Assessing the Quality of Three Dimensional Simplified Spherical Harmonic (SP_N) Radiation Transport Solutions to Source Detector Problems

by

Japan K. Patel

A PROJECT

submitted to

Oregon State University

University Honors College

In partial fulfillment of

the requirement for the

degree of

Honors Baccalaureate of Science in Nuclear Engineering

Presented on June 3, 2011

Commencement June 2011

AN ABSTRACT OF THE THESIS OF

Japan K. Patel for the degree of **Honors Baccalaureate of Science in Nuclear Engineering** to be presented on **June 8, 2011.** Title: **Assessing the Quality of Three-Dimensional Simplified-P_N Radiation Transport Solutions to Source-Detector Problems**

Abstract Approved _____

Dr. Todd Palmer

Abstract Body

The research objective is to assess the performance of the SP_N transport method, with a traditional finite volume spatial discretization in Cartesian geometry, on source detector problems relevant to urban detonation of an improvised nuclear device. In particular, we observe the character of the solution as a function of the number of angular moments. To perform this investigation, we develop a computer code that calculates the solution of the linear system of equations of the SP_N method with finite volume spatial differencing given information about the radiation source, SP_N order, nuclear data, spatial mesh and boundary conditions. Matlab is used as the programming platform. The software is verified via comparison with analytic results and independent numerical results in slab geometry. Upon completion of the comparison we investigate the quality of the solution of complicated 3D problems and the effect of angular expansion order on the quality of results using the industry standard Monte Carlo code MCNP5 results but it is found that the quality of solution improves with increasing expansion order, but errors arising out of poor spatial resolution cannot be reduced with increasing expansion order.

Assessing the Quality of Three Dimensional Simplified Spherical Harmonic (SP_N) Radiation Transport Solutions to Source Detector Problems

by

Japan K. Patel

A PROJECT

submitted to

Oregon State University

University Honors College

In partial fulfillment of

the requirement for the

degree of

Honors Baccalaureate of Science in Nuclear Engineering

Presented on June 3, 2011

Commencement June 2011

Honors Baccalaureate of Science in Nuclear Engineering project of Japan K. Patel presented on June 8, 2011.

APPROVED:

_____

Mentor, representing Nuclear Engineering

_____

Committee Member, representing Nuclear Engineering

_____

Committee Member, representing Mathematics

_____

Department Head, Nuclear Engineering

_____

Dean, University Honors College

I understand that my project will become part of the permanent collection of Oregon State University Honors College. My signature below authorizes release of my project to any reader upon request.

_____

Japan K. Patel, Author

Acknowledgements

At this point, I'd like to extend my heartiest gratitude towards my mentor Dr. Todd Palmer who helped me all the way through the process of writing this thesis. It would have been impossible to write the thesis without his constant encouragement and support.

I'd also like to thank my parents or all their love and intermittent support.

I am very grateful to Dr. Andrew Klein and Mr. S. Todd Keller for agreeing to be on the thesis committee.

Table of Contents

Section	Title	Page
1	Introduction and Background.....	1
	1.1 Transport Theory.....	2
	1.2 Discretization Methods.....	3
	1.3 Approach.....	5
2	Software Production.....	7
	2.1 Angular Discretization.....	7
	2.2 Spatial discretization.....	10
	2.3 Boundary Conditions.....	10
	2.4 Algorithm.....	12
3	Software Verification.....	18
4	Software Solution Assessment - Methodology and Results.....	26
5	Conclusion.....	30
6	Bibliography.....	61

List of Figures

Figure	Title	Page
1	Pictorial representation of test problem for the trend test.....	19
2	Plot of the solution of the first setting of the test problem for the trend test.....	19
3	Plot of the solution of the second setting of the test problem for the trend test.....	19
4	Plot of the solution of the third setting of the test problem for the trend test.....	20
5	Pictorial representation of the test system used for the symmetry test and test using the method of exact solutions.....	21
6	Surface-Contour plot of solution obtained for the symmetry.....	21
7	Surface-Contour plot of solution obtained for testing using method of exact solutions.....	22
8	Pictorial representation of test system for the comparison test.....	24
9	Plot of the data used for comparison test.....	24
10	Semilog plot of the data used for the comparison test.....	25
11	Pictorial representation of the test system for SPn solution's performance assessment.....	26
12	SP_{14} flux profile for the slice at z = 9.5 cm of the 3D test system.....	27
13	Slice wise average error in the SPn solution with respect to MCNP benchmark as a function of slice distance from the origin.....	28
14	Average system error in SPn solution with respect to MCNP benchmark as a function of expansion order.....	28
15	Plots of sample SP12 flux profiles for various slices of solution for the benchmark problem.....	56
16	Plot of sample SP12 error profile for slice at z = 0.5 cm.....	59

List of Appendices

Appendix	Title	Page
A	Legendre Gauss Quadrature Sets.....	31
B	Derivation of Steady state Mono-Energetic Transport Equation.....	33
C	Derivation of Finite Volume Spatial Discretization.....	38
D	Derivation of Finite Volume Spatial Discretization.....	42
E	Derivation of Boundary Conditions.....	45
F	Solution File for Comparison Test.....	47
G	Appendix G - MCNP Input Deck.....	49
H	Generic Code Block Used in the SPn-FVM code.....	51
I	Error Tables.....	54
J	Sample Flux and Error Profiles.....	56
K	Future Work.....	60

Assessing the Quality of Three Dimensional Simplified Spherical Harmonic (SP_N) Radiation Transport Solution to Source Detector Problems

INTRODUCTION and BACKGROUND

The threat of nuclear terrorism is fairly widely known and the possibility of such an attack cannot be ignored. It therefore becomes essential that a tool be developed that would effectively help us predict the damage from these events. The best way to combat such events is of course to prevent them from occurring but in the event of a radiological dispersal device or an improvised nuclear device detonation in an urban setting, we must assess the damage and mitigate its ill effects.

The ability to predict the dispersal of relevant radioactive material through air and the transport of radiation from that material in the environment is a necessary component of this emergency response. This simulation tool involves a three stage development process:

1. Develop a 3-D radiation transport code and assess its performance
2. Develop an aerial particle/particulate dispersal code and assess its performance
3. Unify the transport and dispersal codes to generate the radiological dispersal code and assess its performance

The prediction of the radiological dispersal gives an estimate of the relevant radiation source distribution.

This thesis describes an attempt at the execution of the first step of the previously stated strategy - generation and verification of a three dimensional transport code written for idealized system assuming energy and time independence along with isotropic scattering.

All such transport codes require application of the transport theory. The next section will talk about the transport theory.

1.1 Transport Theory

Predicting the distribution of radiation in different systems, begins with general mathematical formulation of the problem. Transport theory provides the relevant mathematical framework for the radiation distribution. The transport equation was derived by the Ludwig Boltzmann more than a century ago. The equation was used to describe the kinetic behavior of gases, and is still used for that purpose. The same equation can be used describe the transport of neutral particles: photons and neutrons.

This integro-differential equation is an accounting of the number of particles entering, number of particles leaving, the number of particles created, and the number of particles extinguished (absorbed) in the phase space of interest. Analytic solution of the transport equation is impossible in most realistic situations - numerical simulation is most often required. This can take significant computational power and mathematical skill as solution can be a complex function of seven independent variables – space (3), direction (2), time (1) and energy (1).

There are two broad categories of methods used to solve transport problems:

1. Monte Carlo or stochastic methods
2. Deterministic methods

Many deterministic methods have been developed throughout the past sixty years. These methods begin with the general transport equation with dependence on energy, space, time and angular position and involve discretizing the equation in all the independent variables to yield a linear system of equations.

In this research, we assume steady state, isotropic scattering and energy independence. The steady state and energy independence assumptions render the complex transport equation in the following form:

$$\boldsymbol{\Omega}.\boldsymbol{\nabla\Psi}(\boldsymbol{r},\Omega)+\boldsymbol{\sigma_t}(\boldsymbol{r})\boldsymbol{\Psi}(\boldsymbol{r},\Omega)=\ \boldsymbol{Q}(\boldsymbol{r},\Omega)+\ \boldsymbol{\sigma_t}(\boldsymbol{r})\boldsymbol{c}(\boldsymbol{r})\int_{\boldsymbol{0}}^{\boldsymbol{4\pi}}\boldsymbol{f}(\Omega'\rightarrow\Omega)\boldsymbol{\Psi}(\boldsymbol{r},\Omega')\boldsymbol{d}\Omega'$$

Where,

Ω = particle direction vector; Φ = scalar flux; σ_t = macroscopic total cross section; Ψ = angular flux; Q = Source

Note: See appendix B for the derivation of this equation

1.2 Discretization Methods

There is a vast literature base on computational methods for radiation transport problems. Methods of discretizing the **angular variable** of the transport equation fall into categories:

1. Discrete Ordinates

2. Functional Expansion

The discrete ordinates method represents the unknown angular flux by its values at a discrete set of angles. In other words, the angular domain is discretized into a mesh of discrete points. Angular integrals are performed by numerical quadrature.

The functional expansion method involves an expression of the angular dependence of the flux in a finite number of known functions. These functions are polynomials (in one dimension) or spherical harmonics (on the unit sphere). The property of orthogonality is very important while using this method. (Duderstadt and Hamilton, 1976). The method involves solving a set of equations for the coefficients in the expansion.

In the 'Simplified P_N' (SP_N) technique, the angular variable is expanded in Legendre polynomials (1D spherical harmonic functions) transforming the integro- differential Boltzmann equation into a coupled set of elliptic partial differential equations.

Other function expansion methods use the spherical harmonic functions and\or Fourier series expansion.

Discretizations of the **spatial variable** are categorized as:

1. Characteristic methods
2. Finite Element methods
3. Finite Volume methods

Characteristic methods use the method of characteristics to analytically invert the streaming plus collision operator in the discrete ordinates equation. It uses exact equation, given an assumed form of an incident flux and the source. (Palmer, 2011)

Finite Element methods restrict the spatial variation of the transport solution to a fixed (user defined) functional space. (Palmer, 2011)

The Finite Volume methods involve integration of the transport equation over user defined volumes - usually cells of the discretization mesh - to obtain a conservation equation over that volume. The "balance equations" are exact but involve cell average and cell-surface unknowns. Closure approximations, relating cell-surface to cell-average quantities, are used to generate a linear system of equations for the cell-average unknowns. (Palmer, 2011)

A finite volume spatial discretization is used in this thesis. The choice of spatial and angular discretizations dictate the accuracy of the numerical transport solution.

1.3 Approach

The basic approach is to develop a computer code that calculates the solution of the linear system of equations of the SP_N method with finite volume spatial differencing given information about the radiation source distribution, SP_N order, nuclear cross section data and boundary conditions. The software is verified using comparisons with exact solutions, symmetry, and comparisons with independent numerical results in slab geometry. The effect of angular expansion order on the quality of results for source

detector problems is investigated and comparisons with an industry standard Monte Carlo code MCNP5 are made.

