eigenvalue convergence.

[5] These are ideal cross sections and are not assumed to be completely accurate in capturing the cross section behavior in a real reactor. These were chosen merely to demonstrate general reactor physics behaviors.

Figure 4.14 shows the one dimensional flux profile for 2 w/o fuel for the Step, DD, and LC spatial quadratures. The graph shows the flux profile decreasing toward the edge of the fuel pellet until reaching the outer surface of the cladding. At this point, the flux increases slightly in the LC and DD methods indicating more scatter present near the cladding/moderator interface. The Step method does not show the magnitude of the increase in the flux at this boundary and appears to underestimate the flux profile through the cladding to moderator interface. The moderator has a higher scatter cross-section than the cladding and therefore should produce the increase in flux at the fuel rod boundary as seen by the DD and LC methods. The magnitude of the flux increase will be purely dependent on the cross-section data used in the problem.

The flux profile in a unit fuel cell also provides another valuable piece of information, the average flux in each region. For heterogeneous reactor problems, the ability to model every fuel cell in the reactor is not desirable. Instead, unit fuel cells are

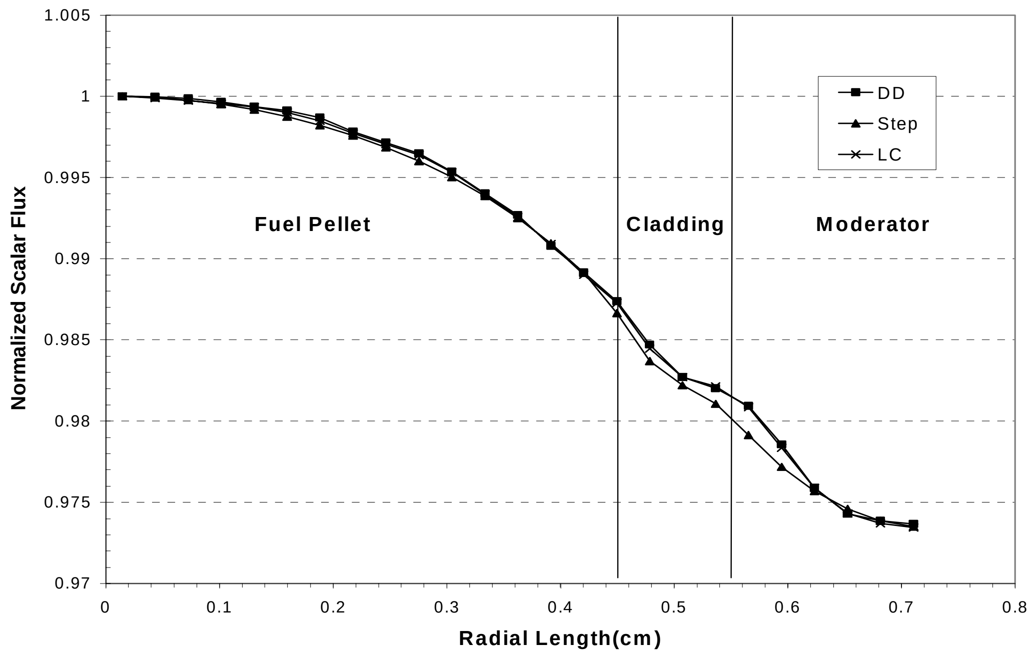

**Figure 4.14: Fuel Cell Scalar Flux Profile Comparison**

used to generate cell homogenized cross section data to simplify the input to large reactor numerical models. The cell homogenization process requires the region scalar average flux to generate flux weighted homogenized cross sections. The homogenized cross section can then be calculated using the following equation over the unit fuel cell:

$$\left\langle \Sigma_{xg} \right\rangle = \frac{\Sigma_{xFg}\overline{\phi}_{Fg} + \Sigma_{xCg}\overline{\phi}_{Cg} + \Sigma_{xMg}\overline{\phi}_{Mg}}{\overline{\phi}_{Fg}V_F + \overline{\phi}_{Cg}V_C + \overline{\phi}_{Mg}V_M} \tag{4.7}$$

where for the fuel(F), cladding(C) and moderator(M):

$$\overline{\phi}_{Fg} = \frac{\int_{V_F} \phi(\vec{r}, E_g) d^3\vec{r}}{\int_{V_F} d^3\vec{r}} \tag{4.8}$$

$$\overline{\phi}_{Cg} = \frac{\int_{V_C} \phi(\vec{r}, E_g) d^3\vec{r}}{\int_{V_C} d^3\vec{r}} \tag{4.9}$$

$$\overline{\phi}_{Mg} = \frac{\int_{V_M} \phi(\vec{r}, E_g) d^3\vec{r}}{\int_{V_M} d^3\vec{r}}. \tag{4.10}$$

The subscript *g* refers to the energy group and the subscript *x* on the cross-section refers to any cross section of interest. The 3D-TRAN code was modified to include this calculation over a unit fuel cell. The homogenized unit fuel cross-sections for the 2 w/o fuel are given in table 4.4.

The unit fuel cell test problem was also run for 3 w/o and 4 w/o fuel. The flux profiles were similar to figure 4.14 in appearance, but differed slightly in the magnitude of the flux in each region. Of course, this flux variation is caused by the change in the fuel cell cross sections for each problem. The higher w/o fuel showed an increase in the flux profile in the fuel with a slightly elevated rise in the flux at the cladding/moderator

|  | Energy Group |  |  |  |
|---|---|---|---|---|
| 2 w/o | 1 | 2 | 3 | 4 |
| $\Sigma_t$ | 1.2276E-01 | 2.8820E-01 | 4.3366E-01 | 1.5311E+00 |
| $\nu\Sigma_f$ | 1.0066E-02 | 4.9549E-04 | 7.7312E-03 | 1.5234E-01 |
| $\Sigma_{g\text{-}g+1}$ | 5.1044E-02 | 7.3730E-02 | 7.6674E-02 | 0.0000E+00 |
| $\sigma_{gg}$ | 6.2846E-02 | 1.9952E-01 | 3.3657E-01 | 1.4336E+00 |
| 3 w/o |  |  |  |  |
| $\Sigma_t$ | 1.2277E-01 | 2.8821E-01 | 4.3650E-01 | 1.5857E+00 |
| $\nu\Sigma_f$ | 1.0209E-02 | 7.4320E-04 | 1.1575E-02 | 2.1982E-01 |
| $\Sigma_{g\text{-}g+1}$ | 5.1047E-02 | 7.3732E-02 | 7.6742E-02 | 0.0000E+00 |
| $\sigma_{gg}$ | 6.2845E-02 | 1.9956E-01 | 3.3676E-01 | 1.4563E+00 |
| 4 w/o |  |  |  |  |
| $\Sigma_t$ | 1.2277E-01 | 2.8822E-01 | 4.3933E-01 | 1.6367E+00 |
| $\nu\Sigma_f$ | 1.0351E-02 | 9.9082E-04 | 1.5405E-02 | 2.8197E-01 |
| $\Sigma_{g\text{-}g+1}$ | 5.1053E-02 | 7.3734E-02 | 7.6812E-02 | 0.0000E+00 |
| $\sigma_{gg}$ | 6.2839E-02 | 1.9960E-01 | 3.3696E-01 | 1.4779E+00 |

**Table 4.4: Homogenized Cross-Section Data ($cm^{-1}$) by w/o U-235 Fuel**

interface. The results of the 3 and 4 w/o fuel cell homogenization are provided in table 4.4 also.

The cell-homogenized cross-sections were calculated using all three spatial quadratures. All three methods produced homogenized cross sections that were in agreement with each other out to three decimal places. The tolerance run on the cross sections was only $10^{-4}$;therefore, the three spatial quadratures are in agreement out to the convergence tolerance limit. For these sample problems, this tolerance was considered good enough since actual real reactor cross-section data was not being used. As a final summary on the unit fuel cell problem, table 4.5 lists the eigenvalues for each type of fuel and compares the number of iterations to convergence, both accelerated and unaccelerated.

| Fuel | K | Accelerated Iteration | Unaccelerated Iteration | Relative % Increase in Speed |
|---|---|---|---|---|
| **2 w/o** | 1.101 | 319 | 883 | 63.9 |
| **3 w/o** | 1.188 | 316 | 879 | 64.1 |
| **4 w/o** | 1.238 | 317 | 876 | 63.8 |

**Table 4.5: Eigenvalue and Performance data for Uranium Based Fuel**

As an example, the data in table 4.5 only shows the DD spatial quadrature values. The LC numbers identically matched DD out to the tolerance limit while the Step method slightly under predicted the eigenvalues. For this problem, the DD and LC answers were considered "correct" and used for the remaining test problems in this chapter. This assumption is based on the fact that both DD and LC were within the optical thickness ratio limit (summarized in figure 3.9), and is therefore considered a stable neutron transport solution to the unit fuel cell test problem.

Table 4.5 shows that as the w/o of fuel is increased, the eigenvalue also increases. This result is expected and does show that more thermal based fissions occur in uranium-235 fuel with a higher percentage of U-235 concentration. At the specified tolerance of $10^{-4}$, the DD spatial quadrature required about 880 iterations to converge. With the addition of acceleration, the overall required number of iterations to convergence dropped to only about 320 iterations. This result shows a relative increase in speed by about 64%. This result is surprisingly good since the convergence accelerator was designed and optimized on the homogenous reactor problems presented earlier in the chapter. The LC and Step methods were not examined for acceleration, although the LC method is expected to perform as well as DD since it took about the same number of unaccelerated iterations to converge to a solution. If DD is used as the initial guess, then the performance for LC is expected to increase to about 85% faster than the unaccelerated solution. This estimate is based on all the evidence presented thus far on the initial guess generator in combination with convergence acceleration.

## Two-Region Assembly Problem

The previous section discussed how the unit fuel cell test problem can be used to generate flux weighted homogenized cross sections. These cross sections are homogenized over the dimensions of the unit fuel cell. Now, if we construct a fuel assembly consisting of a group of unit fuel cells containing one of the fuel types listed in table 4.4, the homogenized cross sections for that unit cell can be used to describe the entire assembly.  Of course, this ideal assembly neglects any support structure or neutron absorbers that may be present in a real assembly.

For this test problem, two different fuel assemblies were used, 2 w/o and 3 w/o U-235. Each assembly assumes that there are 225 fuel cells arranged in 15 x 15 matrix. The dimension for one side of the square assembly, called the assembly pitch, is 21.6cm (rounded up to 22cm for the test problem). The assemblies are also approximately 12 feet high or 364 cm tall. The test problem is set up to model a small section of the reactor core where two assemblies of different w/o are arranged next two each other. The pattern chosen for the assembly arrangement consists of a checkerboard design common in most commercial power reactors. For ease of modeling, the assemblies are arranged in a four square pattern shown in figure 4.15. The boundary conditions for this problem are taken through the xy plane centerline of the outer assemblies to take advantage of symmetry boundary conditions available in a reactor core consisting of the repeated checkerboard pattern of figure 4.15. The height of the assemblies was reduced to 22 cm in order to

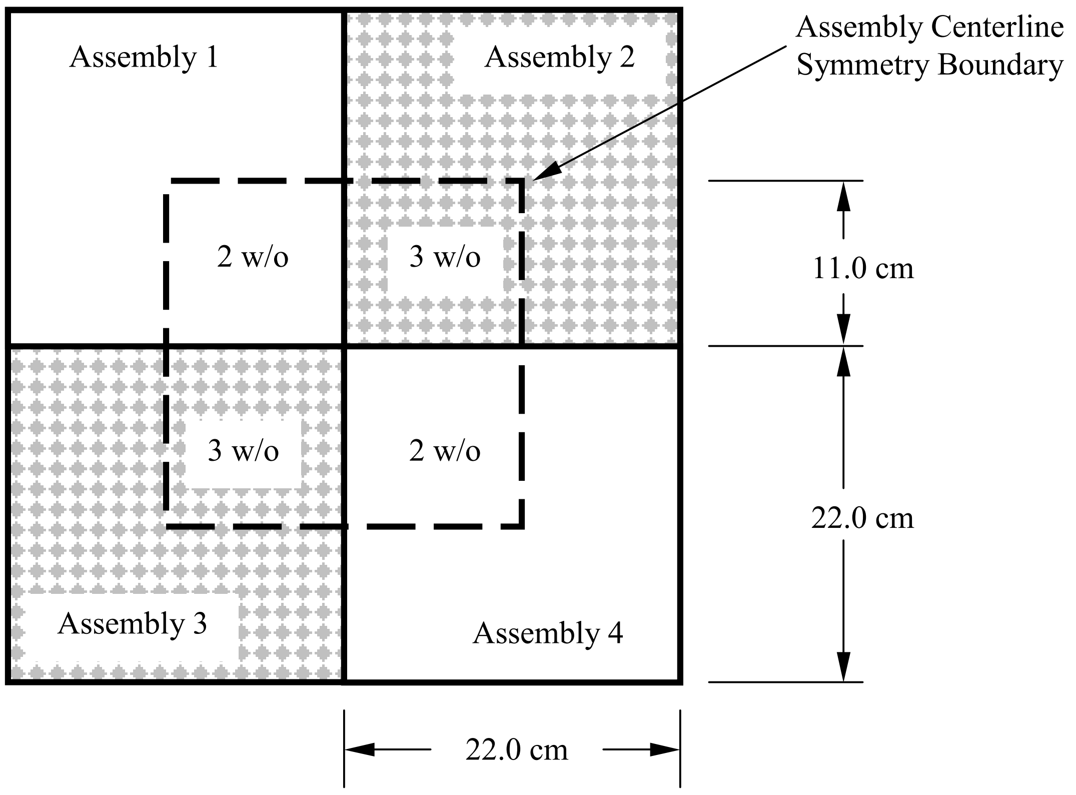

**Figure 4.15: Checkerboard Two-Region Assembly Problem Configuration**

examine a "cube" with z maximum and minimum boundaries taken as ideal symmetry conditions. The resulting test problem models an ideal assembly arrangement in the center of the reactor core.

The problem was checked for convergence, as the spatial mesh was refined. The dimensions of this problem are much larger than the unit fuel cell and mesh refinement was expected to affect the convergence of the eigenvalue and flux profile. Figure 4.16 shows the peak to average flux profile for the assembly arrangement of figure 4.15. The peak to average flux was chosen because fuel management techniques usually involve attempts at designing fuel configurations that reduce the peak to average power ratio as small as possible to flatten the power shape across the core (Cochran, 1990). Since the

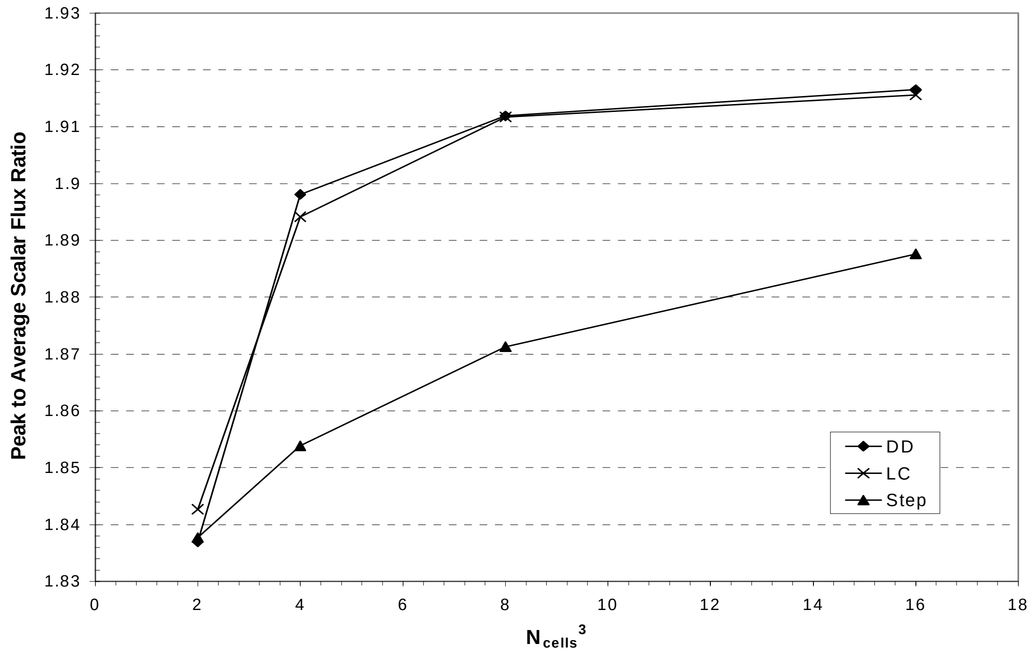

**Figure 4.16: Peak to Average Flux Ratio for Two-Region Assembly Problem**

reactor power is proportional to the flux, the general behavior of the peak to average flux ratio should be consistent with the peak to average power ratio.