2 Software Production

The following four step procedure is used for the production of the SPn-FVM transport code:

1. Discretize the relevant transport equation form with respect to angle
2. Discretize the relevant transport equation form with respect to space
3. Derive relevant equations for vacuum and reflecting boundary conditions
4. Put the discretized equation set in a matrix equation form and solve the system to get the desired scalar flux distribution

Relevant transport equation (as stated in Section 1.1):

$$\Omega.\boldsymbol{\nabla\Psi}(\boldsymbol{r},\Omega)+\sigma_t(\boldsymbol{r})\boldsymbol{\Psi}(\boldsymbol{r},\Omega)=\ Q(\boldsymbol{r},\Omega)+\ \sigma_t(\boldsymbol{r})c(\boldsymbol{r})\int_{\mathbf{0}}^{\mathbf{4\pi}}\boldsymbol{f}(\Omega'\rightarrow\Omega)\boldsymbol{\Psi}(\boldsymbol{r},\Omega')\boldsymbol{d}\Omega'$$

2.1 Angular Discretization

Upon using the spherical harmonic expansion in plane geometry, and carrying out all the calculations one obtains the following SP_{2n-1} equations:

$$\frac{2n+1}{4n+1}\nabla.\Phi_{2n+1}(\boldsymbol{r})+\frac{2n}{4n+1}\nabla.\Phi_{2n-1}(\boldsymbol{r})+\sigma(r)\Phi_{2n}(\boldsymbol{r})=q_0(\boldsymbol{r})\delta_{n0}$$

$$\frac{2n+2}{4n+3}\nabla.\Phi_{2n+2}(\boldsymbol{r})+\frac{2n+1}{4n+3}\nabla.\Phi_{2n}(\boldsymbol{r})+\sigma(\boldsymbol{r})\Phi_{2n+1}(r)=0$$

where 2n-1 represents the moment of flux and δ_{n0} represents the Kronecker delta function.

Note: For derivation see Appendix C

Now, eliminate the odd moments to obtain the only even moments form of the SP_{2n-1} equations. Follow the following steps to get there: Find $\Phi_{2n+1}(r)$ and $\Phi_{2n-1}(r)$from the above equations and substitute them in the first equation of the SP_{2n-1} equations. Upon doing that, the following general equation is obtained:

$$\nabla.(\frac{-1}{\sigma(r)}\nabla\Phi_{2n-2}(\boldsymbol{r}))\frac{2n}{2n+1}\frac{2n}{4n+1}+\nabla.(\frac{-1}{\sigma(r)}\nabla\Phi_{2n}(\boldsymbol{r}))(\frac{2n}{2n+1}\frac{2n}{4n+1}+\frac{2n+1}{4n+1}\frac{2n+1}{4n+3})$$

$$+\nabla.(\frac{-1}{\sigma(r)}\nabla\Phi_{2n+2}(\boldsymbol{r}))\frac{2n+1}{4n+1}\frac{2n+1}{4n+3}+\sigma(r)\Phi_{2n}(\boldsymbol{r})=q_0(\boldsymbol{r})\delta_{n0}$$

This equation in turn can then be written as the general condensed equation:

$$\nabla.(\frac{1}{\sigma(r)}\nabla(CF))-\sigma(\boldsymbol{r})+Q=0$$

which is a tri-diagonal matrix where,

$$F=[\Phi_0\,(r)\,\Phi_2(r)\ldots.\Phi_{2n-2}(\boldsymbol{r})]^T$$

$$Q=\;[q_0(\boldsymbol{r})\;\;0\ldots..0]^T$$

(Coilini et. al., 2002)

The differential operator in the condensed equation can be diagonalized by representing F as a linear combination of eigenvectors W_b of matrix C as

$$F=\sum_{b=1}^{N}\Phi_b W_b.$$

(Coilini et. al., 2002)

From the properties of Legendre polynomials, eigenvalues of A can be related to zeros of $P_{2n}(\mu)$ as eigen values = $\mu_a{}^2$. Also, according to the properties of biorthogonal basis (U_a) formed by eigenvectors of the adjoint matrix of A and the eigenvectors of A, the first component of the basis is chosen to be 1 and eigenvectors are normalized such that $U_a.W_b$=the Kronecker delta function, the first component of W_a turns out to be the weight w_a of the Gauss Legendre integration scheme. Thus by substituting the flux expansion term in the general condensed equation, and taking scalar product with U_a, one observes that

$$\Phi_0 = \sum_{b=1}^{N} \Phi_b w_b$$

(Coilini et. al., 2002)

and that the general equation takes the following form

$$\nabla.\left(\frac{\mu_a^2}{\sigma(r)}\nabla\Phi_a(\boldsymbol{r})\right) - \sigma(\boldsymbol{r})\Phi_a + q_0(\boldsymbol{r}) = 0 \qquad where\ (a = 1,2,\ldots N)$$

where,

$$q_0(\boldsymbol{r}) = \sigma_s(\boldsymbol{r}) \sum_{b=1}^{N} \Phi_b w_b + S(\boldsymbol{r})$$

(Coilini et. al., 2002)

The above equation set is used in the code and represents an SP_N angular discretization on the relevant form of the Boltzmann transport equation.

2.2 Spatial Discretization

Upon observing the SP_N - angle discretized transport equation, it is clear that the only complicated term in the equation is the streaming term. We use the traditional Finite Volume method to discretize the gradient term and using the assumption that the slope on both sides of the node is same, we formulate the following general equation:

$$\boldsymbol{\Phi_{l-1}\frac{-2D_{l-1}D_{jkl}\Delta x_j\Delta y_k}{D_{jkl}\Delta z_{l-1}+D_{l-1}\Delta z_l}+\Phi_{l+1}\frac{-2D_{l+1}D_{jkl}\Delta x_j\Delta y_k}{D_{jkl}\Delta z_{l+1}+D_{l+1}\Delta z_l}+\Phi_{k-1}\frac{-2D_{k-1}D_{jkl}\Delta x_j\Delta z_l}{D_{jkl}\Delta y_{k-1}+D_{k-1}\Delta y_k}}$$

$$\boldsymbol{+\Phi_{k+1}\frac{-2D_{k+1}D_{jkl}\Delta x_j\Delta z_l}{D_{jkl}\Delta y_{k+1}+D_{k+1}\Delta y_k}+\Phi_{j-1}\frac{-2D_{j-1}D_{jkl}\Delta y_k\Delta z_l}{D_{jkl}\Delta x_{j-1}+D_{j-1}\Delta x_j}+\Phi_{j+1}\frac{-2D_{j+1}D_{jkl}\Delta y_k\Delta z_l}{D_{jkl}\Delta x_{j+1}+D_{j+1}\Delta x_j}-\Phi_{jkl}\left(\frac{-2D_{j+1}D_{jkl}\Delta y_k\Delta z_l}{D_{jkl}\Delta x_{j+1}+D_{j+1}\Delta x_j}\right.}$$

$$\boldsymbol{+\frac{-2D_{j-1}D_{jkl}\Delta y_k\Delta z_l}{D_{jkl}\Delta x_{j-1}+D_{j-1}\Delta x_j}+\frac{-2D_{k+1}D_{jkl}\Delta x_j\Delta z_l}{D_{jkl}\Delta y_{k+1}+D_{k+1}\Delta y_k}+\frac{-2D_{k-1}D_{jkl}\Delta x_j\Delta z_l}{D_{jkl}\Delta y_{k-1}+D_{k-1}\Delta y_k}+\frac{-2D_{l+1}D_{jkl}\Delta x_j\Delta y_k}{D_{jkl}\Delta z_{l+1}+D_{l+1}\Delta z_l}+\frac{-2D_{l-1}D_{jkl}\Delta x_j\Delta y_k}{D_{jkl}\Delta z_{l-1}+D_{l-1}\Delta z_l}}$$

$$\boldsymbol{\left.+\sigma_{jkl}\Delta x_j\Delta y_k\Delta z_l\right)=\sigma_s(r)\sum_{b=1}^{N}\Phi_b w_b+S(r)}$$

where,

$$D=\frac{\mu_a^2}{\sigma(r)}$$

Note: See Appendix D for derivation.

The above equation represents general discretized SP_N-FVM difference equation and is extensively used in the transport code.

2.3 Boundary Conditions

The only calculation required before an algorithm can be written is that of the boundary conditions. Two boundary conditions have been considered here - vacuum & reflecting.

Reflected boundary: The coefficient of the term on which the reflecting boundary is placed becomes zero. At the same time the term associated with that boundary term in the summation of Φ_{jkl} also goes to zero. For example, if the k-1th term in the general discretized SP$_N$-FVM difference equation has a reflected boundary, then that equation becomes:

$$\Phi_{l-1}\frac{-2D_{l-1}D_{jkl}\Delta x_j\Delta y_k}{D_{jkl}\Delta z_{l-1}+D_{l-1}\Delta z_l}+\Phi_{l+1}\frac{-2D_{l+1}D_{jkl}\Delta x_j\Delta y_k}{D_{jkl}\Delta z_{l+1}+D_{l+1}\Delta z_l}$$
$$+\Phi_{k+1}\frac{-2D_{k+1}D_{jkl}\Delta x_j\Delta z_l}{D_{jkl}\Delta y_{k+1}+D_{k+1}\Delta y_k}+\Phi_{j-1}\frac{-2D_{j-1}D_{jkl}\Delta y_k\Delta z_l}{D_{jkl}\Delta x_{j-1}+D_{j-1}\Delta x_j}+\Phi_{j+1}\frac{-2D_{j+1}D_{jkl}\Delta y_k\Delta z_l}{D_{jkl}\Delta x_{j+1}+D_{j+1}\Delta x_j}+\Phi_{jkl}\left(\frac{2D_{j+1}D_{jkl}\Delta y_k\Delta z_l}{D_{jkl}\Delta x_{j+1}+D_{j+1}\Delta x_j}\right.$$
$$\left.+\frac{2D_{j-1}D_{jkl}\Delta y_k\Delta z_l}{D_{jkl}\Delta x_{j-1}+D_{j-1}\Delta x_j}+\frac{2D_{k+1}D_{jkl}\Delta x_j\Delta z_l}{D_{jkl}\Delta y_{k+1}+D_{k+1}\Delta y_k}+\frac{2D_{l+1}D_{jkl}\Delta x_j\Delta y_k}{D_{jkl}\Delta z_{l+1}+D_{l+1}\Delta z_l}+\frac{2D_{l-1}D_{jkl}\Delta x_j\Delta y_k}{D_{jkl}\Delta z_{l-1}+D_{l-1}\Delta z_l}+\sigma_{jkl}\Delta x_j\Delta y_k\Delta z_l\right)$$
$$=\sigma_s(r)\sum_{b=1}^{N}\Phi_b w_b+S(r)$$

Similarly if the j-1th term has a reflected boundary, that equation changes.

Vacuum boundary: The coefficient of the term on which vacuum boundary is placed becomes zero (the term vanishes from the equation and hence from the matrix). The term associated with that boundary in the summation term of Φ_{jkl} changes to $\frac{2\mu_a^2}{\sigma_i x_i+2\mu_a}$ where σ_i represents the cell-average total cross section, x_i represents the cell dimension normal to the boundary and μ_a is the Gauss Legendre ordinate for specific moment.

For example, the j-1th term in general discretized SP$_N$-FVM difference equation has a vacuum boundary, the equation becomes:

$$\Phi_{l-1}\frac{-2D_{l-1}D_{jkl}\Delta x_j\Delta y_k}{D_{jkl}\Delta z_{l-1}+D_{l-1}\Delta z_l}+\Phi_{l+1}\frac{-2D_{l+1}D_{jkl}\Delta x_j\Delta y_k}{D_{jkl}\Delta z_{l+1}+D_{l+1}\Delta z_l}+\Phi_{k-1}\frac{-2D_{k-1}D_{jkl}\Delta x_j\Delta z_l}{D_{jkl}\Delta y_{k-1}+D_{k-1}\Delta y_k}$$

$$+\Phi_{k+1}\frac{-2D_{k+1}D_{jkl}\Delta x_j\Delta z_l}{D_{jkl}\Delta y_{k+1}+D_{k+1}\Delta y_k}+\Phi_{j+1}\frac{-2D_{j+1}D_{jkl}\Delta y_k\Delta z_l}{D_{jkl}\Delta x_{j+1}+D_{j+1}\Delta x_j}-\Phi_{jkl}\left(\frac{-2D_{j+1}D_{jkl}\Delta y_k\Delta z_l}{D_{jkl}\Delta x_{j+1}+D_{j+1}\Delta x_j}\right.$$

$$+\frac{2\mu_a^2\Delta y_k\Delta z_l}{\sigma_i x_i+2\mu_a}+\frac{-2D_{k+1}D_{jkl}\Delta x_j\Delta z_l}{D_{jkl}\Delta y_{k+1}+D_{k+1}\Delta y_k}+\frac{-2D_{k-1}D_{jkl}\Delta x_j\Delta z_l}{D_{jkl}\Delta y_{k-1}+D_{k-1}\Delta y_k}+\frac{-2D_{l+1}D_{jkl}\Delta x_j\Delta y_k}{D_{jkl}\Delta z_{l+1}+D_{l+1}\Delta z_l}$$

$$\left.+\frac{-2D_{l-1}D_{jkl}\Delta x_j\Delta y_k}{D_{jkl}\Delta z_{l-1}+D_{l-1}\Delta z_l}+\sigma_{jkl}\Delta x_j\Delta y_k\Delta z_l\right)=\sigma_s(r)\sum_{b=1}^{N}\Phi_b w_b+S(r)$$

Note: For derivation of boundary conditions see Appendix E.

2.4 Algorithm

The following algorithm is used to generate the SP_N-FVM transport code:

1. **Create the spatial and angular discretization matrix**

    - Ask for user input of angular discretization order, total number of cells in x direction, total number of cells in y direction, total number of cells in z direction, cell-wise dimensions in x, y and z directions, cell averaged total cross-sections, cell averaged scattering cross sections, boundary conditions, weights of Gauss quadrature set for specified order, ordinates of the Gauss quadrature set for specified order
    - Generate vectors that specify the indices of cells that are on the boundaries: first create the vector that contains indices of cells on the left boundary by creating a vector starting from 1 and going to i*j*k (total number of cells in mesh) with a spacing of i between each element; then generate a vector that contains indices of cells on the right boundary similarly but with the vector

starting from i instead of 1. Then generate the vector containing indices of cells on the northern boundary and southern boundary by creating *for loops* as shown in the code (in Appendix H); Then after create vector having indices on top and bottom of the cubical/cuboidal system by creating a vector from 1 to i*j to represent cells on the top boundary and a vector from i*j*k-i*j+1 to i*j*k to represent cells on the bottom boundary. Spacing in both cases must be 1.

- Upon doing that, start a general *for loop* going from 1 to order of angular discretization
- Define an empty matrix that takes the form of the coefficient matrix after necessary loops are executed
- Initiate an *if* loop which according to the iteration number of the greater *for* loop generates a vector containing cell wise cell averaged diffusion coefficients
- Under the *if* statement, start a new *for* loop from 1 to i*j*k that would go on to generate the coefficient matrix using to equations 5, 6 and 7. Under the local *for* loop, create a zero row vector containing i*j*k elements. Generate entries in the row vector that correspond to the coefficients of flux elements along z, x and y directions. To make entries in the row vector for the z direction, start an *if* statement and test for non existence of iteration number of local *for* loop in vectors containing upper and lower boundary cell numbers. Upon validation make entries in the (iteration number - i*j)th term (iteration number + i*j)th term and (iteration number)th term using equation 5. If that condition does not hold, test if the cell is on upper boundary using the vector containing numbers cell

indices on the upper boundary. If the test succeeds, then there is no entry required for the (iteration number - i*j)th term and enter the (iteration number + i*j)th term in the row vector. Next test for the boundary condition using boundary condition input vector and enter the (iteration number)th term in the row vector according to equation 6 or 7. If the iteration number does not correspond to any number in the vector containing indices for the upper boundary, it means that the cell exists on the bottom boundary. The (iteration number - i*j)th entry is made in the row vector while the (iteration number + i*j)th term is not entered since it cell is on the bottom boundary of the mesh. Then after boundary condition test is carried out using an *if* statement and relevant entries are made to the (iteration number)th term of the row vector in accordance with equations 6 and 7.

- A similar logic is used for entry of coefficient terms along x and y directions. The terms along the x direction are the (iteration number ± 1)th terms and boundary conditions tested are for the left and right boundary. The terms along the y direction are (iteration number ± i)th terms and the boundary conditions are tested for north and south boundaries. Once all seven terms corresponding to coefficients coming from streaming term are entered into the row vector, the coefficient corresponding to the reaction rate is added to the (iteration number)th term. The first cycle of this loop hence creates the row vector that corresponds to the first cell of the mesh. Once all i*j*k iterations are executed, a coefficient matrix is obtained and the local *for* loop is terminated. Upon its termination, a new matrix is generated which serves as

the first row block of the final discretization coefficient matrix. The formation of this new row block is shown in the code.

- After the local *if* statement is ended, and the coefficient matrix is set to an empty matrix.
- The same exact code block is repeated with changing *if* statements and changing row block matrices. The *if* statement checks for the global *for* loop iteration number and accordingly creates the diffusion coefficient vector and uses it to create a new row vector representing that diffusion coefficient. The number of block rows increases and keeps on increasing till the complete matrix is formed as the global *for* loop iteration number becomes equal to the order specified by user.
- Now the final and complete coefficient matrix representing the set of angularly and spatially discretized set transport equations has been generated.

2. **Upon creation of the spatially and angularly discretized matrix, perform source iteration to obtain flux moments**

- Ask the user to input the cell-averaged source along with tolerance
- Generate the source vector by multiplying cell volume by the cell sources using a *for* loop
- Generate the vector that contains cell wise product of cell averaged scattering cross section and cell volume using a for loop. Call it the scattering vector

- Now create the fixed source by repeating the source vector (expansion order/2) number of times and creating a greater column vector. This is the vector that goes into the matrix equation representing S(**r**).
- Generate initial flux guess for the iteration term on the RHS of the equation. This is a zero vector with i*j*k*order elements
- Set the error to 1 and then start a *while* loop that goes on until the error value is below the tolerance value entered by the user
- Under the *while* loop start a new for loop that generates the vector that representing scattering and on which the iteration is carried out
- Under this *for* loop, element-wise multiply the previously generated scattering vector, corresponding quadrature weight and intital flux guess.
- Sum the vectors obtained as a result of each iteration and create the new vector by repeating the summed vector (expansion order/2) number of times to create a vector that order number of times longer using a new *for* loop. Now add the new summed vector and the fixed source vector to obtain final source vector that is used in flux calculation.
- Set old equal to the initial guess
- Now generate new flux using the backslash operator to solve the matrix equation
- Set new equal to newly obtained flux
- Set initial guess equal to the initial source
- Find maximum error using new and old vectors
- The loop keeps repeating till that error goes below preset tolerance value

3. **Generate the scalar flux**

- Use the equation $\Phi_0 = \sum_{b=1}^{N} \Phi_b w_b$ to obtain the scalar flux. Divide the converged flux row vector obtained after execution of the while loop into order number of row vectors of length i*j*k.
- Multiply relevant row vectors element wise with respective quadrature weights and the add these row vectors to get the scalar flux. Use a simple for loop to carry out this step.

4. **Plot the scalar flux with respect to spatial position**

- To plot the scalar flux with respect to position - varying x and y with fixed z, divide the flux vector into k number of row vectors of size i*j. Choose k and hence extract the k^{th} section of the scalar flux vector of size i*j
- Reshape the vector into a i*j matrix using the reshape function
- Using meshgrid function, create a mesh that corresponds to the recently created matrix cell wise - determines spatial position along mesh
- Plot the flux using surfc function

The required plot depicting the flux distribution in the system can now be obtained by creating a code along the lines of the above algorithm.

3 Software Verification

Four methods are employed in this thesis for the verification of the software:

1. Trend Test
2. Symmetry Test
3. Method of Exact Solutions
4. Comparison Test

Trend testing: For trend testing, a set of calculations are carried out using various input parameters and an attempt is made to discover an emerging trend. In order for the code to pass the trend test, the trend that emerges from the set of calculations performed using the code must match the expected trend. It is the lowest of bars for a code to pass and hence is used generally in the code development stage.

Here, we will gradually increase the lengths of the cells in a source free pure absorber system with sources only on the left boundary of the system. There are reflecting boundaries on all boundaries except for the one on the right side which is a vacuum boundary.

Test problem: 7 by 7 by 7 system with l=b=h=1cm, 5cm, and 7cm. The absorption cross section in the system is 0.5 and is equal to the total cross section as the system is a pure absorber and the source on left is 6 /cc/s. A pictorial representation of the system is given in figure 1. The red area indicates the source region and the blue area indicates the source free region of the system. The lighter face in the picture indicates vacuum boundary and the darker faces are reflecting boundaries.