Figure 4.16 shows that as the mesh is refined, the DD and LC methods tend to converge towards a peak to average ratio of about 1.92. The Step method is under predicting the peak to average ratio, but appears to be converging towards the same value as DD and LC. The refinement of the mesh better resolves the peak flux value and thereby causes the slower convergence to the peak value. The average flux changes very little as the mesh is refined. Therefore, convergence on the peak ratio value is expected with continued refinement of the mesh.

Figure 4.17 shows the convergence of the eigenvalue over the two-region assembly problem. The graph shows that DD and LC tend to converge on an eigenvalue of 1.2157. The LC method has predicted a value of 1.2157 in the initial coarse mesh refinement and tends to oscillate around this answer since the solution convergence has

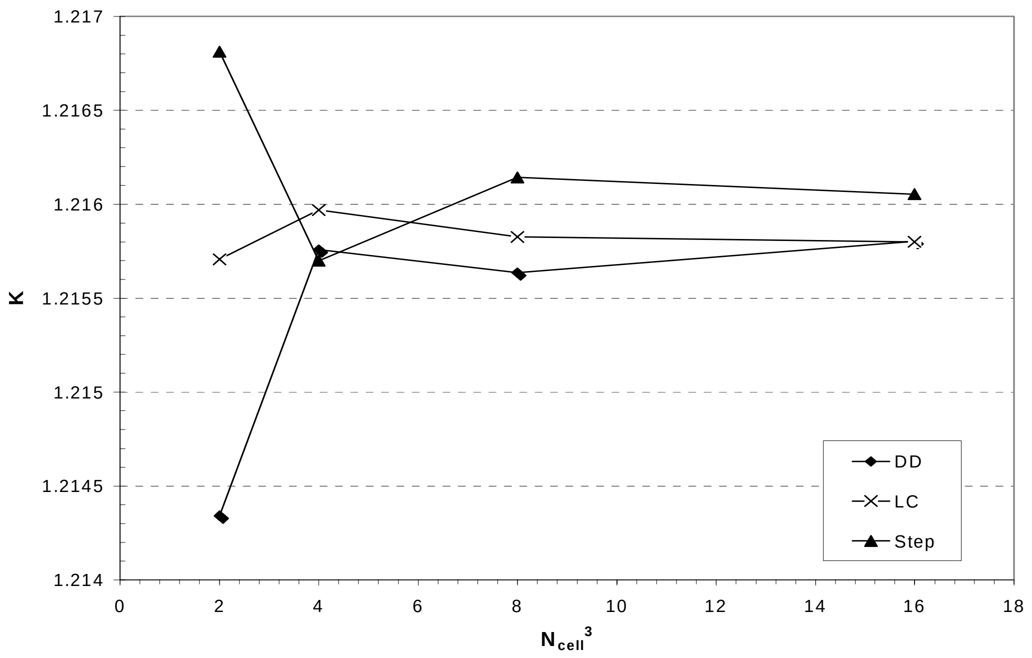

**Figure 4.17: Eigenvalue Convergence Behavior for Two-Region Problem**

been truncated by the smaller problem relative tolerance of $10^{-4}$. The DD method eventually converges in on this eignevalue to within three digits after the first mesh refinement. The Step method also appears to be converging to the 1.2157 eigenvalue; however, the convergence is relatively slow as expected since the method is linear in its order of error reduction.

The convergence behavior for the DD and LC methods is presented in table 4.6. This table shows both the accelerated and unaccelerated number of iterations to convergence for both methods. The LC method also used the DD initial guess generator routine to help reduce the number of iterations to convergence even further. The Step method was not evaluated since its overall iterations to convergence very close mimicked the DD convergence behavior. The mesh refinement factor of 16 was not explored because the overall problem run time was becoming very long. The LC method took about 2900 seconds to calculate the accelerated (refinement of 16) answer. If the

| Mesh Refinement DD | K | Iterations to Convergence | Accelerated Iterations to Convergence | Relative % Increase in Speed |
|---|---|---|---|---|
| 2 | 1.2143 | 119 | 106 | 10.9 |
| 3 | 1.2158 | 137 | 114 | 16.8 |
| 8 | 1.2156 | 144 | 121 | 16.0 |
| 16 | 1.2158 | NA | 129 | NA |
| **Mesh Refinement LC** | | | | |
| 2 | 1.2157 | 117 | 22 | 81.2 |
| 3 | 1.2160 | 136 | 18 | 86.8 |
| 8 | 1.2158 | 141 | 26 | 81.6 |
| 16 | 1.2158 | NA | 10 | NA |

**Table 4.6: Two-Region Assembly Convergence Data**

unaccelerated LC answer was run, it is estimated that the code would take well over 12 hours to calculate an answer assuming the number of iterations to convergence was at least 150. The important feature of table 4.6 is that the convergence accelerator only increased the DD calculation speed by about 16%. The unaccelerated convergence was relatively quick (i.e. less than 150 iterations). The convergence accelerator tends to perform better when a larger number of unaccelerated iterations is required to converge to an eigenvalue. The use of DD as the initial guess appears to have contributed to the relatively few iterations required by LC to converge since the answers for each method were virtually identical within the first three digits of the solution.

Finally, the two dimensional LC flux profile for the two-region assembly problem is presented in figure 4.18. This graph shows the relative difference in the flux between each region. The lower flux is present in the lower w/o fuel (2 w/o). The higher w/o fuel (3 w/o) has a higher flux profile in its region because of the higher concentration of U-235 causing more thermal fissions. The higher w/o fuel tends to drive the fission reaction

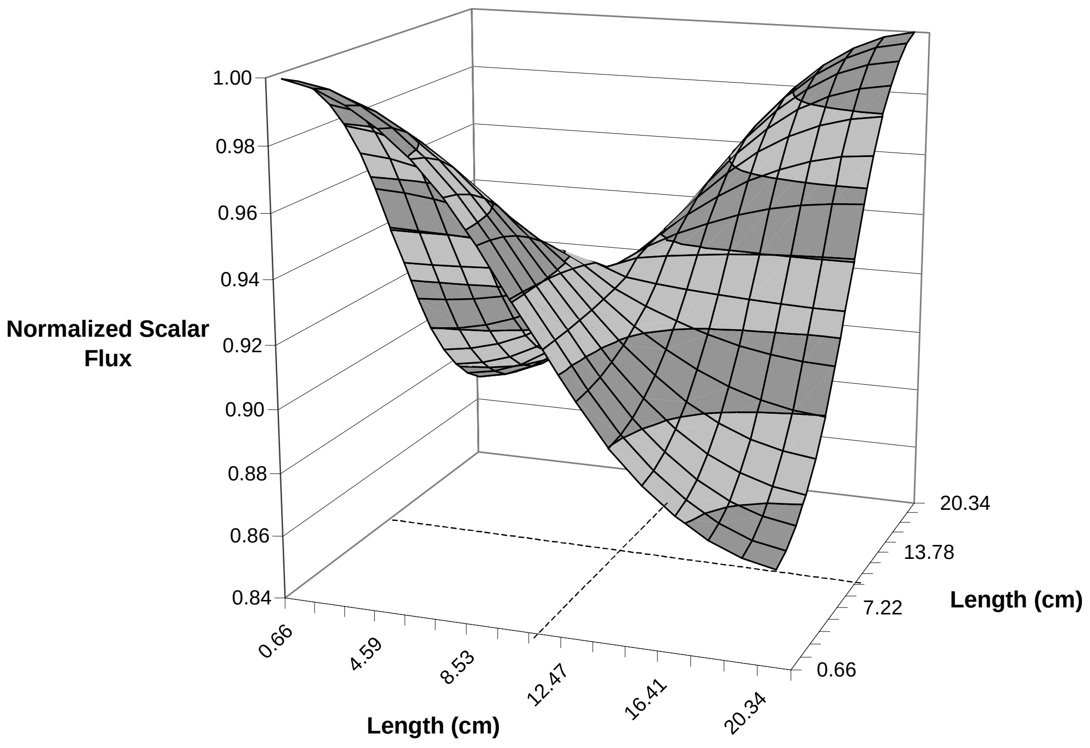

**Figure 4.18: Two-Region Assembly, LC Generated Flux Profile**

and acts more like a source. The lower nearby w/o fuel tends to absorb more neutrons and thereby appears to look like a neutron sink for the higher w/o fuel. These results are expected and demonstrate the source-sink neutron flux relationship between different w/o fuels. This relationship serves as the foundation for fuel management practices that utilize this information to try and optimize core-loading schemes by flattening the overall flux profile across the core (Cochran, 1990). The next section will demonstrate some basic fuel management principles using the 3D-TRAN code to determine the best optimum core configuration.

**Basic Fuel Management: Reactor Core Fuel Loading Patterns**

In the previous section, a two-region assembly problem was presented to test the 3D-TRAN code with a heterogeneous reactor problem and to show the general neutron physics behavior between adjacent fuel assemblies of different fuel compositions. The source-sink philosophy of developing reactor core fuel loading patterns uses the idea that lower w/o fuel absorbs neutrons from nearby higher w/o fuel. This knowledge of fuel interaction allows nuclear fuel mangers to develop fuel loading schemes that reduce power peaks in the core and help maintain a flatter power distribution. The flatness of the power distribution is measured based on the peak to average power ratio (an average core ratio close to one indicates a relatively flat distribution). A flat power distribution is desirable because the fuel will burn more uniformly throughout the core allowing for the maximum possible energy to be extracted from the fuel (Cochran, 1990). This philosophy is very economical and efficient for nuclear power reactors run by utility companies.

In order to efficiently examine the peak to average power ratio, the 3D-TRAN code was modified to include a routine that calculates the fraction of the total core average power in each assembly. First, the average power per assembly is defined as follows:

$$\overline{P} = \frac{P}{N} = \frac{E_f \sum_{j=1}^{N} \left[ \overline{\Sigma}_{fj} \overline{\phi}_j V_j \right]}{N} \tag{4.11}$$

where $P$ represents the total core power, $E_f$ represents the energy per fission, $N$ represents total number of assemblies in the core, $\overline{\Sigma}_{fj}$ represents the average fission macroscopic cross section for the jth assembly, $\overline{\phi}_j$ represents the average scalar flux for the jth assembly, and $V_j$ represents the volume of the jth assembly. The fraction of average power produced by assembly j then becomes:

$$NP_j = \frac{P_j}{\overline{P}} = \frac{N\overline{\Sigma}_{fj}\overline{\phi}_j}{\sum_{j=1}^{N}\overline{\Sigma}_{fj}\overline{\phi}_j} . \tag{4.12}$$

Equations (4.11) and (4.12) are referenced from Chapter 5 of Cochran (1990). These equations give 3D-TRAN the capability to output relative power maps that will directly show which assemblies in the core are higher or lower to the average core power. Peak power locations can then be easily identified and associated with a given assemblies fuel properties. This capability is essential in developing reactor fuel loading schemes that reduce peak power levels in the core.

In this section, the 3D-TRAN code is used to model two basic fuel loading patterns. These patterns are the OUT-IN loading scheme, and the low-leakage core loading (Cochran, 1990). These loading configurations are explained in more detail in the following sections. The reader is referred to Chapter 6 of Cochran (1990) for more details regarding general fuel management practices.

## OUT-IN Fuel Loading Pattern

The first fuel pattern of interest is called the Out-In Loading scheme (Cochran, 1990). The reactor is divided into X batches with each batch containing approximately the same number of assemblies. The lowest w/o fuel is placed in a circle in the center of the core. The next highest w/o fuel is placed in a ring around the center batch. This continues until the highest w/o fuel surrounds the core along its perimeter. The main advantage to this loading scheme is that the peak power is lowered and shifted from the center of the core out towards the edge of the core. The main disadvantage of this scheme is that the higher w/o fuel along the perimeter generates higher power and thus higher neutron levels that strike the pressure vessel containing the core. This is not desirable and will cause radiation damage to the pressure vessel[6].

For this test, the cross section data of table 4.4 is used to model homogenized fuel assemblies containing 2 w/o, 3 w/o, and 4 w/o U-235. Each fuel assembly is 22 cm square and 364 cm high. The assemblies were placed in a quarter core configuration. Symmetry boundary conditions were used to simulate a quarter core symmetric fuel loading. A symmetry boundary was also placed half way down the height of the fuel assemblies since no axial variations in the fuel composition are present in this problem. The remaining boundaries were chosen as vacuum[7].

Figure 4.19 shows the fuel assembly configuration map. The core configuration

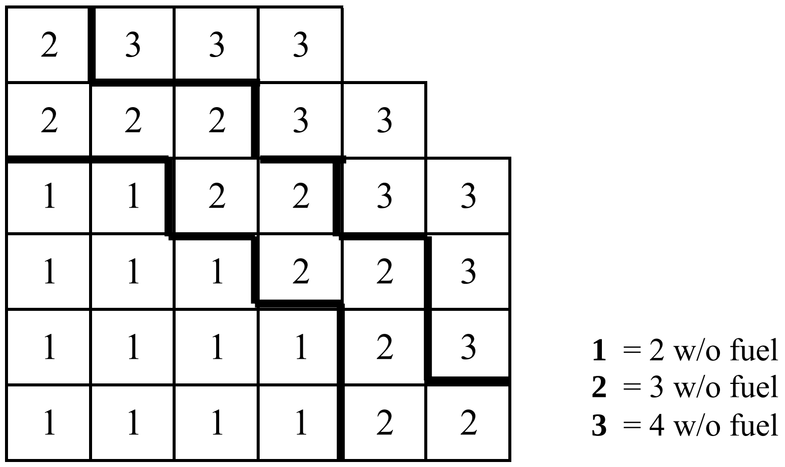

**Figure 4.19: Quarter Core OUT-IN Fuel Loading Configuration**

[6] Once again, the reader is referred to Cochran (1990) for more details on low leakage core configurations and their importance.

[7] In reality, the core is surrounded by a reflective material designed to reflect a percentage of the neutrons exiting the reactor along the perimeter back into the core. This addition boosts the neutron flux along the core perimeter and helps to reduce overall core leakage.

map was designed based on a total of 109 fuel assemblies with 37 assemblies in batch one, 36 assemblies in batch two, and 36 assemblies in batch three. The batch numbers in figure 4.19 correspond to a specific w/o of fuel assigned as listed in the map legend. The problem was run using a mesh size that split each assembly in the xy plane into four sub cells. This was done to maintain smaller calculation cells for DD and LC. The final spatial mesh consisted of a 12x12x20 mesh so that all cells were 11cm by 11cm in xy and 9.1 cm in z.