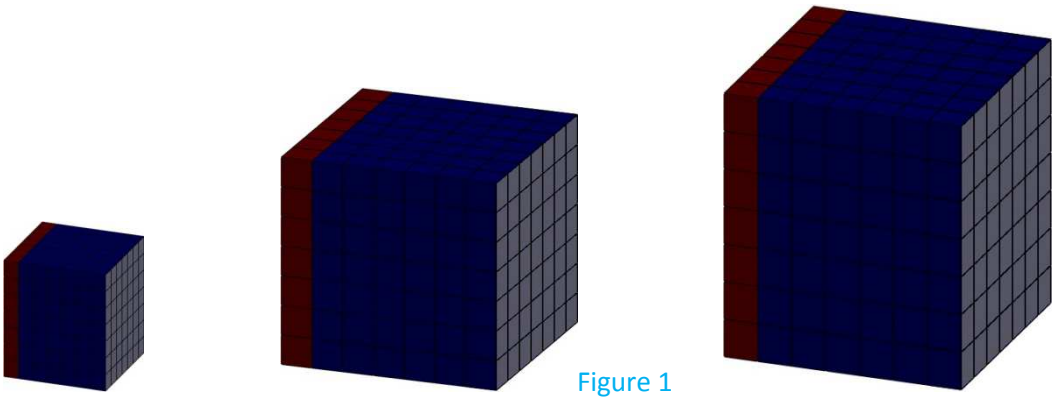


Figure 1

Expected result: The peak on left boundary must approach the infinite medium solution magnitude, here 12. The following plots show the results of this test problem:

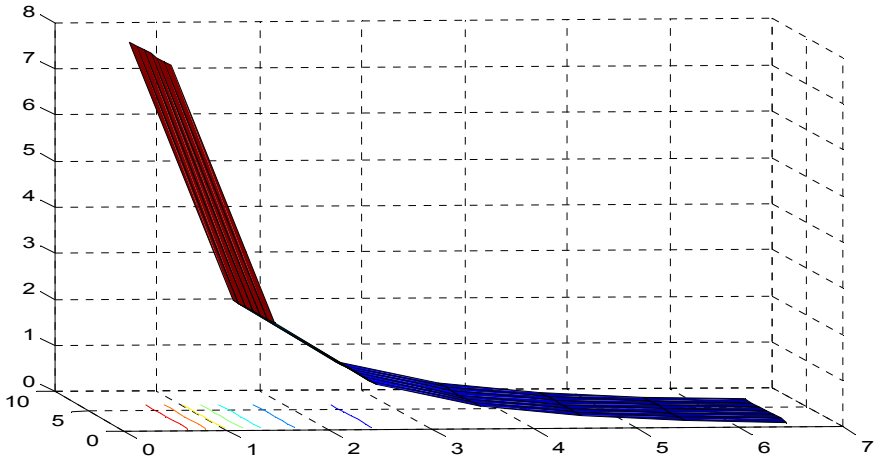


Figure 2

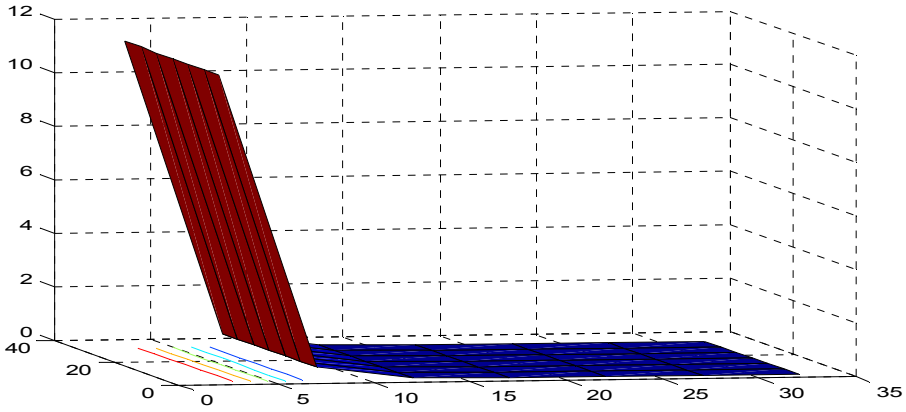


Figure 3

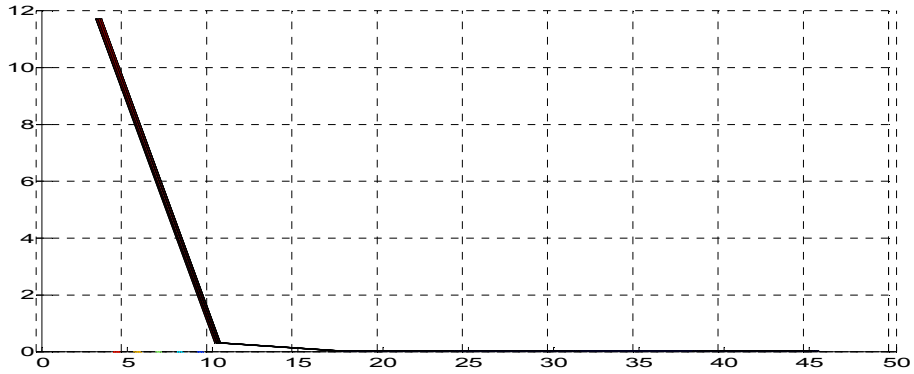


Figure 4

Observe that the shape of flux is similar to that of an exponential curve which is the expected solution of the source free pure absorber problem.

Figures 1, 2 and 3, show that as the length of the cells is increased, the effects of the boundary condition fade away and infinite medium solution is obtained. Here the infinite medium solution is 12 and is clearly approached as the length of the cells is increased. This was the expected trend and hence the code passes the trend test.

Symmetry testing: This method of testing incorporates checking for solution symmetry which can be tested without prior knowledge of exact solution. We set up a problem which must have a symmetric solution and then execute the problem using our code. A symmetric solution reduces the probability of the code being flawed. This is the first and only case to be tested here. The two other cases generally used, that are not used here, are checking for coordinate invariance and performing a symmetric calculation using a 3D code. (Knupp, 2000)

Test problem: Consider a 5 by 5 by 5 mesh for a system of with l=b=h=2cm for each cell. Let the cell averaged total cross section across the mesh be 0.5 /cm and the cell averaged scattering cross section be 0.2 /cm across the mesh. Let all boundaries be vacuum and the source in all cells be 6 /cc/s. The resultant plot must be symmetric to pass this test

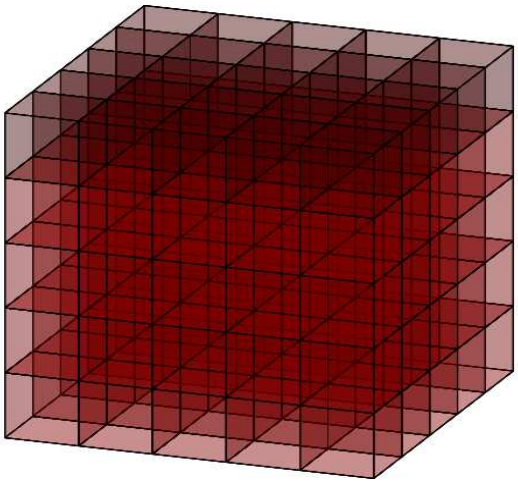


Figure 5

A pictorial representation of the mesh can also be seen in the picture to the left.

The plot below, for z=4.5, is symmetric.

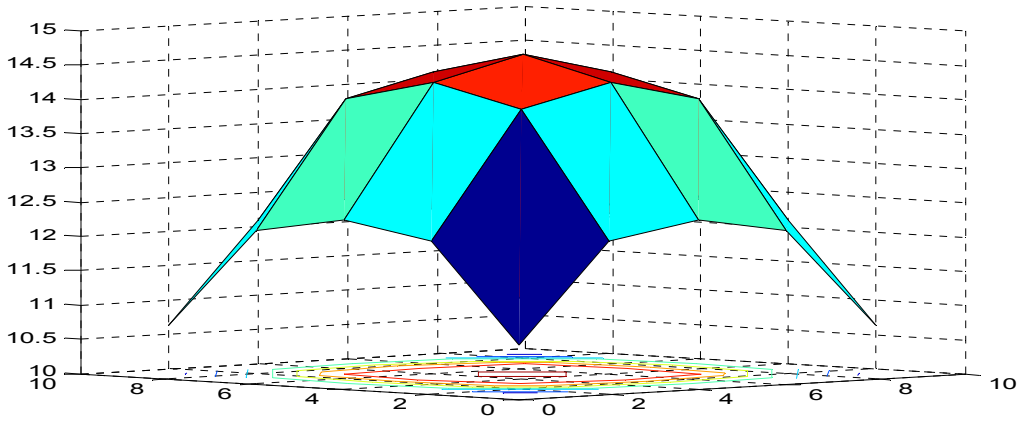


Figure 6

All slices through the mesh display the symmetry expected in the solution.

Method of exact solutions: Here, we solve simple problems for which exact solutions are available, and compare numerical and analytic solutions. In particular, we consider an infinite homogeneous medium We model infinite medium by placing reflecting boundaries around the system.

Test problem: Consider a 5*5*5 mesh for a system of with l=b=h=2cm for each cell. Let the cell averaged total cross sections across the mesh be 0.5 /cm and the cell averaged scattering cross section be 0.2 /cm across the mesh. Let all boundaries be reflecting and the source in all cells be that of 6 /cc/s. The resultant plot must be flat to pass this test and the magnitude of flux must be 20.

Figure 7 shows a plot of the solution at z=4.5cm.

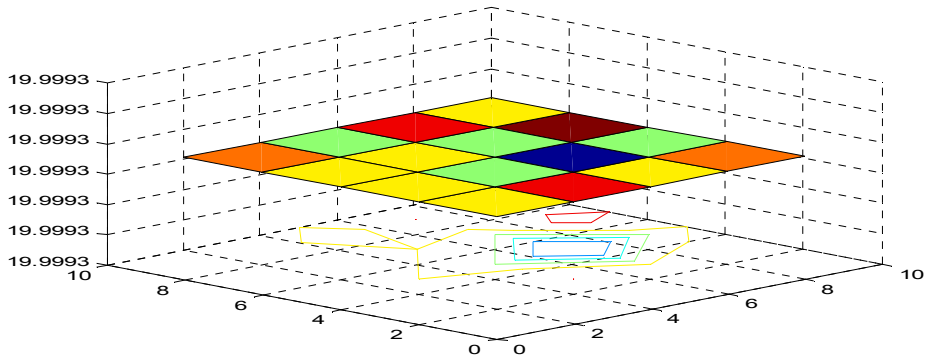


Figure 7

This flat solution of flux = 19.9993 is true throughout the spatial domain.

Comparison Test: Here, the solution obtained from the code being tested is compared with the solution of another existing transport code. The code is considered to have passed the test if the results are significantly close.

The comparison code used here is a slab geometry discrete ordinates-linear characteristic transport code. To compare with this 1D code, the length of the system in one of the three dimensions is made considerably longer than that the other two dimensions and the faces along both of those dimensions are made reflecting.

Test problem: 3 material/region system

1st region:

- Cell averaged total macroscopic cross section = 1.5 /cm; cell averaged macroscopic scattering cross section = 0 /cm
- 15 (by 3 by 3) system where l = b = h = 1 cm
- There is a source of 20 /cc/s in all the cells

2nd region

- Cell averaged total macroscopic cross section = 0.0000001 /cm; cell averaged macroscopic scattering cross section = 0 /cm
- 25 (by 3 by 3) system where l = b = h = 1 cm
- There is a source of 0 /cc/s in all the cells. This is a void region

3rd region:

- Cell averaged total macroscopic cross section = 3 /cm; cell averaged macroscopic scattering cross section = 2 /cm
- 10 (by 3 by 3) system where l = b = h = 1 cm
- There is a source of 10 /cc/s on the right boundary cell/s.

The left and right boundaries are vacuum and all others are reflecting. A pictorial representation of the system is as follows:

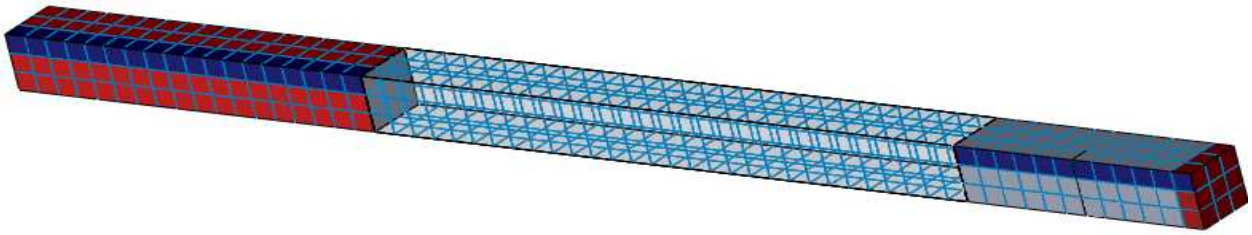


Figure 8

The red cells represent source containing cells. The dark blue cells and those between them are the cells for which the solution is plotted.

The SP_N solution in the x, y, and z directions are shown with the LC solution in figure 9.

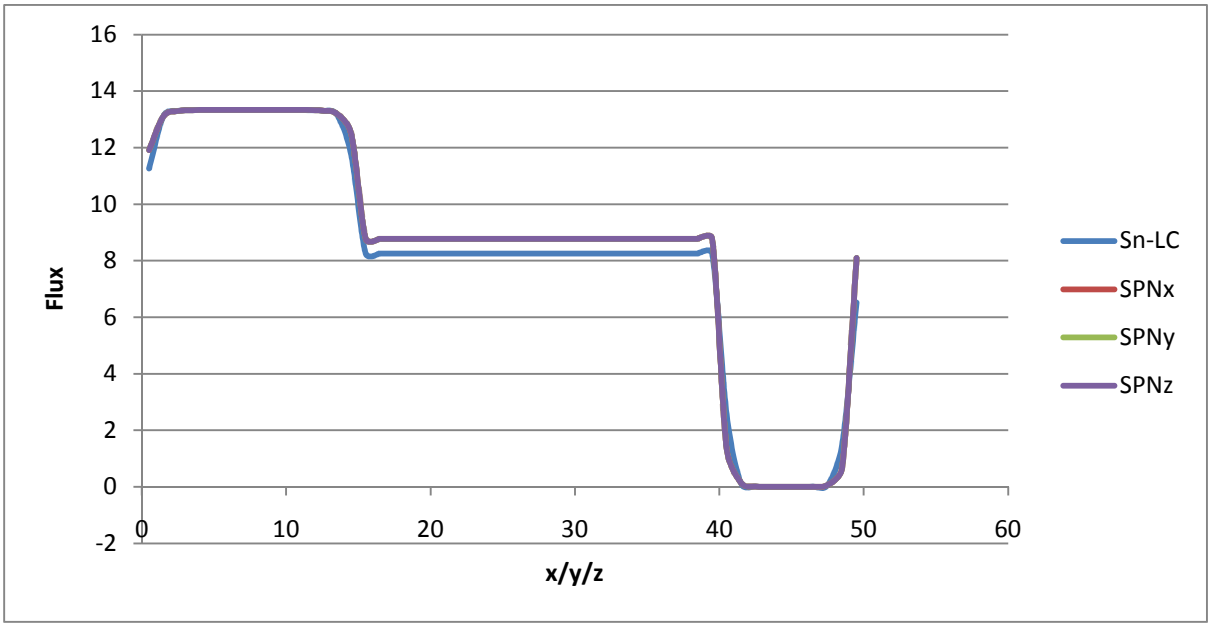


Figure 9

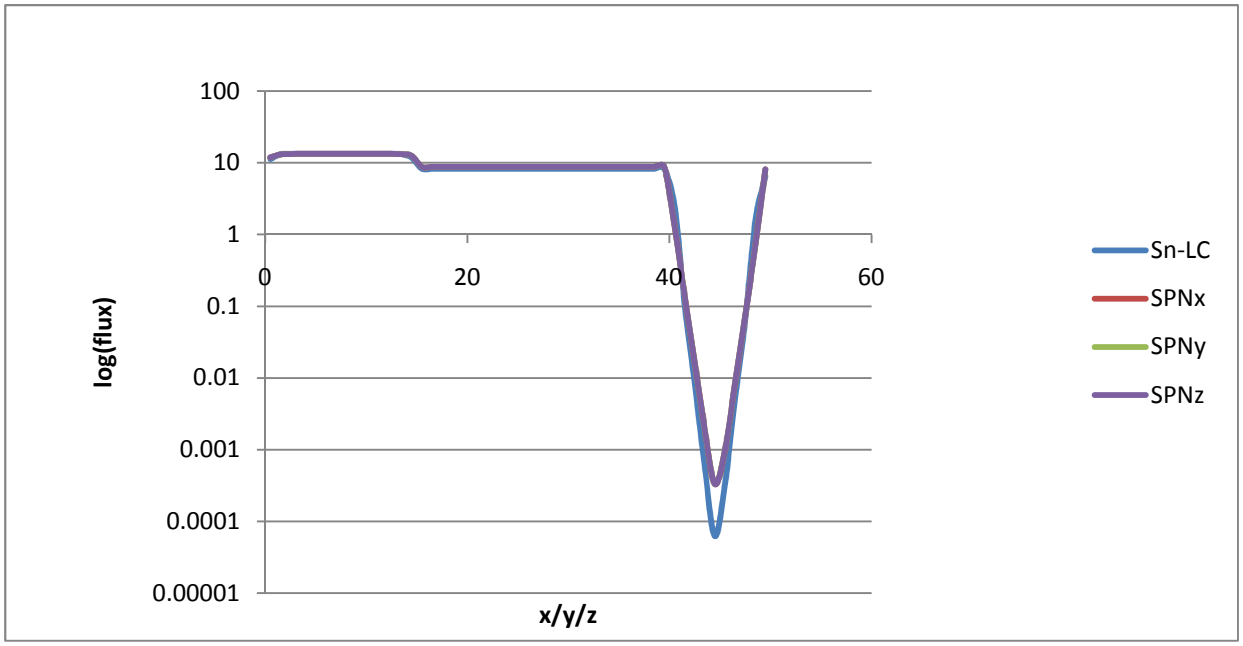


Figure 10

Figure 9 above is the regular solution plot and figure 10 is a semilog solution plot. The plots agree fairly well except in the void region. Note: See Appendix G for numerical solution values

The results of these test problems suggest that the code is functioning properly. The next section will assess the behavior of SP_N solution with increasing expansion order on complex 3D problems.

4 Software Solution Assessment - Methodology and Results

The purpose of this thesis is to assess the SP_N solution of 3D transport problems with increasing expansion order. We define a test problem mimicking a simplistic, miniaturized urban setting with suspended fixed aerial source. Figure 11 shows this geometry.

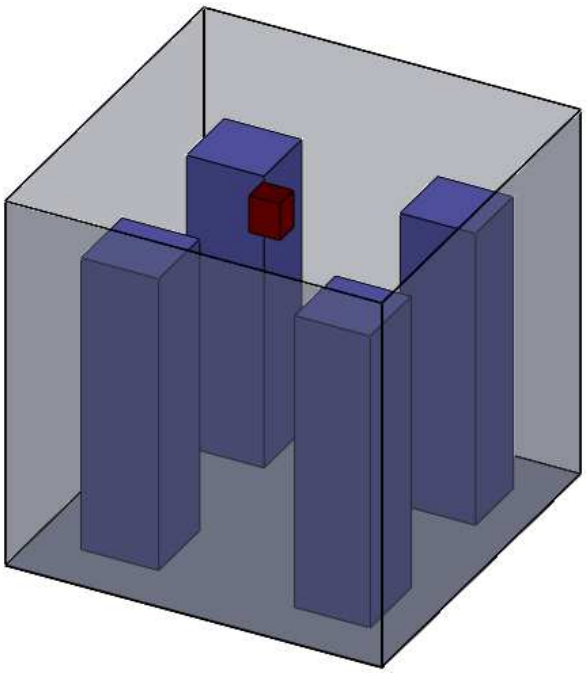


Figure 11

In this problem, l=b=h=1cm. For the extruded blue structures, σ = 0.55 /cm; σ_s = 0.35 /cm; for rest of the system σ = 0.1 /cm; σ_s = 0.03 /cm. Source= 3000 /cc/s in the elevated red region. There are vacuum boundary conditions on all external faces, and the grid is 10 by 10 by 10. Each of these extruded structures has dimensions - 2 cm by 2 cm by 8 cm and are located at a diagonal distance of 1.41 cm from the respective nearest corners on the bottom face of the system. The source, 1 cm by 1cm by 1cm

cell, is at a height of 7 cm from the system base and is at a distance of 4 cm from both left and front faces of the system.

To obtain a benchmark flux distribution in the above system, the problem was simulated in MCNP5 to obtain flux distribution. See Appendix G for the MCNP deck.

Figure 12 shows an SP14 flux profile for the slice at z = 9.5 cm of the test system with respect to x and y. The profile shows a peak at (4.5, 4.5, 9.5) directly above the source and reduces exponentially in the low cross section region (which for this slice is the entire slice domain) as expected. The average error in the solution for this slice with respect to MCNP solution about 4%.