The OUT-IN loading pattern was run using all three spatial quadratures. The resulting criticality eigenvalues for this core are listed in table 4.7. Both DD and LC predict the reactor cores eigenvalue to within four digits. Once again, the two methods are in agreement out to the relative convergence tolerance of $10^{-4}$. The Step method under predicted the eigenvalue with a relative percent difference from Step and DD of 8.0%. The number of iterations to convergence is also presented to show improvements made by the convergence acceleration method developed earlier in this chapter. The DD method saw the largest increase in speed at 72%. The Step method saw a 60% increase. These results tend to support the fact that the linear accelerator routine appears to perform best when calculating problems requiring a large number of iterations. This tendency has been seen on several occasions in this chapter (i.e. table 4.6 and table 4.5). A non-

| **Method** | **K** | **Iterations to Convergence** | **Accelerated Iterations to Convergence** | **Relative % Increase in Speed** |
|---|---|---|---|---|
| **DD** | 1.17959 | 914 | 256 | 72.0 |
| **Step** | 1.08552 | 485 | 198 | 59.2 |
| **LC** | 1.17982 | NA | 138 | NA |

**Table 4.7: OUT-IN Core Eigenvalue Comparison**

accelerated run was not performed for LC because the accelerated run time at 138 iterations took 17,787 seconds or about 5 hours. If the number of unaccelerated iterations required to convergence is close to DD, the LC method would take about 33 hours to run. The DD run took approximately 338 seconds accelerated and 1222 seconds unaccelerated. Therefore, since the two methods were close in agreement with each other, the DD method would be used instead of the LC method in order to produce results in less time. The accuracy of LC is still considered to be better than DD; however, at the run times shown so far, full reactor problems will take hours to run under the LC method. The LC method will be better utilized for reactor type problems where the fuel contains neutron absorbers and DD requires a very large spatial mesh size to maintain stability.

Figure 4.20 shows the average power per assembly for the OUT-IN core configuration using the LC method. The power map corresponds directly with the assembly fuel locations shown in figure 4.19. The DD method reproduced virtually the same numbers with agreement between the first two digits of LC for every power number

| | | | | | |
|---|---|---|---|---|---|
| 0.8606 | 0.8666 | 0.7747 | 0.4228 | | |
| 1.5296 | 1.6271 | 1.7135 | 1.3894 | 0.5368 | |
| 0.8516 | 1.0793 | 1.9046 | 1.9756 | 1.3890 | 0.4226 |
| 0.4690 | 0.6478 | 1.0683 | 1.9039 | 1.7124 | 0.7741 |
| 0.2822 | 0.3895 | 0.6475 | 1.0784 | 1.6254 | 0.8656 |
| 0.2043 | 0.2821 | 0.4686 | 0.8505 | 1.5274 | 0.8593 |

**Figure 4.20: OUT-IN Average Power per Assembly Map using LC**

listed in the map. This map shows that the peak to average power ratio falls in the second batch of fuel very close to the edge of the core. The max power ratio is 1.9756. The center of the core shows reduced power levels. In fact, the center is severely depressed with only about 25% of the average power being produced in these assemblies. From a fuel management standpoint, this configuration is highly lopsided with most of the power in the core being produced along the outer edge of the core. The wide spread in the NP numbers from 0.2043 out to 1.9756 (difference of 1.7713) also show indications that the power profile for this core is not very flat. Therefore, according to fuel management principles, the core should be rearranged to yield a more even burn up rate for the fuel across the entire core.

The lopsided power distribution is accentuated in figure 4.21 which shows a two

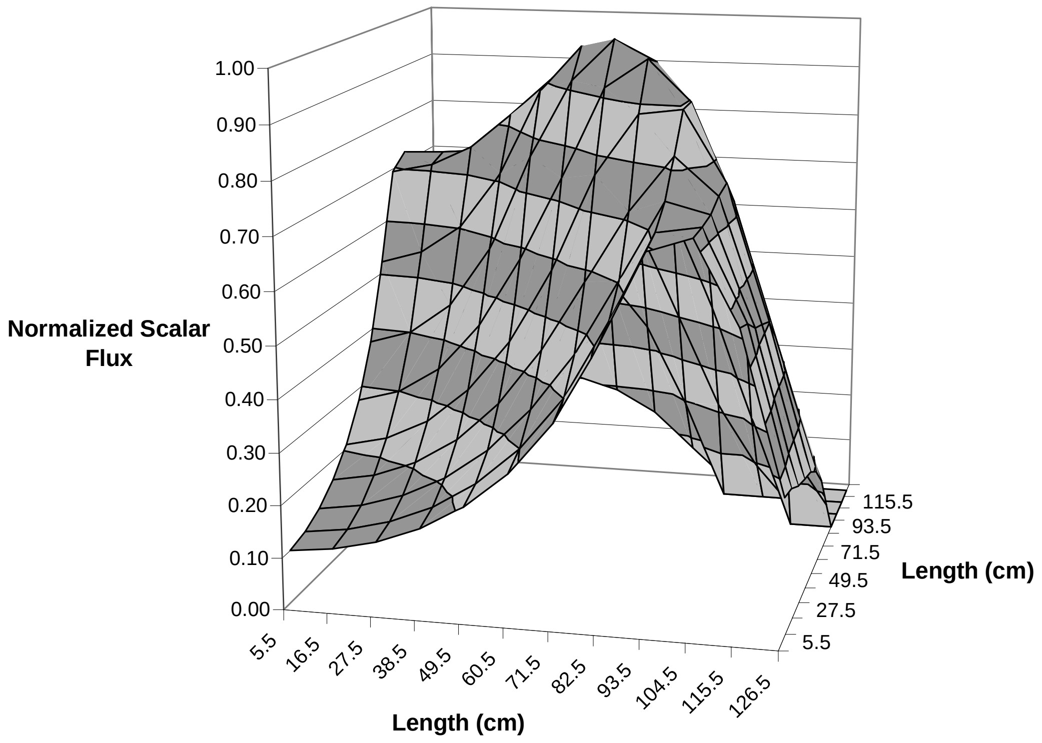

**Figure 4.21: OUT-IN Quarter Core Flux Profile using LC**

dimensional surface plot of the flux profile across the core. Since the power is proportional to the flux, figure 4.20 does provide a good visual representation of the power profile.  The plot shows a quarter core profile in the same orientation as figures 4.19 and 4.20. The profile shows the flux depression in the center of the core and the peak flux formation along the edge of the core. The peak is directly consistent with the location of a batch of high-enriched fuel along the back corner of the core where the "curved" (actually jagged) surface is located.  The next section will discuss a more elegant fuel arrangement that reduces the peaks and troughs associated with the OUT-IN fuel pattern.

**The Low-Leakage Fuel Loading Pattern**

The second and final fuel-loading pattern demonstrated with 3D-TRAN is called the low leakage core-loading scheme. This fuel-loading pattern is designed based on X batches of fuel as with the OUT-IN fuel loading scheme. For the test case, the fuel consists of the same three batches of fuel consistent with the previous OUT-IN core loading.  The core configuration is modified to include a checkerboard fuel arrangement inside the reactor core. This configuration places higher w/o fuel next to lower w/o fuel as demonstrated with the two-region assembly problem earlier in this chapter. The checkerboard pattern, as with the OUT-IN configuration, places the highest w/o fuel along the periphery of the core. The low leakage core modifies the checkerboard pattern by moving the highest w/o fuel off the core perimeter and placing it approximately one row inside the edge. The outer ring of assemblies is replaced with lower w/o fuel to reduce the number of neutrons produced along the edge of the core thereby reducing the neutron leakage out the sides of the core.

This test problem uses the same spatial meshing, assembly dimensions, and core dimensions as the OUT-IN test. The same cross sections for the 2 w/o, 3 w/o, and 4 w/o fuel are also used. The DD spatial quadrature was used to calculate the core eigenvalue. The Step and LC methods were not implemented for this problem. This test problem is merely a demonstration of fuel management techniques and will be used solely to show how the 3D-TRAN code can be used to aid the student in designing reactor core fuel loading patterns. The comparison of Step, LC, and DD made with the OUT-IN core showed that LC and DD were virtually identical in eigenvalue and power profiles. Therefore, the DD method was chosen primarily based on its calculation speed and its proven agreement in accuracy with LC.

For this test problem, the fuel in the OUT-IN core configuration, given in figure 4.19, was moved around in the core to yield the new core configuration of figure 4.22. The new configuration was picked through a trial and error process of picking a core

| | | | | | |
|---|---|---|---|---|---|
| 1 | 2 | 1 | 2 | | |
| 3 | 3 | 3 | 3 | 1 | |
| 2 | 3 | 2 | 1 | 3 | 2 |
| 1 | 1 | 2 | 2 | 3 | 1 |
| 2 | 1 | 1 | 3 | 3 | 2 |
| 1 | 2 | 1 | 2 | 3 | 1 |

**1** = 2 w/o
**2** = 3 w/o
**3** = 4 w/o

**Figure 4.22: Quarter Core Low Leakage Fuel Loading Configuration**

configuration and running the 3D-TRAN code to identify peak power regions. The power peaks were reduced by transplanting the higher-powered assemblies to regions of the core where the power was low. The resulting eigenvalue for this configuration was 1.1873. This is only a difference of 0.0077 or a 0.7% increase from the DD predicted eigenvalue of table 4.7 for the OUT-IN core loading; however, the core power profile is completely different.

Figure 4.23 shows the average power per assembly for the low leakage core. This profile shows that the peak to average power ratio dropped from 1.9756 to 1.6429. The location of the peak also shifted inward by one assembly when compared to figure 4.20. The flux profile in the center of the core is also greater than the average power, which is a significant improvement over the OUT-IN core-loading scheme where the power is severely depressed below the average power. The assemblies along the perimeter of the core are also producing less power than the OUT-IN power configuration of figure 4.20. The neutron flux is reduced thereby producing a lower leakage core than the OUT-IN scheme. In general, the low leakage fuel-loading scheme produces a better core wide power shape and will consume fuel more efficiently than the OUT-IN loading scheme.

| | | | | | |
|---|---|---|---|---|---|
| 0.3989 | 0.4951 | 0.3237 | 0.2108 | | |
| 1.2911 | 1.2252 | 1.0448 | 0.6587 | 0.1907 | |
| 1.6134 | 1.6098 | 1.5291 | 0.9279 | 0.6581 | 0.2105 |
| 1.1583 | 1.1534 | 1.6429 | 1.5277 | 1.0432 | 0.3230 |
| 1.3787 | 1.0542 | 1.1522 | 1.6069 | 1.2224 | 0.4938 |
| 1.0274 | 1.3776 | 1.1561 | 1.6094 | 1.2875 | 0.3977 |

**Figure 4.23: Low-Leakage Average Power per Assembly Map using DD**

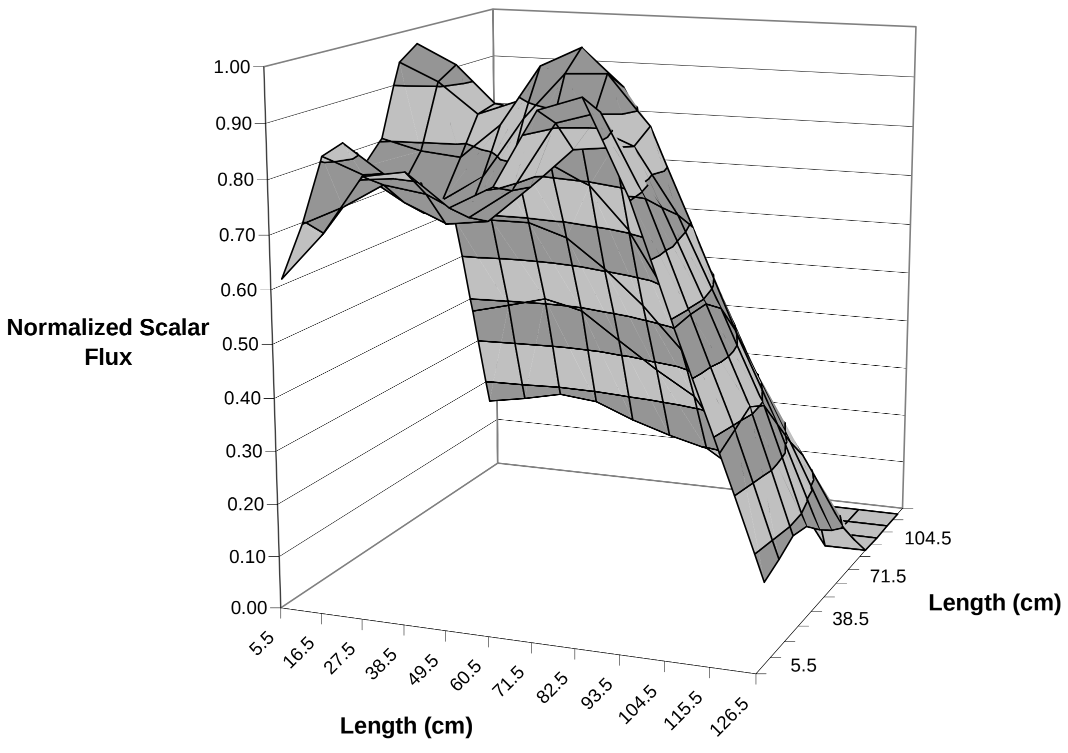

**Figure 4.24: Low Leakage Quarter Core Flux Profile using DD**

As a final note, the neutron flux profile for the low leakage core is given in figure 4.24. The mixing of fuel assemblies inside the core produces a more jagged flux profile than the OUT-IN loading pattern of figure 4.21. The flux depression at the center of the core is caused by the placement of the lowest w/o fuel surrounded by higher w/o fuel assemblies at the center. The end result is that the flux profile is raised in the center of the core and the outer edge power peak is reduced when compared to figure 4.21.

## Concluding Remarks about 3D-TRAN

The results presented in this chapter show 3D-TRAN's versatility as a heterogeneous reactor code. The numerical transport comparison to analytical diffusion proved that 3D-TRAN can indeed model a homogenous reactor with flux predictions comparing almost exactly to those of the analytical diffusion solutions (in the limit as the spatial meshes are refined). The heterogeneous fuel pin and two-region assembly tests showed 3D-TRAN's ability to correctly initialize a four energy group, multi-region reactor problem as well as show the source-sink relationship between different w/o fuel assemblies. The heterogeneous tests were expanded to full core models and very successfully modeled the fuel management practices outlined in Cochran (1990). The low leakage core configuration did indeed produce flatter power profiles than the OUT-IN core configuration as described in the text (Cochran, 1990).

The results also showed that the successful use of DD as a spatial quadrature method depends purely on the mesh size and material used. The LC method is considered the more accurate method over coarser mesh sizes (based on Chapter 3 results), especially with absorbing materials; however, the use of the LC method as a model for full core fuel loading schemes is currently hampered by its extensive calculation time. In the heterogeneous reactor test cases and the spatial meshes used thus far, the DD method and LC method predict virtually the same answers making DD the more viable method for producing full core power profiles because of its speedy calculation time. The only problem is that heterogeneous cores with high absorbers have not been explored and will more than likely cause DD to produce erroneous results over less refined spatial meshes. These errors must be carefully scrutinized for when using 3D-TRAN to model any reactor based test problem.

# Chapter V: Conclusions and Recommendations

## Conclusions about 3D-TRAN code implementation

The 3D-TRAN code was designed to numerically model neutron transport theory for a three dimensional spatial problem. The code uses three different types of spatial quadratures that model the left hand side of the Boltzmann Transport Equation (BTE). A careful comparison of these quadratures showed that the Linear Characteristic (LC) quadrature produced results that were more accurate than Diamond Difference (DD), or Step. The error of the LC method tends to be approximately a factor of 10 lower than DD and a factor of 100 lower than Step. This fact means that the LC method produces more accurate answers over the same size spatial mesh when compared to either DD or Step. This result suggests that a less refined spatial mesh can be used to model larger dimensioned problems without the hassle of computer "out of memory" errors. DD and Step's requirement for more refined spatial meshes make these two quadratures more susceptible to computer memory limitations.

The heterogeneous uniform source results revealed that the LC method tended to produce more stable results over material region boundaries than the DD method. The DD method requires more refined spatial meshes near material boundaries especially when the optical thickness ratio becomes larger than 4. Without the refinement, the DD method calculated negative flux results that oscillated negative and positive about zero. These results are numerically induced and do not correctly model the particle interaction and attenuation in higher absorbing media. The LC method retained stability through an optical thickness ratio of 10 or less. This means that the LC method required less mesh refinement near material boundaries to produce stable accurate answers. The Step method also performed well through material region boundaries, but tended to over predict the flux profiles. This result is directly contributed to Step's assumption of smearing the source as a constant function within a computational cell. Step's inherently linear order of error reduction means that Step required more refined meshes to produce answers as accurate as DD or LC.

The cost of running each of the three spatial quadratures is highly different. The DD and Step methods required relatively little time to compute one phase space cell (2.7E-6 seconds). This means that larger more intensive spatial, angular, and energy mesh problems will calculate quickly as long as memory requirements stay well within the limitations of the computer. These fast computation times are directly linked to the small number of equations required to model the left hand side of the BTE (only about 4 or 5 simple equations per method). The LC method was quite the opposite. The time required to calculate one phase space cell was about a factor of 60 times longer than DD or Step. The computation time is directly linked to the large number of linear transformations, cell splitting, and linear algebra manipulations required to generate one cell answer. Larger meshing problems took substantially longer than DD or Step to calculate an answer. The trade-off of longer calculation time is that the LC method is more accurate than DD or Step over the same spatial mesh size. This accuracy is accentuated when comparing results over multi region problems where absorbing materials are used. The end results were that LC produced more accurate answers over higher absorbing media than DD or Step; however, the computation costs are much higher than the other two methods.

The addition of eigenvalue calculations to the neutron transport equation gave the 3D-TRAN code the ability to model isotropic fission sources. The results for the homogenous reactors show that the 3D-TRAN code accurately modeled the analytical diffusion equation solutions for the parallelepiped and right circular cylindrical reactors. The results of the cylindrical reactor were compared to the work of Harman (2001) and were shown to accurately calculate diffusion-based eigenvalue solutions. These results verified the numerical solving routines used in the neutron transport code.

The addition of eigenvalue calculations to the transport code also introduced the threat of high computation times for generating eigenvalue solutions. These times are caused by the slow convergence behavior of the power iteration method. Decreasing the time to convergence for these calculations is vital to making 3D-TRAN a viable student teaching tool. The use of a modified linear extrapolation/shooting method showed increases to computation speed on the order of about 60%. The method also performed better (approximately a 65% – 70% increase) for problems that required a large number of unaccelerated iterations (on the order of thousands). The convergence acceleration method did not work well for problems that converged quickly and only produced about a 10% increase in speed. Initial results for the accelerator were optimized over homogenous fission reactor problems, but showed consistently good results for heterogeneous problems (on the order of a 60% increase in speed also). The accelerator still needs to be verified over heterogeneous problems and is left as a task for future work on this project.

The homogenous reactor models of 3D-TRAN were expanded to include heterogeneous problems. The fuel pin calculation over a unit fuel cell showed that cross-section homogenization is possible and can be used to define larger similar composition fuel assemblies. The two-region assembly problem demonstrated the source-sink relationship between different weight percent fuels and reproduced the correct reactor physics behavior for such a configuration.  Multiple assemblies can be linked together to model realistic commercial reactor fuel configurations where the student can easily manipulate material input parameters to practice fuel management techniques. Example reactor configurations demonstrated the fundamental concepts of fuel management outlined in Cochran (1990).

All these results indicate that the 3D-TRAN code is properly modeling the reactor physics behavior of an isotropic fission reactor. The addition of benchmarking the spatial quadratures against a uniform source also gives 3D-TRAN the ability to model simple neutron radiation shielding problems where an isotropic source is known. This versatility in design allows 3D-TRAN to be used for a wide range of student problems that can serve as an aid to the teaching and understanding of neutron particle physics.

## Conclusions about 3D-TRAN as a teaching tool

The main goal of this research was to develop a reactor numerical model that accurately modeled neutron particle physics in a reactor based environment. The two methods of choice were the diffusion theory and the neutron transport theory. Since diffusion theory is a simplification to neutron transport, the neutron transport method was pursued for its versatility in modeling heterogeneous regions as well as its capability to model the diffusion theory subset of problems. This ability provides a more rounded tool for students to use in modeling reactors of simple or complex geometries. Therefore, the main goal of this research has been accomplished.

The 3D-TRAN code was designed for multiple uses by the student. The code has the capability to model simple diffusion based reactor geometries such as the parallelepiped and the circular cylinder. Each reactor can be changed from homogenous to heterogeneous by correctly assigning material cross sections to the appropriate regions. More complex heterogeneous reactor material initialization routines were written to allow for easy problem designs by the student. These routines include fuel pin geometries, fuel assemblies and commercial reactor type geometries. The use of region input maps similar to those shown in Chapter 4 will allow the student to easily change core configurations by simple rearranging of the core input fuel assembly map. This type of input is similar to many commercial reactor codes available to the nuclear power industry today.

Finally, the output of the code was designed to print out important data such as flux profiles by energy group and reactor core average power per assembly maps. This output data is crucial for the student to learn how reactors operate and how fuel management techniques are integrally linked to the neutron particle interactions in the reactor core.

## Recommendations for Future Work

As with all research, there are always new possibilities and ideas to help expand or improve on current research. The following few paragraphs outline some suggestions to help further the numerical reactor model developed in this research.

The input and output of the 3D-TRAN code can easily be integrated into a windows based environment using techniques similar to those presented by Harman (2001). This final push in code development should help make the 3D-TRAN code more user friendly and extremely easy for the student to design test problems.

The linear convergence accelerator should be optimized and/or analyzed for overall performance over heterogeneous problems.  The range of problems should cover highly absorbing regions verses lower absorbing regions as well as multiple regions of varying types. Hopefully the convergence accelerator can be designed smartly and have the ability to choose over-relaxation parameters or shooting iteration numbers that best match the type of problem being solved. There are also a number of other convergence accelerators that may work better than the linear extrapolation model used in this research.  Some of these methods are the coarse mesh rebalance method, polynomial interpolation and extrapolation techniques, and synthetic diffusion acceleration.

A material cross section generation routine needs to be added to calculate proper cross section data as input to the code. The lack of a generator on 3D-TRAN forces the student to hand input cross sections from another source, either referenced from a book or another code.  Material cross section data files can be developed as reference libraries for the code. A set of user defined inputs such as temperature, and energy profiles should be added so the code can properly calculate material cross sections for problems of interest. This feature will also aid problem design when time dependence is added to the program since material cross sections will have to be calculated at each new time step depending on the reactor kinetics and thermal hydraulic properties of the system.

Perturbation theory should be added to help determine criticality of the system. The main thrust here should be to have the student design a reactor core configuration with an eigenvalue that is close to, but greater than one. The student can then perturb the cross sections of the fuel uniformly across the core by adding increasing soluble boron concentrations in the moderator until an eigenvalue of one is achieved[1]. The reason the core should be designed greater than one is so that the reactor core retains enough fuel to remain critical over a defined fuel cycle period. If the reactor is initially designed at a criticality value of one without soluble boron, the core will not operate very long before becoming sub critical. Therefore, initial super criticality will determine the lifetime of the core and will allow the student to vary boron concentration levels to maintain an eigenvalue of one during the fuel cycle. At the end of the cycle, the core should be critical without any boron present in the core.

The variation of the core fuel composition over the fuel cycle will require the addition of depletion calculations and the addition of time dependence to the transport code. Time dependence also introduces the ability to input time dependent material cross sections that can be designed to respond to temperature, pressure, and flow variations caused by the reactor thermal hydraulic system. A direct coupling of both reactor physics and thermal hydraulics introduces the concepts of reactor kinetics and delayed neutrons into the code. All these modifications will require time and patience to implement.

The final improvement is to look at increasing the overall speed by optimizing the code as well as enlisting the help of convergence accelerators. A properly optimized code and data structure will help increase overall computational efficiency. This effort will be required if time dependence is added to the code. Another factor that will aid the codes

[1] This method applies to pressurized water reactors only.

computational speed is the type of computer system. Computers are becoming faster and available computer memory is growing. In a few years, it is estimated that the 3D-TRAN will easily be able to run large test problems on a desktop computer efficiently and within a reasonable time frame (minutes vs. hours). All these improvements will aid and expand the development of 3D-TRAN as a teaching tool for future use in reactor studies.

# APPENDIX A: Angular Quadrature

The Discrete Ordinates quadrature sets are defined based on a unit sphere with unit direction vector $\hat{\Omega}$ residing within the sphere in a given direction. Two angular coordinates (θ, ω) determine the direction of $\hat{\Omega}$. These angular variables are generally defined relative to a set of orthogonal spatial coordinates (Lewis and Miller, 1993). Figure A1 below shows the general orientation of the variables to the axes in Cartesian coordinates.

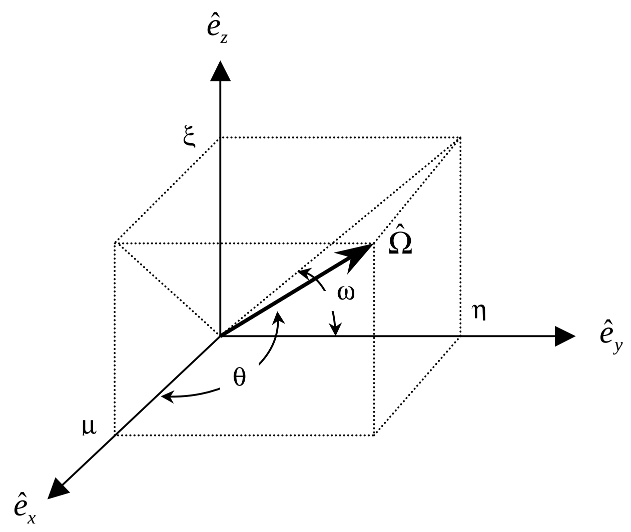

**Figure A1: Angular Directional Variables**

The direction cosines (μ, η, ξ) are linked geometrically by the following relations:

$$\begin{aligned}\mu &= \cos\,\theta \\ \eta &= \sqrt{1-\mu^2}\,\cos\,\omega \\ \xi &= \sqrt{1-\mu^2}\,\sin\,\omega\end{aligned} \tag{A.1}$$

Only two of the direction cosines can be specified independently (Lewis and Miller, 1993); therefore, they must satisfy the relation:

$$\mu^2 + \eta^2 + \xi^2 = 1. \tag{A.2}$$

In three-dimensional space, the unit directional vector $\hat{\Omega}$ is specified by a dependent set of direction cosines ($\mu$, $\eta$, $\xi$) that form one directional ordinate, n. Therefore, the Boltzmann Transport Equation (BTE) is discretized over specific directional ordinates to approximate the continuous space that neutral particles can flow. The BTE is then solved for specific angles as follows:

$$[\hat{\Omega}_n \cdot \vec{\nabla} + \sigma(\vec{r}, E)]\Psi(\vec{r}, \hat{\Omega}_n, E) = S(\vec{r}, \hat{\Omega}_n, E). \tag{A.3}$$

The final step after solving for equation (A.3) over each angle is to use the numerical quadrature rule of equation (2.4) to solve for the scalar flux. The quadrature rule uses a Gaussian Quadrature technique (Atkinson, 1989) to approximate the angular integration of the BTE. The general Guassian quadrature formula representation is:

$$\int_{-1}^{1} f(x)\, dx \cong \sum_{i}^{n} w_i f(x_i). \tag{A.4}$$

This equation is simple to solve numerically, but requires a careful derivation of the weights ($w_i$) to approximate the continuous integration on the left hand side of equation (A.4).

For general applications in solving the BTE, a level symmetric quadrature set can be used to solve many different kinds of problems (Lewis and Miller, 1993) (Carlson 1971). "A level symmetric quadrature ($S_N$) uses the same set of N/2 positive values of the direction cosines with respect to each of the three axes" (Lewis and Miller, 1993). This statement means that a set of direction ordinates defined over the first octant of the unit sphere can be symmetrically reflected about any of three orthogonal axes to generate the remaining seven octants of three-dimensional phase space and still maintain the same equal distribution (or weight) of ordinates across all eight octants. All axes are therefore weighted the same and the unit sphere becomes tiled in equal or non-equal weighted directional ordinates. For example, figure A2 shows the $S_2$ ordinate set in the positive octant of the sphere. The ordinate is equally spaced between the three orthogonal axes with $\mu = \eta = \xi = 0.57735$ (from equation (A.2)). The reflection of this ordinate about the three orthogonal axes yields the other seven ordinates in the set with each weight equaling 1/8 or in general,

$$\sum_{n=1}^{N(N+2)/8} w_n = 1 \tag{A.5}$$

for any level symmetric ordinate set (Lewis and Miller, 1993). A detailed description of level symmetry is given in Lewis and Miller (1993) pages 158-165 with some more descriptive figures referencing the key points mentioned here.

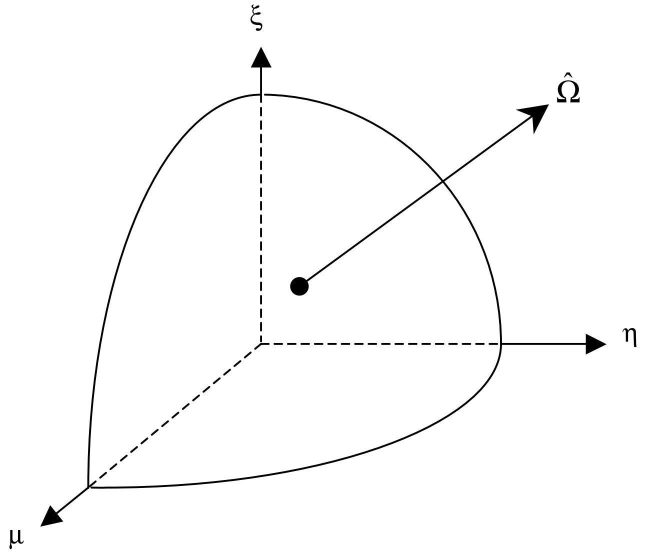

**Figure A2: Level Symmetric $S_2$ Ordinate Set in the Positive Octant**

# APPENDIX B: Transport Boundary Conditions

The Boltzmann Transport Equation requires the use of boundary conditions along the outside surface of the problem domain in order to generate a solution. These boundary conditions can be reflective, albedo, vacuum, or constant flux to name a few. The 3D-TRAN code developed in this research uses three boundary conditions: full reflective, albedo reflective, and vacuum. These boundary conditions are of the following form:

$$\Psi(\bar{r},\hat{\Omega},E) = \alpha(E)\ \Psi(\bar{r},\hat{\Omega}',E), \quad \text{for } \hat{\Omega}\cdot\hat{n} < 0 \tag{B.1}$$

where $\hat{n}$ is the outward unit normal vector to the boundary surface, $\bar{r}$ resides within the problem boundaries, and $\alpha(E)$ is a known isotropic albedo. For vacuum boundary conditions, $\alpha(E)$ equals zero and for full reflection, $\alpha(E)$ equals one. Any fraction between zero and one constitutes a reflective isotropic albedo (or a portion of the outgoing flux returns into the problem along a reflection ray incident to the boundary surface). The $\hat{\Omega}'$ term represents the directional ordinate that is the reflection of $\Omega$, the incident direction ordinate. This reflection is easy to accomplish if a level symmetric quadrature set is used. For example, a reflective boundary means that the angular flux for a given direction at the boundary cell is passed into the reflected direction ordinate angular array for that cell. In a level symmetric quadrature, there is always an even number of ordinates with each ordinate being the reflection of one of the other ordinates in the set. This property enables a computer code to build reflection arrays during initialization that map the reflected pairs together and identify their ordinate numbers in the set. This makes reflection boundary conditions very easy to implement in transport problems.