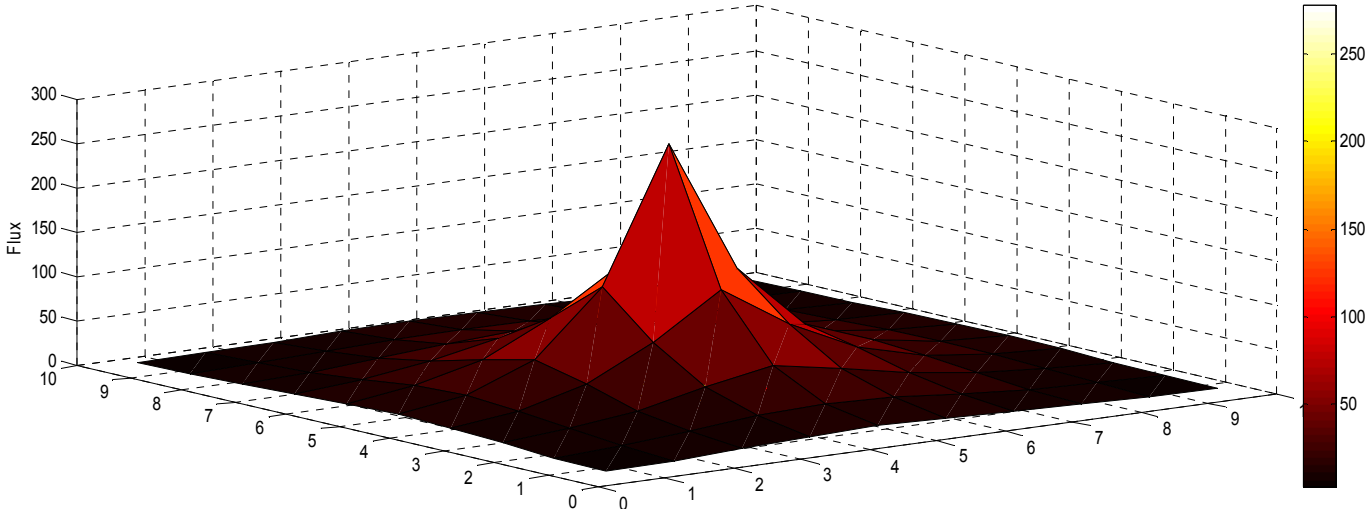


Figure 12

The flux distribution for different slices is different. A complete set of flux distribution plots from the SP_{12} simulation has been provided in appendix J.

The average relative error obtained for various slices of the system (z = 0.5, 1.5, 2.5, 3.5, 4.5, 5.5, 6.5, 7.5, 8.5, and 9.5 cm) is plotted in figure 13.

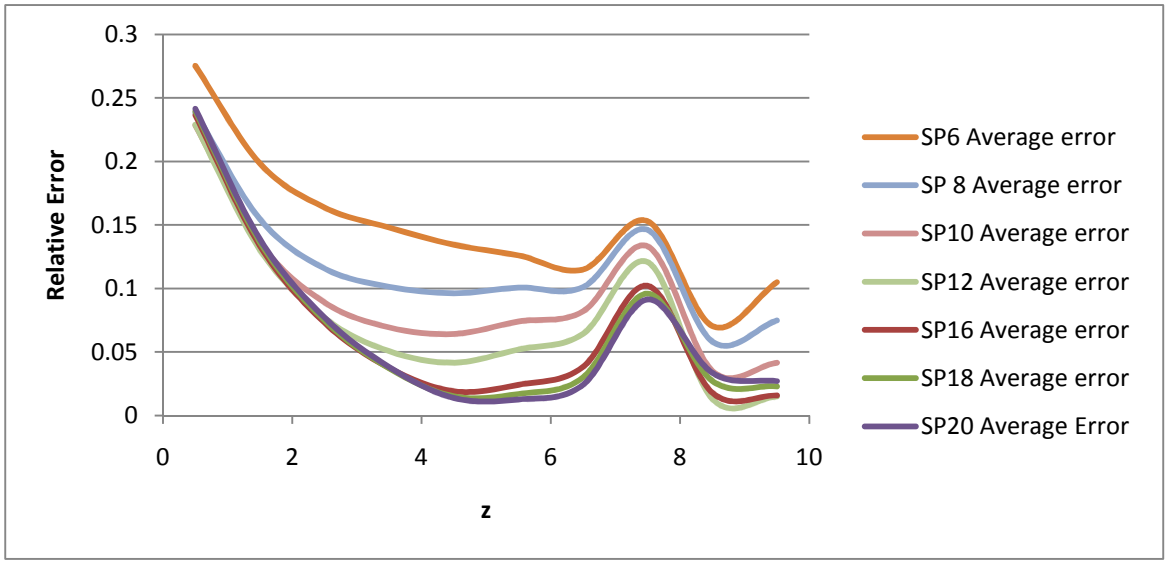


Figure 13

Note: See Appendix I for numerical data.

The following plot represents average error through the system as a function of expansion order.

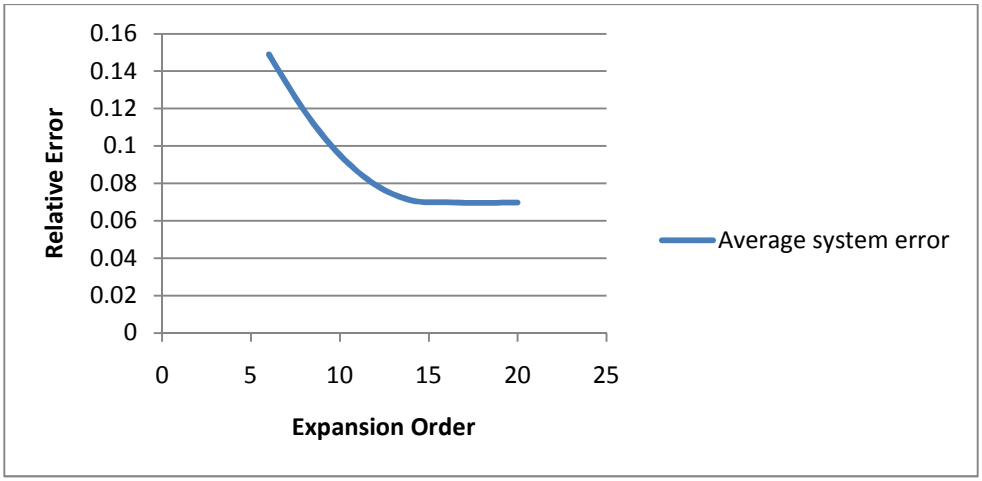


Figure 14

The plots presented in this section indicate that as the expansion order is increased, the accuracy of the solution increases.

A few discrepancies are, however, seen in the data set.

- The average error for the first slice is ~22% and does not decrease significantly with increasing expansion order.
- The average error in the eighth slice containing slice containing the source remains high with increasing expansion order.
- The error stops decreasing with increasing expansion after SP_{14}.

The magnitude of average error in the z = 0.5 cm and z =7.5 cm slices have a common underlying trend - they are relatively well behaved everywhere except when the cells are on and around the corners of the system. This may be a result of sudden cross section change as particles go from relatively thicker (of extruded structures) to thinner air material in the system. There is likely not enough spatial resolution to capture these gradients in the solution. A refinement of the mesh is necessary but due to computational constraints, an extensive study of mesh refinement and resulting improvement in the quality of solutions cannot be presented here.

Note: SP_{12} errors for the slice at z = 0.5 cm has been tabulated and plotted in Appendix J.

An analysis of the data shows that the error for the 10 by 10 by 10 mesh, stops changing rapidly once the expansion order hits 14 . This indicates that the solutions of the given system are starting to converge in the angular variable. Figure 13 shows that the errors converge to 0.3% as the expansion order goes from 16 to 18.

Conclusion

We have developed a capability for solving the energy independent, steady state transport equation using an SP_N-FVM discretization transport code which was tested using various methods. We tested the software on a variety of problems and compared the solution of a complicated three dimensional problem against a benchmark solution from an MCNP5 simulation.

Solution quality improved drastically with increasing angular expansion order. The error of the solution relative to the MCNP solution did not decrease for the cells on and around the corners of the given system. This is attributed to material interfaces around the corners and poor spatial resolution. Mesh refinement is required but due to lack of computational resources (wasteful coding, actually), a finer mesh was not possible.

Solutions from the SP_N method compare well with solution from an industry standard code - MCNP5, for most parts of the system since the system-average error is under 10% for all expansion orders above 10 in spite of large errors in some of the cells. The error, as expected, decreased with incremental expansion order. Increasing the expansion order has no significant impact on the errors introduced by the coarseness of the spatial mesh.

Appendix A - Gauss Legendre Quadrature Sets

Order	Weights (±)	Abscissa (±)
2	1	0.5773502691
4	0.6521451549 0.3478548451	0.3399810435 0.8611363115
6	0.4679139346 0.3607615730 0.1713244924	0.2386191860 0.6612093864 0.9324695142
8	0.3626837834 0.3137066459 0.2223810344 0.1012285363	0.1834346424 0.5255324099 0.7966664774 0.9602898564
10	0.2955242247 0.2692667193 0.2190863625 0.1494513492 0.0666713443	0.1488743389 0.4333953941 0.6794095682 0.8650633663 0.9739065285
12	0.2491470458 0.2334925365 0.2031674267 0.1600783286 0.1069393260 0.0471753364	0.1252334085 0.3678314989 0.5873179542 0.7699026741 0.9041172563 0.9815606342
14	0.2152638534632 0.2051984637213 0.1855383974779 0.1572031671582 0.1215185706879 0.08015808715976 0.03511946033175	0.1080549487073 0.3191123689279 0.5152486363582 0.6872929048117 0.8272013150698 0.9284348836636 0.9862838086968

	Weights (±)	Abscissa (±)
16	0.1894506104551 0.1826034150449 0.1691565193950 0.1495959888166 0.1246289712555 0.09515851168249 0.06225352393865 0.02715245941175	0.09501250983764 0.2816035507793 0.4580167776572 0.6178762444026 0.7554044083550 0.8656312023878 0.9445750230732 0.9894009349916
18	0.1691423829631 0.1642764837458 0.1546846751263 0.1406429146707 0.1225552067115 0.1009420441063 0.07642573025489 0.04971454889497 0.02161601352648	0.08477501304174 0.2518862256915 0.4117511614628 0.5597708310739 0.6916870430604 0.8037049589725 0.8926024664976 0.9558239495714 0.9915651684209
20	0.1527533871307 0.1491729864726 0.1420961093184 0.1316886384492 0.1181945319615 0.1019301198172 0.08327674157670 0.06267204833411 0.04060142980039 0.01761400713915	0.07652652113350 0.2277858511416 0.3737060887154 0.5108670019508 0.6360536807265 0.7463319064602 0.8391169718222 0.9122344282513 0.9639719272779 0.9931285991851

Appendix B - Derivation of the Steady State, Mono-Energetic Transport Equation

Particle conservation in phase space dictates that, the rate of change = production rate - loss rate. Under steady state, production rate - loss rate = 0.