Actual implementation of the boundary conditions can be somewhat of an art form. During the numerical walk through the spatial mesh in x, y, and z directions, the boundaries must be taken into account numerically at some time during the calculation. This can be handled one of several ways. First, a directional ordinate is picked. The mesh walk then begins by following the general direction of the ordinate vector. As a boundary is approached, the boundary condition can be applied at the end of each row calculated. For instance, say the mesh walk begins by picking a y value and calculating across the row in terms of x. When the maximum x value is reached, the boundary condition can be applied. This is repeated every time the maximum value is reached in the row. If a parallelepiped problem domain exists, computational speed can be obtained by walking through the entire mesh, then calculating all boundary conditions at the end of the spatial walk by treating the boundary conditions along an entire plane. This technique is easy to implement in FORTAN 95 using array notation (see Ellis, 1994) for the angular flux.

On the other hand, if we intend to handle a circle with vacuum boundary conditions using a Cartesian (x,y,z) mesh (such as our circular reactor geometry), the boundary conditions are better implemented as they are approached during the calculation. This technique helps us deal with the non-uniform jagged edge of the curved boundary created by approximating the curved circular edge in the Cartesian mesh. Also, the wasted material array space[1] outside the circle can be initialized to zero, which gives

[1] Since we are using Cartesian coordinates and matrix arrays, a curved edge leaves part of a material array non usable because it resides outside the problem domain of interest. Therefore, larger arrays will be specified than required in order to approximate the curved region along the problem boundary. See figure B1.

the computer code a value to search for when walking through the material mesh. When the zero value is reached, the code knows it has crossed a boundary and is now outside of the problem domain. The code can make appropriate adjustments and implement the correct boundary condition at the curved edge. If one is careful, the code can be modified to adjust the spatial walk during the transport calculation to only calculate those spatial mesh cells that contain material (only those that lie within the problem domain). This technique is implemented in the 3D-TRAN code and improves overall calculation efficiency by as much as 22%[2] for a two or three dimensional problem. Figure B1 shows a diagram of a curved boundary represented as a jagged boundary in a Cartesian mesh. This configuration (figure B1) is the common shape of most commercial pressurized water reactors in service today and is modeled in the 3D-TRAN code.

[2] This efficiency is generated by looking at the area/volume of the cells not contained within the circular/cylindrical boundary domain and then comparing it with the area/volume of the total number of cells in the array.

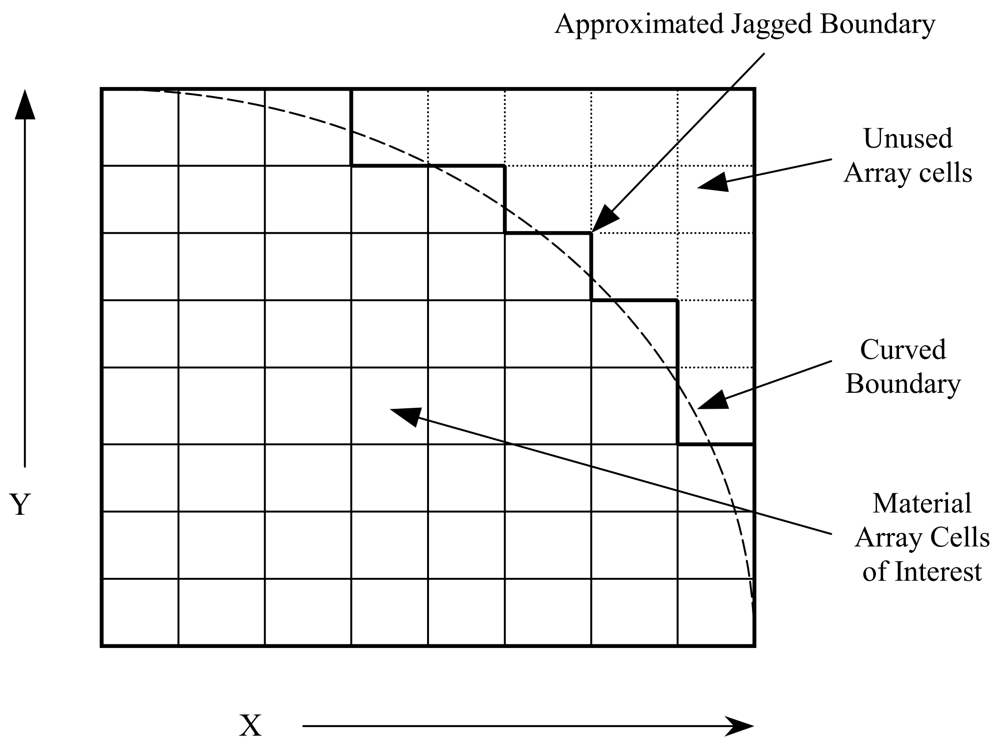
Approximated Jagged Boundary
Unused Array cells
Curved Boundary
Material Array Cells of Interest
Y
X

**Figure B1: Two Dimensional (x, y) Curved Boundary Edge Representation**

# APPENDIX C: Derivation of Linear Characteristic Sub Cell Equations

The Linear Characteristic Method assumes a linear source distribution of the form:

$$S(u,v,w) = \mathrm{A} + \mathrm{B}_{\mathrm{u}}u + \mathrm{B}_{\mathrm{v}}v + \mathrm{B}_{\mathrm{w}}w \tag{C.1}$$

The coordinates of the (x, y, z) unit cell mesh of figure 2.4 are transformed via a Jacobian transformation matrix to a unit mesh cell with each side of the cube equal to one. The transformed coordinate system changes from (x, y, z) to (*u, v, w*)[1]. We now define a cell moment operator to simplify the volumetric zeroth and first moment derivations. This unitary operator is as follows:

$$M[g(\vec{R})] = \frac{1}{V}\iiint_{cell} g(\vec{R})\; d^3\vec{R} \tag{C.2}$$

where *V* equals the cell volume and g is a function in (*u, v, w*) space. A face operator similar to equation (C.2) is also defined:

$$M^f[g(\vec{R})] = \frac{1}{A^f}\iint_{face} g(\vec{R})\; d^2\vec{R} \tag{C.3}$$

where $A^f$ equals the area of the inflow/outflow cell face space. Now, the linear characteristic equations can be derived for each of the three sub unit volumes: the tetrahedron, the prism, and the parallelepiped.

[1] The Jacobian transformation method is summarized in the paper authored by Mathews, Miller, and Brennan (2000). A summary of the technique pertaining to this research is given in Appendix D.

Tetrahedron

The tetrahedron equations have already been derived (see Mathews, Miller, and Brennan, 2000) for a unit tetrahedron. For clarity, the equations are reproduced from the paper in the following paragraphs.

First, the unit tetrahedron is defined as in figure C1. This figure shows the orientation of the unit tetrahedron in relation to the streaming direction $\hat{\Omega}$. The *w* axis is aligned with the streaming direction. The unit vectors $\vec{E}_1$, $\vec{E}_2$, and $\vec{E}_3$ form the orthogonal Jacobian basis vectors of the tetrahedron. Each of the six arbitrary tetrahedron in figure 2.4 is transformed into the unit tetrahedron of figure C1 by defining these basis vectors; therefore, the transport equations are only derived once per unit sub volume.

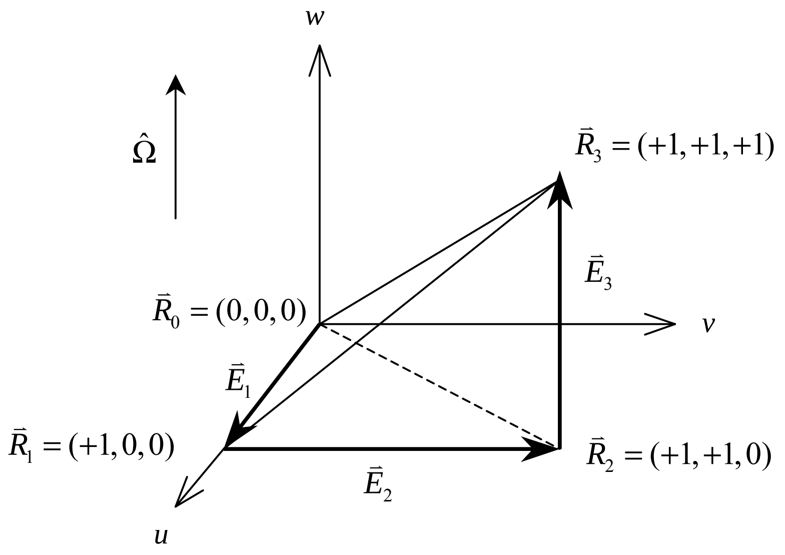

**Figure C1: The Unit Tetrahedron**

The balance equations for the tetrahedron are derived by using the moment operator equation:

$$M[g(u,v,w)] = 6\int_0^1 du \int_0^u dv \int_0^v dw\, g(u,v,w). \qquad \text{(C.4)}$$

Applying equation (C.4) on the BTE with the following representation of the streaming operator:

$$\hat{\Omega}\square\vec{\nabla}\Psi(u,v,w)=\frac{\partial}{\partial w}\Psi(u,v,w)\,, \tag{C.5}$$

we get for the zeroth balance equation:

$$3\Psi_{\mathrm{A}}^{\mathrm{out}}-3\Psi_{\mathrm{A}}^{\mathrm{in}}+\sigma\Psi_{\mathrm{A}}=l\ S_{\mathrm{A}} \tag{C.6}$$

where *l* is the length of the splitting line made by $\hat{\Omega}$ through the global mesh cell, σ is the material total cross section in the sub cell, and the average or zeroth flux across the inflow face, outflow face and within the cell volume make up the right hand side of the equation. The first moments are found by taking the operator of *M*[(*u, v, or w*) BTE] yielding the first moment balance equation for (*u, v, w*) :

$$3\Psi_{\mathrm{u}}^{\mathrm{out}}-3\Psi_{\mathrm{u}}^{\mathrm{in}}+\sigma\Psi_{\mathrm{u}}=l\ S_{\mathrm{u}} \tag{C.7}$$

$$3\Psi_{\mathrm{v}}^{\mathrm{out}}-3\Psi_{\mathrm{v}}^{\mathrm{in}}+\sigma\Psi_{\mathrm{v}}=l\ S_{\mathrm{v}} \tag{C.8}$$

$$3\Psi_{\mathrm{v}}^{\mathrm{out}}-3\Psi_{\mathrm{A}}+\sigma\Psi_{\mathrm{w}}=l\ S_{\mathrm{w}}\ . \tag{C.9}$$

These equations are used to check for sub cell neutron balance and verify the linear characteristic equations that are derived below.

The source distribution of equation (C.1) substituted into the moment operator equation (C.4) to generate the zeroth and first moment equations of the source. The equations are as follows:

$$S_{\mathrm{A}}=M[S(u,v,w)]=A+\frac{3}{4}B_u+\frac{1}{2}B_v+\frac{1}{4}B_w \tag{C.10}$$

$$S_{\mathrm{u}}=M[u\ S(u,v,w)]=\frac{3}{4}A+\frac{3}{5}B_u+\frac{2}{5}B_v+\frac{1}{5}B_w \tag{C.11}$$

$$S_{\mathrm{v}} = M[v\ S(u,v,w)] = \frac{1}{2}A + \frac{2}{5}B_u + \frac{3}{10}B_v + \frac{3}{20}B_w \qquad \text{(C.12)}$$

$$S_{\mathrm{w}} = M[w\ S(u,v,w)] = \frac{1}{4}A + \frac{1}{5}B_u + \frac{3}{20}B_v + \frac{1}{10}B_w \qquad \text{(C.13)}$$

where the volume multiplication factor, V equals 1/6 for the unit tetrahedron, $S_A$ equals the zeroth source moment, and $S_u$, $S_v$, and $S_w$ equal the first moments of the source with respect to the given axes[2]. The coefficients are then found by solving equations (C.10) through (C.13) in terms of the source moments:

$$A = 16S_{\mathrm{A}} - 20S_{\mathrm{u}} \qquad \text{(C.14)}$$

$$B_u = -20S_{\mathrm{A}} + 40S_{\mathrm{u}} - 20S_v \qquad \text{(C.15)}$$

$$B_v = -20S_{\mathrm{u}} + 40S_{\mathrm{v}} - 20S_w \qquad \text{(C.16)}$$

$$B_w = -20S_{\mathrm{u}} + 40S_w \qquad \text{(C.17)}$$

The source distribution is now known within the cell because the source moments are either specified as initial conditions or are provided from the previous iteration.

The next step is to define the flux distribution across the inflow faces. This requires a face flux distribution function of the following linear form:

$$\Psi_{in}^{f}(u^f, v^f) = A^f + B_u^f u^f + B_v^f v^f \qquad \text{(C.18)}$$

where the super script *f* refers to the inflow face coordinate system of the master unit cube cell. This coordinate system should not be confused with the volume coordinate system shown in figure C1. A new face moment operator for the tetrahedron is defined as follows:

[2] The first moments derived in the sub volume cells are defined for corner moments. These moments need to be converted back to central moments when compiling the outflow face first moments. This is accomplished in order to maintain central values distributed about the center of the cell where the cell average flux resides.

$$M^f[g(u^f,v^f)] = 2\int_0^1 du^f \int_0^u dv^f \; g(u^f,v^f) \tag{C.19}$$

where $A^f$ is the area of the inflow face (1/2 for a unit triangle). By applying the face moment operator, the zeroth and first moments of the inflow face flux are found:

$$\Psi_{\mathrm{A}}^{\mathrm{in}} = M^f[\Psi_{in}^f(u^f,v^f)] = A^f + \frac{2}{3}B_u^f + \frac{1}{3}B_v^f \tag{C.20}$$

$$\Psi_{\mathrm{u}}^{\mathrm{in}} = M^f[u^f\Psi_{in}^f(u^f,v^f)] = \frac{2}{3}A^f + \frac{1}{2}B_u^f + \frac{1}{4}B_v^f \tag{C.21}$$

$$\Psi_{\mathrm{v}}^{\mathrm{in}} = M^f[v^f\Psi_{in}^f(u^f,v^f)] = \frac{1}{3}A^f + \frac{1}{4}B_u^f + \frac{1}{6}B_v^f \tag{C.22}$$

where $\Psi_{\mathrm{A}}^{\mathrm{in}}$ equals the zeroth moment across the inflow face, and $\Psi_u^{in}$ and $\Psi_v^{in}$ equal the inflow face first moments. These fluxes contribute to the neutron flux within the cell and are added to the cell source neutron flux. Now, the face coefficients are found by solving equations (C.20) through (C.22) for the flux moments:

$$A^f = 9\Psi_{\mathrm{A}}^{\mathrm{in}} - 12\Psi_{\mathrm{u}}^{\mathrm{in}} \tag{C.23}$$

$$B_u^f = -12\Psi_{\mathrm{A}}^{\mathrm{in}} + 24\Psi_{\mathrm{u}}^{\mathrm{in}} - 12\Psi_{\mathrm{v}}^{\mathrm{in}} \tag{C.24}$$

$$B_v^f = -12\Psi_{\mathrm{A}}^{\mathrm{in}} + 24\Psi_{\mathrm{v}}^{\mathrm{in}}. \tag{C.25}$$

The flux distribution across the inflow face is now known because the flux is either specified at a boundary, or the flux is known from the output flux of the preceding cell in the mesh. We now have all the required values to solve for the flux inside the cell.

For any point in (*u*,*v*,*w*), the flux inside the cell is given by the characteristic integral:

$$\Psi(u,v,w) = \Psi^{in}(u,v)\; e^{-\varepsilon w} + l\int_0^w dw'\; S(u,v,w)\; e^{-\varepsilon(w-w')} \tag{C.26}$$

where the integration is performed along the streaming direction $w'$, and $\varepsilon$ is the optical thickness along $w'$. The operators of equations (C.2) and (C.19) are applied to equation (C.26) for both the zeroth and first moments of the fluxes. A new function is used to compactly write the integrals that fall out of performing the integration on equation (C.26) for all moments. This function is called the *K* function and was developed by Miller (1998), Brennan (1996) and Mathews (2000). The general form of the *K* function is as follows:

$$K_{i_1,i_2,\ldots,i_m}(\varepsilon) = \int_0^1 dt_1 \int_0^{t_1} dt_2 \ldots \int_0^{t_{m-1}} dt_m t_1^{i_1} t_2^{i_2} \ldots t_m^{i_m} e^{-\varepsilon t_m} \qquad \text{(C.27)}$$

The volumetric moments of the sub cell flux can then be derived and defined in a compact form as follows:

$$\begin{aligned}\Psi_{\mathrm{A}} &= 6A^f K_{0,0,0}(\varepsilon) + 6B_u^f K_{1,0,0}(\varepsilon) + 6B_v^f K_{0,1,0}(\varepsilon) + 6Al\ K_{0,0,0,0}(\varepsilon) \\ &\quad + 6B_u l\ K_{1,0,0,0}(\varepsilon) + 6B_v l\ K_{0,1,0,0}(\varepsilon) + 6B_w l[\ K_{0,0,1,0}(\varepsilon) - K_{0,0,0,1}(\varepsilon)]\end{aligned} \qquad \text{(C.28)}$$

$$\begin{aligned}\Psi_{\mathrm{u}} &= 6A^f K_{1,0,0}(\varepsilon) + 6B_u^f K_{2,0,0}(\varepsilon) + 6B_v^f K_{1,1,0}(\varepsilon) + 6Al\ K_{1,0,0,0}(\varepsilon) \\ &\quad + 6B_u l\ K_{2,0,0,0}(\varepsilon) + 6B_v l\ K_{1,1,0,0}(\varepsilon) + 6B_w l[\ K_{1,0,1,0}(\varepsilon) - K_{1,0,0,1}(\varepsilon)]\end{aligned} \qquad \text{(C.29)}$$

$$\begin{aligned}\Psi_{\mathrm{v}} &= 6A^f K_{0,1,0}(\varepsilon) + 6B_u^f K_{1,1,0}(\varepsilon) + 6B_v^f K_{0,2,0}(\varepsilon) + 6Al\ K_{0,1,0,0}(\varepsilon) \\ &\quad + 6B_u l\ K_{1,1,0,0}(\varepsilon) + 6B_v l\ K_{0,2,0,0}(\varepsilon) + 6B_w l[\ K_{0,1,1,0}(\varepsilon) - K_{0,1,0,1}(\varepsilon)]\end{aligned} \qquad \text{(C.30)}$$

$$\begin{aligned}\Psi_{\mathrm{w}} &= 6A^f K_{0,0,1}(\varepsilon) + 6B_u^f K_{1,0,1}(\varepsilon) + 6B_v^f K_{0,1,1}(\varepsilon) + 6Al\ K_{0,0,1,0}(\varepsilon) \\ &\quad + 6B_u l\ K_{1,0,1,0}(\varepsilon) + 6B_v l\ K_{0,1,1,0}(\varepsilon) + 6B_w l[\ K_{0,0,2,0}(\varepsilon) - K_{0,0,1,1}(\varepsilon)]\end{aligned} \qquad \text{(C.31)}$$

where those terms multiplied by *l* represent the flux contribution from the source and the terms multiplied by the face coefficients represent the flux contributions from the inflow faces.

Finally, the outflow fluxes are calculated. The outflow face of the tetrahedron is the plane where *w* = *v*, so the characteristic integral for the outflow face becomes:

$$\Psi^{out}(u,v) = \Psi^{in}(u,v)\ e^{-\varepsilon v} + l\int_0^v dw'\ S(u,v,w)\ e^{-\varepsilon(v-w')} \tag{C.32}$$

The outflow flux moments then become:

$$\begin{aligned}\Psi_{\mathrm{A}}^{\mathrm{out}} &= 2A^f K_{0,0}(\varepsilon) + 2B_u^f K_{1,0}(\varepsilon) + 2B_v^f K_{0,1}(\varepsilon) + 2Al\ K_{0,0,0}(\varepsilon) \\ &\quad + 2B_u l\ K_{1,0,0}(\varepsilon) + 2(B_v + B_w) l\ K_{0,1,0}(\varepsilon) - 2B_w l\ K_{0,0,1}(\varepsilon)\end{aligned} \tag{C.33}$$

$$\begin{aligned}\Psi_{\mathrm{u}}^{\mathrm{out}} &= 2A^f K_{1,0}(\varepsilon) + 2B_u^f K_{2,0}(\varepsilon) + 2B_v^f K_{1,1}(\varepsilon) + 2Al\ K_{1,0,0}(\varepsilon) \\ &\quad + 2B_u l\ K_{2,0,0}(\varepsilon) + 2(B_v + B_w) l\ K_{1,1,0}(\varepsilon) - 2B_w l\ K_{1,0,1}(\varepsilon)\end{aligned} \tag{C.34}$$

$$\begin{aligned}\Psi_{\mathrm{v}}^{\mathrm{out}} &= 2A^f K_{0,1}(\varepsilon) + 2B_u^f K_{1,1}(\varepsilon) + 2B_v^f K_{0,2}(\varepsilon) + 2Al\ K_{0,1,0}(\varepsilon) \\ &\quad + 2B_u l\ K_{1,1,0}(\varepsilon) + 2(B_v + B_w) l\ K_{0,2,0}(\varepsilon) - 2B_w l\ K_{0,1,1}(\varepsilon)\end{aligned} \tag{C.35}$$

The paper by Mathew, Miller, and Brennan (2000) discusses numerical implementation of these equations. The use of recursion relations on the *K* functions allows for stable, accurate (to within 16 decimal places) (Mathews, Miller, and Brennan, 2000) evaluations of all the required integrals. By using a subroutine developed by Miller (1998), the *K* functions can be calculated quickly and efficiently by using series expansions and recursive relations. Therefore, no new developments were made to the *K* function routines other than those implemented by Miller (1998).

The following sections cover the prism and the parallelepiped. The derivation of the equations for each volume follows the same procedure as the tetrahedrons and is outlined in Suriano (2001); therefore, for compactness, only the equations and unit volume diagrams will be presented.

Prism

The prism sub cell equations were derived in conjunction with the work of Suriano (2001). The derivations follow the procedure outlined in the previous section; therefore, only the equations are provided for reference. Figure C2 illustrates the orientation of the unit prism and its basis vectors.

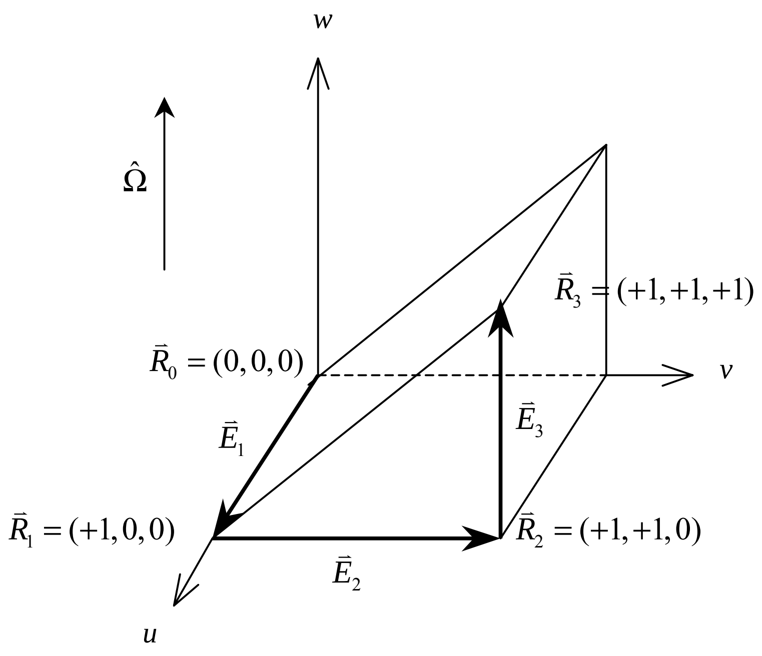

**Figure C2: The Unit Prism**

The volumetric moment operator for deriving the prism equations is as follows:

$$M[g(u,v,w)] = 2\int_0^1 du \int_0^1 dv \int_0^v dw\, g(u,v,w) \tag{C.36}$$

and the face moment operator is:

$$M^f[g(u^f,v^f)] = \int_0^1 du^f \int_0^1 dv^f \; g(u^f,v^f) \tag{C.37}$$

The balance equations for the prism are as follows:

Zeroth: $2\Psi_{\mathrm{A}}^{Out} - 2\Psi_{\mathrm{A}}^{in} + \sigma\Psi_{\mathrm{A}} = S_{\mathrm{A}}$ (C.38)

First $u$ moment: $2\Psi_{\mathrm{u}}^{Out} - 2\Psi_{\mathrm{u}}^{in} + \sigma\Psi_{\mathrm{u}} = S_{\mathrm{u}}$ (C.39)

First $v$ moment: $2\Psi_{\mathrm{v}}^{Out} - 2\Psi_{\mathrm{v}}^{in} + \sigma\Psi_{\mathrm{v}} = S_{\mathrm{v}}$ (C.40)

First $w$ moment: $2\Psi_{\mathrm{v}}^{Out} - 2\Psi_{\mathrm{A}} + \sigma\Psi_{\mathrm{w}} = S_{\mathrm{w}}$ (C.41)

The volumetric source coefficients are:

$$A = 12S_{\mathrm{A}} - 6S_u - 12S_v \tag{C.42}$$

$$B_u = -6S_{\mathrm{A}} + 12S_u \tag{C.43}$$

$$B_v = -12S_{\mathrm{A}} + 24S_u - 12S_w \tag{C.44}$$

$$B_w = -12S_v + 24S_w \tag{C.45}$$

The inflow face coefficients are:

$$A^f = 7\Psi_{\mathrm{A}}^{in} - 6\Psi_u^{in} - 6\Psi_v^{in} \tag{C.46}$$

$$B_u^f = -6\Psi_{\mathrm{A}}^{in} + 12\Psi_u^{in} \tag{C.47}$$

$$B_v^f = -6\Psi_{\mathrm{A}}^{in} + 12\Psi_v^{in} \tag{C.48}$$

The volumetric moments of the sub cell prism are:

$$\begin{aligned}\Psi_{\mathrm{A}} &= (2A^f + B_u^f)K_{0,0}(\varepsilon) + 2B_v^f K_{1,0}(\varepsilon) + l(2A + B_u)K_{0,0,0}(\varepsilon) \\ &+2l[B_v K_{1,0,0}(\varepsilon) + B_w K_{0,1,0}(\varepsilon) - B_w K_{0,0,1}(\varepsilon)]\end{aligned} \tag{C.49}$$

$$\begin{aligned}\Psi_u &= (A^f + \frac{2}{3}B_u^f)K_{0,0}(\varepsilon) + B_v^f K_{1,0}(\varepsilon) + l(A + \frac{2}{3}B_u)K_{0,0,0}(\varepsilon) \\ &+l[B_v K_{1,0,0}(\varepsilon) + B_w K_{0,1,0}(\varepsilon) - B_w K_{0,0,1}(\varepsilon)]\end{aligned} \tag{C.50}$$

$$\begin{aligned}\Psi_v &= (2A^f + B_u^f)K_{1,0}(\varepsilon) + 2B_v^f K_{2,0}(\varepsilon) + l(2A + B_u)K_{1,0,0}(\varepsilon) \\ &+2l[B_v K_{2,0,0}(\varepsilon) + B_w K_{1,1,0}(\varepsilon) - B_w K_{1,0,1}(\varepsilon)]\end{aligned} \tag{C.51}$$

$$\begin{aligned}\mathbf{\Psi}_w &= (2A^f + B_u^f)K_{0,1}(\varepsilon) + 2B_v^f K_{1,1}(\varepsilon) + l(2A + B_u)K_{0,1,0}(\varepsilon) \\ &+2l[B_v K_{1,1,0}(\varepsilon) + B_w K_{0,2,0}(\varepsilon) - B_w K_{0,1,1}(\varepsilon)]\end{aligned} \tag{C.52}$$

Finally, the outflow fluxes are:

$$\begin{aligned}\mathbf{\Psi}_{\text{A}}^{out} &= (A^f + \frac{1}{2}B_u^f)K_0(\varepsilon) + B_v^f K_1(\varepsilon) + l(A + \frac{1}{2}B_u)K_{0,0}(\varepsilon) \\ &+ l(B_v + B_w)K_{1,0}(\varepsilon) - l\ B_w K_{0,1}(\varepsilon)\end{aligned} \tag{C.53}$$

$$\begin{aligned}\mathbf{\Psi}_u^{out} &= (\frac{1}{2}A^f + \frac{1}{3}B_u^f)K_0(\varepsilon) + \frac{1}{2}B_v^f K_1(\varepsilon) + l(\frac{1}{2}A + \frac{1}{3}B_u)K_{0,0}(\varepsilon) \\ &+ l\frac{1}{2}(B_v + B_w)K_{1,0}(\varepsilon) - l\frac{1}{2}B_w K_{0,1}(\varepsilon)\end{aligned} \tag{C.54}$$

$$\begin{aligned}\mathbf{\Psi}_v^{out} &= (A^f + \frac{1}{2}B_u^f)K_1(\varepsilon) + B_v^f K_2(\varepsilon) + l(A + \frac{1}{2}B_u)K_{1,0}(\varepsilon) \\ &+ l(B_v + B_w)K_{2,0}(\varepsilon) - l\ B_w K_{1,1}(\varepsilon)\end{aligned} \tag{C.55}$$

Parallelepiped (Unit Cube)

The parallelepiped sub cell equations were also derived in conjunction with the work of Suriano (2001). The derivations follow the procedure outlined in the tetrahedron section; therefore, only the equations are provided for reference. Figure C3 illustrates the orientation of the unit parallelepiped and its basis vectors.