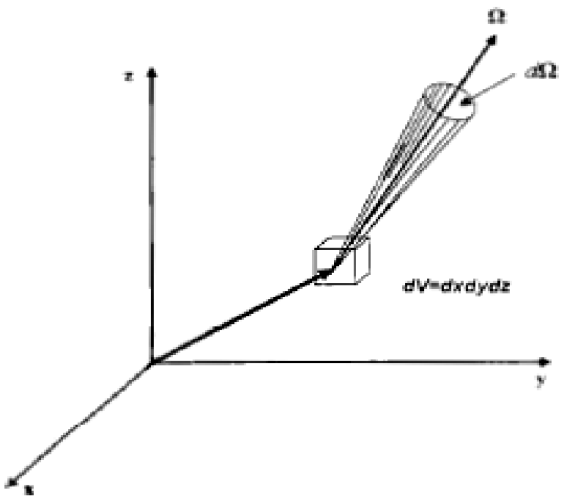
Particle production: Particles are provided from the following sources

- Fixed source
- In volume creation - fission, thermal emission etc. (not taken into account here, since target problem set doesn't include such events)
- Streaming in
- In-leakage in momentum space (not considered here due to assumption of mono-energetic source)
- Out scatter

Particle Loss:

- Out leakage
- Absorption in the background medium
- Out leakage in momentum space (not taken into account due to assumption of mono-energetic source)
- Decay (not taken into account due to the assumption of steady state)
- Out scatter

Consider neutral particles located in an incremental volume dV located at **r** travelling along the direction d**Ω** about direction **Ω** as indicated in the figure below. Here dV = dxdydz



(Bell and Glasstone, 1970)

The polar angle is represented by θ and the azimuthal angle by φ, then $d\boldsymbol{\Omega} = \sin\theta\, d\theta\, d\varphi$ = the solid angle around θ and φ. Also if $\mu = \cos\theta$, $d\boldsymbol{\Omega} = d\varphi\, d\mu = \sin\theta\, d\theta\, d\varphi$.

Suppose N(**r, Ω**) is the number of neutrons at **r** travelling in the direction **Ω**. To find the flux, we integrate N(**r, Ω**) over all possible angles and multiply by the neutron speed v.

That is, Φ(**r**) = v over 4π. If the quantity is not integrated, it is the angular flux, Ψ. The current vector, **J** = v .

Now, if **n** is the normal to the relevant surface, then $\mathbf{n}.\mathbf{J}(\mathbf{r})$ = v is the net number of neutrons crossing the surface in positive direction per second per cm^2.

To derive the transport equation, employ a cylindrical volume (replacing the cubic volume) in the above figure. The base of such an element would be at **r** and its axis would be parallel to **Ω**, the direction of travel. Let the height be Δu and cross sectional area be ΔA. The volume therefore is $\Delta V = \Delta u \Delta A$. The balance equation for neutrons traveling in the **Ω** direction is:

Number of neutrons leaving from right - number of neutrons leaving from the left = number of neutrons scattering in the direction **Ω** in volume ΔV + total number of neutrons emitted in the phase space volume - number of collisions in ΔV per unit time (that remove the particles from given phase space)

Number of neutrons leaving from the right per second: If v = velocity, $\mathrm{N}(\mathbf{r}+\Delta\mathrm{u}, \mathbf{\Omega})$ is the neutron density at right face of cylindrical volume. If **Ω** is the direction of travel and ΔA is the cross sectional area,

then the number of neutrons through the face = $\mathrm{vN}(\mathbf{r}+\Delta\mathrm{u}, \mathbf{\Omega})\ \Delta\mathrm{A}$

Similarly, **number of neutrons leaving from the left** = $\mathrm{vN}(\mathbf{r}, \mathbf{\Omega})\ \Delta\mathrm{A}$

Number of collisions in the phase space volume: flux at **r** in direction **Ω** times the probability of collision times the volume of the imaginary cylinder = $\sigma_t(\mathbf{r})\ \mathrm{vN}(\mathbf{r}, \mathbf{\Omega})\ \Delta\mathrm{V}$, where $\sigma_t(\mathbf{r})$ is the macroscopic total cross section.

Total in scatter into the phase space: Assuming isotropic scattering, the total number of neutrons in any direction per unit time = $\Delta\mathrm{V}\ \mathrm{c(r)}\sigma_t(\mathbf{r})\ \Phi(\mathbf{r})/\ 4\pi$. Where c(**r**) is the number of neutrons produced (scattered into phase space) by collision at **r**.

Total number neutrons generated in the phase space volume = $\Delta\mathrm{V}\ \mathrm{Q}(\mathbf{r}, \Omega)$

Now substitute the equations in balance equation and simplify it to obtain the following form:

$$\frac{\mathrm{vN}(\mathbf{r}+\Delta\mathrm{u}, \Omega)\ -\ \mathrm{vN}(\mathbf{r}, \Omega)}{\Delta\mathrm{u}} = \mathrm{Q}(\mathbf{r}, \Omega) + \frac{\mathrm{c(r)}\sigma_t(\mathbf{r})\Phi(\mathbf{r})}{4\pi} - \sigma_t(\mathbf{r})\ \mathrm{vN}(\mathbf{r}, \Omega)$$

As limit $\Delta u \rightarrow 0$, the balance equation becomes:

$$\frac{d\mathrm{vN}(\mathbf{r},\Omega)}{du} = \mathrm{Q}(\mathbf{r},\Omega) + \frac{\mathrm{c(r)}\sigma_t(\mathbf{r})\Phi(\mathbf{r})}{4\pi} - \sigma_t(\mathbf{r})\,\mathrm{vN}(\mathbf{r},\Omega)$$

But we know that,

$$\frac{d}{du} = \frac{\partial}{\partial x}\frac{dx}{du} + \frac{\partial}{\partial y}\frac{dy}{du} + \frac{\partial}{\partial z}\frac{dz}{du}$$

which is equal to $\boldsymbol{\Omega}.\boldsymbol{\nabla}$. The equation may be written as follows:

$$\boldsymbol{\Omega}.\boldsymbol{\nabla}\mathrm{vN}(\mathbf{r},\Omega) + \sigma_t(\mathbf{r})\mathrm{vN}(\mathbf{r},\Omega) = \mathrm{Q}(\mathbf{r},\Omega) + \frac{\mathrm{c(r)}\sigma_t(\mathbf{r})\Phi(\mathbf{r})}{4\pi}$$

The above equation may be rewritten as follows (eliminating the isotropic scatter assumption):

$$\Omega.\nabla\mathrm{vN}(\mathrm{r},\Omega) + \sigma_t(r)\mathrm{vN}(\mathrm{r},\Omega) = \mathrm{Q}(\mathrm{r},\Omega) + \sigma_t(r)\mathrm{c(r)}\int_0^{4\pi}\mathrm{f}(\Omega'\rightarrow\Omega)\mathrm{vN}(\mathrm{r},\Omega')\mathrm{d}\Omega'$$

Where $\mathrm{c(r)f}(\Omega'\rightarrow\Omega)$ is the function representing the angular distribution of neutrons emerging from collisions at **r**. The function $\mathrm{f}(\Omega'\rightarrow\Omega)$ is normalized to unity -

$$\int \mathrm{f}(\Omega'\rightarrow\Omega)\,\mathrm{d}\Omega = 1$$

(Bell and Glasstone, 1970)

and c(**r**) is the mean number of neutrons emerging from collisions at **r**. (Bell and Glasstone, 1970)

Now substitute $\Psi(\mathbf{r},\Omega)$, the angular flux for $\mathrm{vN}(\mathbf{r},\Omega)$ in the above equation to obtain desired form of the transport equation.

The equation is:

$$\boldsymbol{\Omega} . \boldsymbol{\nabla}\Psi(\boldsymbol{r},\Omega) + \sigma_t(r)\Psi(\boldsymbol{r},\Omega) = \ Q(\boldsymbol{r},\Omega) + \ \sigma_t(r)c(r)\int_0^{4\pi} f(\Omega' \rightarrow \Omega)\Psi(\boldsymbol{r},\Omega')d\Omega'$$

(Bell and Glasstone, 1970)

Appendix C - Derivation of SP_N equations

The following information has been taken from the book Nuclear Reactor theory by Bell and Glasstone.

The final equation derived in Appendix C may be written as follows for plane geometry:

$$\mu\frac{\partial\Psi(x,\mu)}{\partial x}+\sigma_t(x)\Psi(x,\mu)=\ Q(x,\mu)+c(x)\sigma_t(x)\int_0^{2\pi}d\varphi'\ \int_{-1}^{1}f(\Omega'\rightarrow\Omega)\Psi(x,\mu')d\mu'$$

where x is the mean free path.

Here the function $f(\Omega'\rightarrow\Omega)$ is expanded using the Legendre polynomials, i.e.

$$f(\Omega'\rightarrow\Omega)=f(\mu\circ)=\sum_{l=0}^{\infty}\frac{2l+1}{4\pi}f_lP_l(\mu_0)$$

By orthogonality of these polynomials,

$$f_l = 2\pi\int_{-1}^{1}f(\mu_0)P_l(\mu_0)\,d\mu_0$$

with normalization condition

$$f_0=2\pi\int_{-1}^{1}f(\mu_0)\,d\mu_0=1$$

Now according to the addition theorem of Legendre polynomials,

$$P_l\ (\mu_0)=P_l\ (\mu)P_l(\mu')+2\sum_{m=1}^{l}\frac{(l-m)!}{(l+m)!}P_l^m\ (\mu)\ P_l^m\ (\mu')cosm(\varphi-\varphi')$$

Here, $P_l^m\ (\mu)$ are the associated Legendre polynomials.