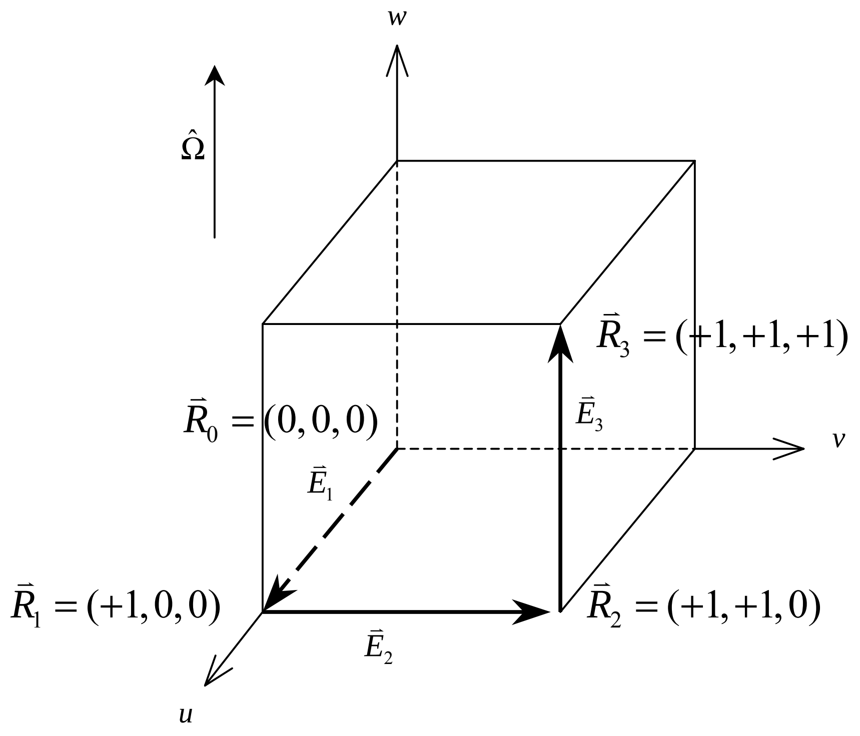

**Figure C3: The Unit Parallelepiped (Cube)**

The volumetric moment operator for deriving the prism equations is as follows:

$$M[g(u,v,w)] = \int_0^1 du \int_0^1 dv \int_0^1 dw \, g(u,v,w) \qquad \text{(C.56)}$$

and the face moment operator is:

$$M^f[g(u^f,v^f)] = \int_0^1 du^f \int_0^1 dv^f \; g(u^f,v^f) \qquad \text{(C.57)}$$

The balance equations for the prism are as follows:

Zeroth: $\Psi_{\mathrm{A}}^{out} - \Psi_{\mathrm{A}}^{in} + \sigma\Psi_{\mathrm{A}} = S_{\mathrm{A}}$ (C.58)

First $u$ moment: $\Psi_{u}^{out} - \Psi_{u}^{in} + \sigma\Psi_{u} = S_{u}$ (C.59)

First v moment: $\Psi_{v}^{out} - \Psi_{v}^{in} + \sigma\Psi_{v} = S_{v}$ (C.60)

First w moment: $\Psi_{\mathrm{A}}^{out} - \Psi_{\mathrm{A}} + \sigma\Psi_{w} = S_{w}$ (C.61)

The volumetric source coefficients are:

$$A = 10S_{\mathrm{A}} - 6S_u - 6S_v - 6S_w \tag{C.62}$$

$$B_u^f = -6S_{\mathrm{A}} + 12S_u \tag{C.63}$$

$$B_v^f = -6S_{\mathrm{A}} + 12S_v \tag{C.64}$$

$$B_w^f = -6S_{\mathrm{A}} + 12S_w \tag{C.65}$$

The inflow face coefficients are the same as derived for the prism, equations (C.46) through (C.48). The volumetric moments of the sub cell prism are:

$$\begin{aligned}\Psi_{\mathrm{A}} = {} & [A^f + \frac{1}{2}B_u^f + \frac{1}{2}B_v^f]\, M_0(\varepsilon) + l[A + \frac{1}{2}B_u + \frac{1}{2}B_v]K_{0,0}(\varepsilon) \\ & + l\, B_w[K_{1,0}(\varepsilon) - K_{0,1}(\varepsilon)]\end{aligned} \tag{C.66}$$

$$\begin{aligned}\Psi_{u} = {} & [\frac{1}{2}A^f + \frac{1}{3}B_u^f + \frac{1}{4}B_v^f]\, M_0(\varepsilon) + l[\frac{1}{2}A + \frac{1}{3}B_u + \frac{1}{4}B_v]K_{0,0}(\varepsilon) \\ & + l\frac{1}{2}B_w[K_{1,0}(\varepsilon) - K_{0,1}(\varepsilon)]\end{aligned} \tag{C.67}$$

$$\begin{aligned}\Psi_{v} = {} & [\frac{1}{2}A^f + \frac{1}{4}B_u^f + \frac{1}{3}B_v^f]\, M_0(\varepsilon) + l[\frac{1}{2}A + \frac{1}{4}B_u + \frac{1}{3}B_v]K_{0,0}(\varepsilon) \\ & + l\frac{1}{2}B_w[K_{1,0}(\varepsilon) - K_{0,1}(\varepsilon)]\end{aligned} \tag{C.68}$$

$$\begin{aligned}\Psi_{w} = {} & [A^f + \frac{1}{2}B_u^f + \frac{1}{2}B_v^f]\, [M_0(\varepsilon)(1\text{-}\rho(\varepsilon))] + l[A + \frac{1}{2}B_u + \frac{1}{2}B_v]K_{1,0}(\varepsilon) \\ & + l\, B_w[K_{2,0}(\varepsilon) - K_{1,1}(\varepsilon)]\end{aligned} \tag{C.69}$$

Finally, the outflow fluxes are:

$$\Psi_{\mathrm{A}}^{Out} = [A^f + \frac{1}{2}B_u^f + \frac{1}{2}B_v^f]\, e^{-\varepsilon} + l[A + \frac{1}{2}B_u + \frac{1}{2}B_v + B_w \rho(\varepsilon)]M_0(\varepsilon) \qquad \text{(C.70)}$$

$$\Psi_u^{Out} = [\frac{1}{2}A^f + \frac{1}{3}B_u^f + \frac{1}{4}B_v^f]\, e^{-\varepsilon} + l[\frac{1}{2}A + \frac{1}{3}B_u + \frac{1}{4}B_v + \frac{1}{2}B_w \rho(\varepsilon)]M_0(\varepsilon) \qquad \text{(C.71)}$$

$$\Psi_v^{Out} = [\frac{1}{2}A^f + \frac{1}{4}B_u^f + \frac{1}{3}B_v^f]\, e^{-\varepsilon} + l[\frac{1}{2}A + \frac{1}{4}B_u + \frac{1}{3}B_v + \frac{1}{2}B_w \rho(\varepsilon)]M_0(\varepsilon) \qquad \text{(C.72)}$$

The final set of equations use a new set of integral functions similar to the k functions, but more complex (the $M_0(\varepsilon)$ function). These functions are called exponential moment functions. The derivation and exploration of these functions can be found in Minor (1993), Brennan (1996), and Miller (1998).

The exponential moment function of order *n* is represented as[3]

$$M_n(\varepsilon) = \int_0^1 (1-t)^n e^{-\varepsilon t} dt\,. \qquad \text{(C.73)}$$

Therefore, our moment function is equal to

$$M_0(\varepsilon) = \int_0^1 e^{-\varepsilon t} dt = \frac{1-e^{-\varepsilon}}{\varepsilon}\,. \qquad \text{(C.74)}$$

In the derivation of the parallelepiped equations, we encounter exponential moment functions of order *n* = 0 and *n* = 1. A recursive relation exists between these two functions that allows us to write (Mathews, Sjoden, and Minor, 1994):

$$M_n(\varepsilon) = \frac{1 - nM_{n-1}(\varepsilon)}{\varepsilon} \qquad \text{(C.75)}$$

And, for greater simplification of the derivation, let

$$\rho(\varepsilon) = \frac{M_1(\varepsilon)}{M_0(\varepsilon)}\,. \qquad \text{(C.76)}$$

[3] A good explanation of one-dimensional moment functions is provided in the paper by Mathews, Sjoden, and Minor (1994). As with the K functions, the exponential functions must be carefully implemented to retain numerical stability, especially when ε is small (see equation (C.74)).

# APPENDIX D: Procedure for Linear Characteristic Calculations

The linear characteristic method is part of the family of numerical integration techniques that solves the BTE by performing integrations along the directional path of particle flow to derive a set of fundamental numerical based equations. These equations are derived for a unit computational cell that resides in an array of similar unit cells. In three dimensions, the most common unit cell is the boxoid or cubic type cell. The problem with using this type of cell is that the orientation of the neutron particle flow changes with respect to the cells fixed position in a mesh array. Therefore, the derivation of the linear characteristic equations must be performed efficiently in a way that unit volumes or subsequent unit sub volumes are oriented along the particle streaming direction. These derivations were performed in Appendix C. In this section, the procedure for orienting and defining the unit sub volumes is described as well as the procedure for the breaking down of and recombining of the global inflow, outflow, and volumetric fluxes.

## Defining the Sub Cell Volumes

We begin our discussion by looking at the sub division of the global unit cell in figure D1. The sub volumes are determined by the orientation of the most recent directional ordinate picked from the $S_N$ set. For figure D1, we choose a z dominant (or $\xi$ dominant) directional ordinate to define our sub volumes[1]. If the ordinate lies off the

[1] The z dominant directional ordinate (as pictured in figure D1) serves as the basis for defining the orientation of all sub volumes in this research. For x and y dominated directional ordinates, a rotation of the unit cell is performed to avoid unnecessary derivations of new sub cells. This rotation of the global unit cell requires a swap between the inflow and outflow faces that corresponds directly with the correct x or y dominant case.

diagonal of the boxoid cell, the global unit cell is broken down into eleven different sub volumes[2] listed as in figure D1. Each inflow face is given a reference name. When looking at figure 2.2 in Chapter 2, Face 1 corresponds directly with the back inflow face, Face 2 corresponds directly with the down inflow face, and Face 3 corresponds directly with the left inflow face. Each inflow face has a given number of sub volumes associated with it based on whether or not the sub volume inflow face resides on the global cell inflow face. For instance, Face 3 consists of two tetrahedrons and a prism where tetrahedron 1 and the prism both exit on the top face of the cube while tetrahedron 2 exits out the front face.

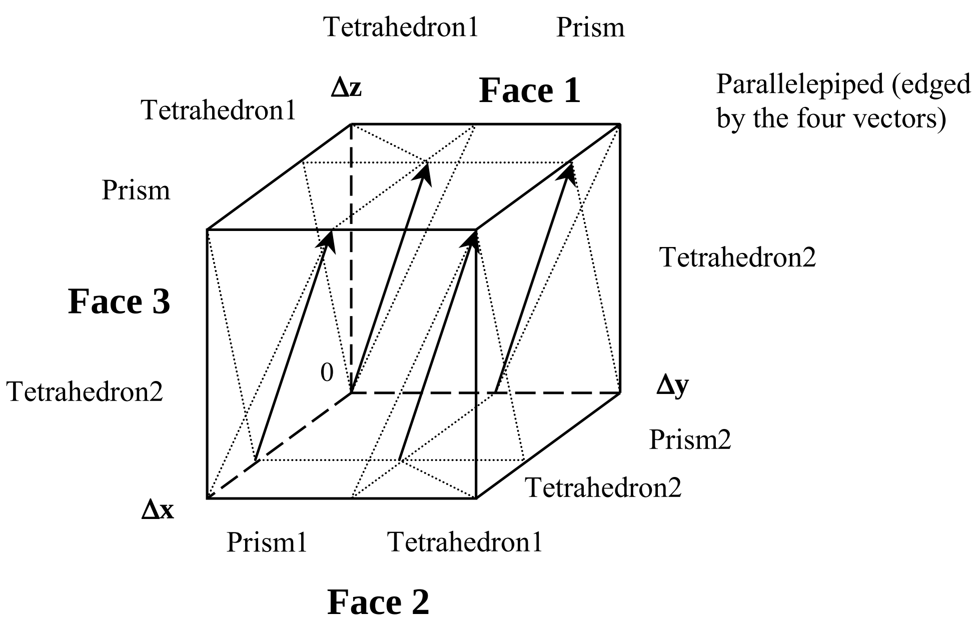

**Figure D1: Subdivision of Global Unit cell - Off Diagonal Ordinate**

[2] If the directional ordinate falls on the diagonal of the cell, five of the sub volumes disappear and only the six tetrahedrons remain.

The exiting faces of the unit cell (the front, right, and top face) are also composed of the exiting faces of the sub cells. Each face, both inflow and outflow, has a specified orientation depicted by the directional ordinate vector. For tetrahedrons, we have inflow and outflow faces defined by triangles while the prism faces are defined by parallelograms and rectangles. The parallelepiped has square inflow and outflow faces. The sub cell faces and volumes are defined by sets of basis vectors. For all volumes, the basis vectors are formed by tracing a path along three adjoining edges in a direction that finishes with the last vector pointing in the direction of the directional ordinate vector. The boundaries of the inflow and outflow areas are traced by two adjoining vectors. In both cases, the vectors are arranged so that the previous vector's head meets the tail of the next adjoining vector.

To illustrate the concept of vector basis formulation, we examine figure D2 for tetrahedron 1 on Face 1. The volumetric basis vectors that form Face 1, tetrahedron 1 consist of vectors $\vec{E}_1$, $\vec{E}_2$, and $\vec{E}_3$. The inflow face resides on the backside of the cube and is constructed from vectors $\vec{E}_1$ and $\vec{E}_2$. The outflow face is constructed with vector $\vec{E}_1$ and the vector formed by connecting the heads of vectors $\vec{E}_1$ and $\vec{E}_3$. The vectors are then defined by using the x or y distance as in figure D2 with subscripts A or B. The subscript A refers to the x or y distance that lies between the z-axis and the intersection of the directional ordinates vector head with the top of the unit cell. The B subscript refers to the remaining x or y distance. The points $X_a$ and $Y_a$ are found using the slope of the directional ordinate or for z dominant ordinates:

$$X_a = Abs(\mu / \xi)\ \Delta z \tag{D.1}$$

$$Y_a = Abs(\eta / \xi)\ \Delta z\ . \tag{D.2}$$

The basis vectors for the volume and area faces are defined in table D1 for all the sub cell shapes and the global unit cell. These volumetric basis vectors serve as global (x, y, z) reference matrices for the Jacobian transformations that convert the sub cells over to unit tetrahedrons, unit prisms, and unit cubes (see appendix C for unit cell orientations) in (*u, v, w*) space[3]. The basis vectors are always chosen so that the Jacobian transformation process yields a right handed set of orthogonal axes. Notice that the third basis vector, $\vec{E}_3$, lies in the same vector orientation as the directional ordinate. This must always be true because the transport equations of appendix C are defined along the streaming direction; therefore, the orientation of the sub cells in (x, y, z) to the streaming direction must match the orientation of the unit volumes with the streaming direction.

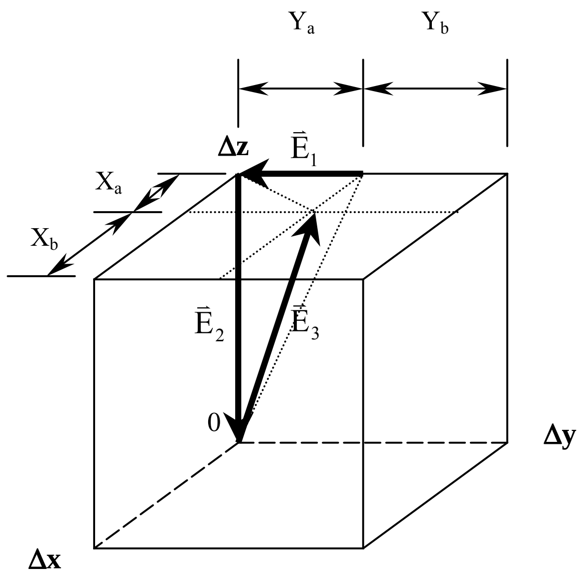

**Figure D2: Basis Vector Formulation for Tetrahedron 1, Face 1**

[3] The $\vec{E}_1$, $\vec{E}_2$, and $\vec{E}_3$ vectors correspond directly with the *u*, *v*, and *w* axis in the transformed coordinate system of the unit sub volumes of appendix C.

## Transformation and Calculation of Unit Cell Moments

The exact procedure for performing the transformation process for the Linear Characteristic method as well as the assembly of the fluxes from the unit sub volumes is described in detail in the paper by Mathews, Brennan, and Miller (2000). A brief overview of the method is provided in the following paragraphs with care taken not to show any detail. The reader is directed to the aforementioned paper for greater clarification and in depth understanding of the theory behind the transformation process and procedures required for the linear characteristic method.

In summary, the calculation of the unit cell flux requires that the zeroth and first moments across the cell inflow faces be known. The source zeroth and volumetric moments must also be known with in the cell volume. These values are either specified or known from previous iterations. The known values are used to compute global cell source coefficients across each input face to the global unit cell. The global coefficients describing the incoming flux shape are then transformed from (x, y, z) to ($u, v, w$) space using the Jacobi basis vectors (described in the previous section) of the global unit cell and the Jacobi matrix of the sub cells. The transformed coefficients are used to calculate the sub cell zeroth and first moments in the volume and across the exiting face. Each sub volume is calculated in a similar way using its own Jacobi matrix as the basis for the calculation. The transport equations required for this step were derived in appendix C. After the calculations are performed, the sub volume flux moments are transformed back from ($u, v, w$) to (x, y, z) space. The flux moments across the exiting face of the sub volume are also transformed from ($u, v$) to (x, y). The moments from all the sub cell volumes are averaged together using a volume-weighted average to calculate one volumetric zeroth moment and three volumetric first moments for the global unit cell. The flux moments across each exiting sub cell face are recombined using a similar technique of area weighted averaging to compile the zeroth and first moments of the flux across each of the three outflow faces of the global unit cell. The process repeats itself for each computational cell in the mesh.

| | Volumetric | | | Inflow Face | | Outflow Face | |
|---|---|---|---|---|---|---|---|
| | $\vec{E}_1$ | $\vec{E}_2$ | $\vec{E}_3$ | $\vec{E}_1$ | $\vec{E}_2$ | $\vec{E}_1$ | $\vec{E}_4$ |
| **Face 1** | | | | | | | |
| tetrahedron 1 | (0, -Ya, 0) | (0, 0, -Δz) | (Xa,Ya, Δz) | (0, -Ya, 0) | (0 ,0, -Δz) | (0, -Ya, 0) | (Xa, Ya, 0) |
| prism | (0, -Yb, 0) | (0,-Ya, -Δz) | (Xa,Ya, Δz) | (0, -Yb, 0) | (0,-Ya, -Δz) | (0, -Yb, 0) | (Xa, 0, 0) |
| tetrahedron 2 | (0, 0, Δz) | (0,-Ya, -Δz) | (Xa,Ya, Δz) | (0, 0, Δz) | (0,-Ya, -Δz) | (0, 0, Δz) | (Xa, 0, 0) |
| **Face 2** | | | | | | | |
| prism 1 | (0, Yb, 0) | (-Xa, -Ya, 0) | (Xa,Ya, Δz) | (0, Yb, 0) | (-Xa, 0, 0) | (0, Yb, 0) | (0,Ya, Δz) |
| tetrahedron 1 | (0, Ya, 0) | (-Xa, 0, 0) | (Xa,Ya, Δz) | (0, Ya, 0) | (-Xa, -Ya, 0) | (0, Ya, 0) | (0, 0, Δz) |
| tetrahedron 2 | (-Xa, 0, 0) | (0, -Ya, 0) | (Xa,Ya, Δz) | (-Xa, 0, 0) | (0, -Ya, 0) | (-Xa, 0, 0) | (Xa, 0, Δz) |
| prism 2 | (-Xb, 0, 0) | (0, -Ya, 0) | (Xa,Ya, Δz) | (-Xb, 0, 0) | (0, -Ya, 0) | (-Xb, 0, 0) | (Xa, 0, Δz) |
| parallepiped | (Xb, 0, 0) | (0, Yb, 0) | (Xa,Ya, Δz) | (Xb, 0, 0) | (0, Yb, 0) | (Xb, 0, 0) | (0, Yb, 0) |
| **Face 3** | | | | | | | |
| tetrahedron 1 | (Xa, 0, 0) | (-Xa, 0, -Δz) | (Xa,Ya, Δz) | (Xa, 0, 0) | (-Xa, 0, -Δz) | (Xa, 0, 0) | (0, Ya, 0) |
| prism | (Xb, 0, 0) | (-Xa, 0, -Δz) | (Xa,Ya, Δz) | (Xb, 0, 0) | (-Xa, 0, -Δz) | (Xb, 0, 0) | (0, Ya, 0) |
| tetrahedron 2 | (0, 0, -Δz) | (-Xa, 0, 0) | (Xa,Ya, Δz) | (0, 0, -Δz) | (-Xa, 0, 0) | (0, 0, -Δz) | (0,Ya, Δz) |
| **Global cell** | (Δx, 0, 0) | (0, Δy, 0) | (0, 0, Δz) | | | | |
| Left Face | | | | (Δx, 0, 0) | (0, 0, Δz) | | |
| Back Face | | | | (0, -Δy, 0) | (0, 0, Δz) | | |
| Down Face | | | | (-Δx, 0, 0) | (0, Δy, 0) | | |
| Right Face | | | | | | (-Δx, 0, 0) | (0, 0, Δz) |
| Front Face | | | | | | (0, Δy, 0) | (0, 0, Δz) |
| Top Face | | | | | | (Δx, 0, 0) | (0, Δy, 0) |

**Table D1: Basis Vectors for Sub Volume Cells and Global Unit Cell**

# APPENDIX E: Reactor Test Problem Input Data

The following tables summarize the input data used in the heterogeneous reactor test problems in Chapter 4. Table E1-E4 summarizes the data from table 7-1 of Duderstadt and Hamilton (1976). The macroscopic cross-section data was calculated based on 2 w/o, 3 w/o, and 4 w/o fuel. The fuel is assumed to be $UO_2$ at 9.975 g/cm$^3$. The cladding is assumed to by pure zirconium with cross section data taken from Bridgeman (1999).

| | U-235 | | | | |
|---|---|---|---|---|---|
| g | $\sigma_{tr}$ | $\nu$ | $\sigma_f$ | $\sigma_a$ | $\sigma_{gg}$ |
| 1 | 4.70 | 2.65 | 1.30 | 1.50 | 1.90 |
| 2 | 7.00 | 2.55 | 1.40 | 0.30 | 5.30 |
| 3 | 51.00 | 2.50 | 23.00 | 18.01 | 9.99 |
| 4 | 597.00 | 2.50 | 490.00 | 97.00 | 10.00 |

**Table E1: Four-Group U-235 Cross-Section Data (in Barns)**

| | U-238 | | | | |
|---|---|---|---|---|---|
| g | $\sigma_{tr}$ | $\nu$ | $\sigma_f$ | $\sigma_a$ | $\sigma_{gg}$ |
| 1 | 4.70 | 2.65 | 0.53 | 2.14 | 2.03 |
| 2 | 7.00 | 0.00 | 0.00 | 0.18 | 6.82 |
| 3 | 11.00 | 0.00 | 0.00 | 0.81 | 10.19 |
| 4 | 13.00 | 0.00 | 0.00 | 2.40 | 10.60 |

**Table E2: Four-Group U-238 Cross-Section Data (in Barns)**

|  |  | $H_2O$ |  |  |  | Zr |  |
|---|---|---|---|---|---|---|---|
| g | $\chi(E)$ | $\sigma_{tr}$ | $\sigma_a$ | $\sigma_{g\text{-}g+1}$ | $\sigma_{gg}$ | $\sigma_{tr}$ | $\sigma_{gg}$ |
| 1 | 0.575 | 3.08 | 0.00 | 2.81 | 0.27 | 7.11 | 5.53 |
| 2 | 0.425 | 10.52 | 0.00 | 4.04 | 6.48 | 7.7 | 7.65 |
| 3 | 0.00 | 16.55 | 0.04 | 4.14 | 12.38 | 7.7 | 7.65 |
| 4 | 0.00 | 68.60 | 0.57 | 0.00 | 68.03 | 8.1 | 8 |

**Table E3: Four Group Water and Zr Cross-Section Data (in Barns)**

| Material | Atom density (atoms/cm$^3$) | w/o |
|---|---|---|
| U-235 | 6.7582E+20 | 3 |
| U-235 | 9.0108E+20 | 4 |
| U-235 | 1.1263E+21 | 5 |
| U-238 | 2.1570E+22 | 3 |
| U-238 | 2.1353E+22 | 4 |
| U-238 | 2.1130E+22 | 5 |
| Water | 3.3427E+22 | NA |
| Zr | 4.2910E+22 | NA |

**Table E4: Atom Density for Reactor Materials**

# **Bibliography**

Acton, Forman, *Real Computing made Real – Preventing Errors in Scientific Calculations*, Princeton university Press, 1996

Atkinson, Kendall E., *An Introduction to numerical Analysis*, John Wiley & Sons, 1989

Brennan, C. R., *Characteristic Spatial Quadratures for Discrete Ordinates Neutral Particle Transport on Arbitrary Tetrahedral Meshes*, PhD Dissertation, Air Force Institute of Technology (June1996)

Bridgman, Charles J., *The Physics of Nuclear Explosives*, Class Notes, Air Force Institute of Technology, Fall 1999

Burden, Richard L. and Faires, Douglas J., *Numerical Analysis*, Brooks/Cole Publishing Company, 1997

Carlson, B. G., *Tables of Equal Weight Quadrature Eqn Over the Unit Sphere*, Los Alamos Scientific Laboratory, LA-4734, 1971

Carlson, B. G. and Lathrop, K. D., *Transport Theory, The Method of Discrete Ordinates*, Los Alamos Scientific Laboratory, LA-3251, 1965

Cochran, Robert G. and Tsoulfanidis, Nicholas, *The Nuclear Fuel Cycle: Analysis and Management*, The American Nuclear Society, 1990.

Clark, Melville and Hansen, Kent F., *Numerical Methods of Reactor Analysis*, Academic Press, 1964

Duderstadt, James J. and Hamilton, Louis J., *Nuclear Reactor Analysis*, John Wiley and Sons, Inc., 1976

Ellis, T. M. and Phillips, Ivor, *Fortran 90 Programming*, Addison_Wesley Publishing Company, 1994

Feltus, M.A., *Nuclear Engineering 431 Class Notes*, Penn State Univeristy, Spring 1995

Kreyszig, Erwin, *Advanced Engineering Mathematics*, John Wiley & Sons, 1993

Larmarsh, John R. *Introduction to Nuclear Engineering 2$^{nd}$ Edition*, Addison-Wesley Publishing Company, 1983

Lathrop, K.D., *Spatial Differencing of the Transport Equation: Positivity vs. Accuracy*, J. Comput. Phys., 4, p. 475-498, 1969

Lewis, E. E. and Miller, W.F., *Computational Methods of Neutron Transport*, American Nuclear Society, 1993

Miller, R. L., *Development of a Discrete Ordinates Code System for Unstructured Meshes of Tetrahedral Cells with Serial and Parallel Implementations*, PhD Dissertation, Air Force Institute of Technology (Dec. 1998)

Minor, Bryan M., *Exponential Characteristic Spatial Quadrature for Discrete Ordinates Neutral Particle Transport in Two-Dimensional Cartesian Coordinates*, Ph.D. Dissertation, Air Force Institute of Technology (Sept. 1993)

Marchuk, G. I., and Lebedev, V. I., *Numerical Methods in the Theory of Neutron Transport*, Hardwood Publishers, 1986

Mathews, Kirk, *On the Propagation of Rays in Discrete Ordinates*, Nuclear Science and Engineering:132, 155-180 (1999)

Mathews, Kirk and Miller, Rodney, and Brennan, Charlie, *Split-Cell Linear Characteristic Transport Method for Unstructured Tetrahedral Meshes*, Nuclear Science and Engineering: 136, 1-24(2000)

Mathews, Kirk and Sjoden, Glenn, and Minor, Bryon, *Exponential Characteristic Spatial Quadrature for Discrete Ordinates Radiation Transport in Slab Geometry*, Nuclear Science and Engineering:118, 24-37 (1994)

O'Dell, D. R. and Alcouffe, R. E., *Transport Calculations for Nuclear Analysis: Theory and Guidelines for Effective use of Transport Codes*, Los Alamos Scientific Laboratory, LA-10983-MS, September 1987

Ott, Karl O. and Neuhold, Robert J., *Introductory Nuclear Reactor Dynamics*, American Nuclear Society, 1985

Ott, Karl O. and Bezella, Winfred A., *Introductory Nuclear Reactor Statics*, American Nuclear Society, 1989

Stakgold, Ivar, *Green's Functions and Boundary Value Problems*, John Wiley & Sons, 1998

Suriano, M., *Independent Characteristic Spatial Quadrature Methods for Neutral Particle Transport*, to be published as PhD Dissertation, Air Force Institute of Technology (Dec. 2001)

# Vita

Captain Kristofer S. Labowski was born [redacted] in [redacted] Pennsylvania. He entered undergraduate studies at the Pennsylvania State University where he graduated with a Bachelor of Science degree in Nuclear Engineering and was commissioned in the U.S. Air Force on 06 Jan. 1996. He received an Educational Delay to pursue his Masters of Science in Nuclear Engineering at the Pennsylvania State University. There, he studied under the tutelage of reactor kinetics specialist where he developed a computer parallelization routine designed to link a diffusion based, three dimensional reactor kinetics code to a thermal hydraulic safety analysis code. His success earned him his master's degree in May 1997. Two years later in August 1999, he earned his Masters in Business Administration from Webster University, at the Florida Space Coast campus in Merritt Island.

Before entering graduate school at the Air Force Institute of Technology, his military career included an assignment to the Air Force technical Applications Center (AFTAC) located at Patrick AFB in Florida. There, he served as a laboratory liaison officer, and a branch chief where his responsibilities included overseeing seven national laboratory contracts vital to the AFTAC mission. He was also responsible for the acquisition of new environmental laboratory services designed to replace vital, lost Air Force laboratory services previously located at McClellan AFB, CA. After graduation, Captain Labowski received an assignment to US Strategic Command (STRATCOM) at Offutt AFB in Omaha, Nebraska.