Now substitute the above relation into the equation for $\mathrm{f}(\Omega' \rightarrow \Omega)$ and then substitute the resultant relation into the transport equation to obtain the following form of the transport equation:

$$\mu \frac{\partial \Psi(x,\mu)}{\partial x} + \Psi(x,\mu) = \ Q(x,\mu) + \frac{c}{2} \sum_{l=0}^{\infty} (2l+1)\, f_l P_l(\mu) \int_{-1}^{1} \Psi(x,\mu') P_l(\mu') d\mu'$$

The source and the angular flux can now be expanded in Legendre polynomials to obtain the following equations:

$$\Psi(\boldsymbol{x},\mu) = \sum_{\mathrm{l}=0}^{\infty} \frac{2\mathrm{l}+1}{4\pi} \Phi_m(\mathrm{x}) \mathrm{P}_m(\mu)$$

and

$$Q(\boldsymbol{x},\mu) = \sum_{\mathrm{l}=0}^{\infty} \frac{2\mathrm{l}+1}{4\pi} \mathrm{Q}_m(\mathrm{x}) \mathrm{P}_m(\mu)$$

Now, according to orthogonality of Legendre polynomials,

$$\Phi_m(\mathrm{x}) = \int \Psi(x,\mu) \mathrm{P}_m(\mu) \mathrm{d}\,\Omega = \ 2\pi \int_{-1}^{1} \Psi(\boldsymbol{x},\mu) \mathrm{P}_m(\mu) \mathrm{d}\,\mu$$

Similarly,

$$Q(\boldsymbol{x},\mu) = 2\pi \int_{-1}^{1} \Psi(\boldsymbol{x},\mu) \mathrm{P}_m(\mu) \mathrm{d}\,\mu$$

Now, upon using the recurrence relation of Legendre polynomials after substitute the above two expansions into the latest form of the transport equation, one obtains the following form of the transport equation:

$$\sum_{l=0}^{\infty}[\frac{\mathrm{d}\Phi(\mathrm{x})}{\mathrm{dx}}\big((\mathrm{m}+1)\mathrm{P}_{m+1}\,(\mu)+\,\mathrm{m}\mathrm{P}_{m-1}(\mu)\big)+\,(2\mathrm{m}+1)\Phi_m(\mathrm{x})\mathrm{P}_m(\mu)]$$

$$=\mathrm{c}\sum_{l=0}^{\infty}(2l+1)\,\Phi_l f_l P_l(\mu)+\,\sum_{m=0}^{\infty}(2m+1)\mathrm{Q}_m\,\mathrm{P}_m(\mu)$$

Now, multiply both sides by $0.5*(2n+1)\,Pn(\mu)$ and integrate over μ from -1 to 1 to obtain the following equation using orthogonality of Legendre polynomials:

$$(\mathrm{n}+1)\,\frac{\mathrm{d}\Phi_{\mathrm{n}+1}(\mathrm{x})}{\mathrm{dx}}+(\mathrm{n})\frac{\mathrm{d}\Phi_{\mathrm{n}-1}(\mathrm{x})}{\mathrm{dx}}+(2\mathrm{n}+1)(1-\mathrm{cf}_{\mathrm{n}})\Phi_{\mathrm{n}}(\mathrm{x})=(2\mathrm{n}+1)\mathrm{Q}_{\mathrm{n}}(\mathrm{x})$$

where n = 0, 1, 2 , 3... (Bell and Glasstone, 1970).

Now divide the equation by 2n+1 and assume isotropic scatter to eliminate the cf_{n} term in the above equation. Also, substitute a spatial variable, z, for mean free path, x, in the above equation, ($\frac{\partial}{\partial r}=\sigma(z)\frac{\partial}{\partial x}$). Carrying out these steps returns the following form of the transport equation:

$$\frac{n+1}{2n+1}\frac{d}{dz}\Phi_{n+1}(z)+\frac{n}{2n+1}\frac{d}{dz}\Phi_{n-1}(z)+\sigma(\mathrm{z})\Phi_n(z)=Q_n(z)$$

Now let n=2n to obtain the following form of the transport equation:

$$\frac{2n+1}{4n+1}\frac{d}{dz}\Phi_{2n+1}(z)+\frac{2n}{4n+1}\frac{d}{dz}\Phi_{2n-1}(z)+\sigma(\mathrm{r})\Phi_{2n}(z)=q_0(z)\delta_{n0}$$

For n = n+1,

$$\frac{2n+2}{4n+3}\frac{d}{dz}\Phi_{2n+2}(z)+\frac{2n+1}{4n+3}\frac{d}{dz}\Phi_{2n}(z)+\sigma(\mathrm{r})\Phi_{2n+1}(z)=0$$

Now obtain the 3D SPn equations by replacing $\frac{d}{dz}$ by spatial gradient for even flux moments and divergence for odd flux moments. Also, replace the spatial dimension z with position r to obtain the SP_N angular discretization equations:

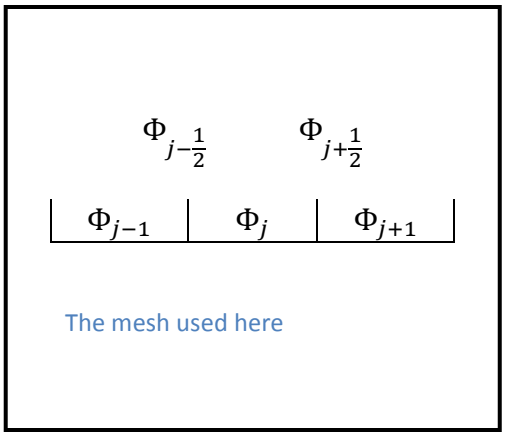
$$\boxed{\frac{2n+1}{4n+1}\nabla.\Phi_{2n+1}(\boldsymbol{r})+\frac{2n}{4n+1}\nabla.\Phi_{2n-1}(\boldsymbol{r})+\sigma(\mathrm{r})\Phi_{2n}(\boldsymbol{r})=q_0(\boldsymbol{r})\delta_{n0}}$$

$$\boxed{\frac{2n+2}{4n+3}\nabla.\Phi_{2n+2}(\boldsymbol{r})+\frac{2n+1}{4n+3}\nabla.\Phi_{2n}(\boldsymbol{r})+\sigma(\mathrm{r})\Phi_{2n+1}(\boldsymbol{r})=0}$$

Appendix D - Derivation of Finite Volume Spatial discretization

The final version of the SP_N equations (in section 2.1) like the diffusion equation.

Here, we derive a spatial discretization of the diffusion equation and then substitute the derived results into the SP_N equation.



Diffusion equation: $\frac{-d}{dx}(D\frac{d\phi}{dx}) + \sigma_a\phi = S$

Source: $\int_{x_{j-1/2}}^{x_{j+1/2}} dx S_j = S_j \Delta x_j$ for the j[th] cell

Absorption term: $\sigma_j \int_{x_{j-1/2}}^{x_{j+1/2}} dx \phi_j = \sigma_{aj}\phi_j \Delta x_j$

Leakage term: $\int \frac{-d}{dx} D\frac{d\phi}{dx} = -D\frac{d\phi}{dx}$ at $x_{j+1/2} + D\frac{d\phi}{dx}$ at $x_{j-1/2}$

Assume $-D\frac{d\phi}{dx}$ at $x_{j+1/2} = -D_j\frac{\phi_{j+1/2}-\phi_j}{\Delta x_{j/2}} = -D_{j+1}\frac{\phi_{j+1}-\phi_{j+1/2}}{\Delta x_{j+1/2}}$

Solution of the above equation results in the following equation:

$$-D\frac{d\phi}{dx} \text{ at } x_{j+1/2} = -2D_jD_{j+1}\frac{\phi_{j+1}-\phi_j}{D_{j+1}\Delta x_j + D_j\Delta x_{j+1}}$$

Similarly, an equation can also be obtained for $-D\frac{d\phi}{dx}$ at $x_{j-1/2}$. Once the gradient values are known, a discrete equation set called the difference equation set can be formulated to yields a linear system of equations that can be solved using digital computers.

The 3D case:

Equation: $-\nabla . D(r)\nabla\Phi + \sigma_a\phi = S$

Source: $\sigma_a \int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} \int_{y_{j-\frac{1}{2}}}^{y_{j+\frac{1}{2}}} \int_{z_{k-\frac{1}{2}}}^{z_{k+\frac{1}{2}}} dxdydz S_{ijk} = S_{ijk}\Delta x\Delta y\Delta z$

Absorption term: $\sigma_a \int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} \int_{y_{j-\frac{1}{2}}}^{y_{j+\frac{1}{2}}} \int_{z_{k-\frac{1}{2}}}^{z_{k+\frac{1}{2}}} dxdydz\Phi_{ijk} = \sigma_a\Phi_{ijk}\Delta x\Delta y\Delta z$

Leakage term: $-\nabla . D(r)\nabla\Phi = -\nabla . D(r)(\frac{\partial\phi}{\partial x}\boldsymbol{e_x} + \frac{\partial\phi}{\partial y}\boldsymbol{e_y} + \frac{\partial\phi}{\partial z}\boldsymbol{e_y})$

$= -\frac{\partial}{\partial x}(D(r)\frac{\partial\phi}{\partial x}) + \frac{\partial}{\partial y}(D(r)\frac{\partial\phi}{\partial y}) + \frac{\partial}{\partial z}D(r)\frac{\partial\phi}{\partial z})$

Upon integration, $\int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} \int_{y_{j-\frac{1}{2}}}^{y_{j+\frac{1}{2}}} \int_{z_{k-\frac{1}{2}}}^{z_{k+\frac{1}{2}}} -(\frac{\partial}{\partial x}(D(r)\frac{\partial\phi}{\partial x}) + \frac{\partial}{\partial y}(D(r)\frac{\partial\phi}{\partial y}) + \frac{\partial}{\partial z}D(r)\frac{\partial\phi}{\partial z})\, dxdydz$

$= [-D\frac{d\phi}{dx}$ at $x_{i+1/2} + D\frac{d\phi}{dx}$ at $x_{i-1/2}]\,\Delta y\Delta z + [-D\frac{d\phi}{dy}$ at $y_{j+1/2} + D\frac{d\phi}{dy}$ at $x_{j-1/2}]\,\Delta x\Delta z +$

$[-D\frac{d\phi}{dz}$ at $z_{j+1/2} + D\frac{d\phi}{dz}$ at $z_{j-1/2}]\,\Delta y\Delta x$

The above equation is discretized piecewise in the exact same manner the equation for 1D case. Such a discretization followed by substitution in the final SP_N renders the transport equation in the following form:

$$\boldsymbol{\Phi_{l-1}}\frac{-2\boldsymbol{D_{l-1}D_{jkl}}\Delta\boldsymbol{x_j}\Delta\boldsymbol{y_k}}{\boldsymbol{D_{jkl}}\Delta\boldsymbol{z_{l-1}} + \boldsymbol{D_{l-1}}\Delta\boldsymbol{z_l}} + \boldsymbol{\Phi_{l+1}}\frac{-2\boldsymbol{D_{l+1}D_{jkl}}\Delta\boldsymbol{x_j}\Delta\boldsymbol{y_k}}{\boldsymbol{D_{jkl}}\Delta\boldsymbol{z_{l+1}} + \boldsymbol{D_{l+1}}\Delta\boldsymbol{z_l}} + \boldsymbol{\Phi_{k-1}}\frac{-2\boldsymbol{D_{k-1}D_{jkl}}\Delta\boldsymbol{x_j}\Delta\boldsymbol{z_l}}{\boldsymbol{D_{jkl}}\Delta\boldsymbol{y_{k-1}} + \boldsymbol{D_{k-1}}\Delta\boldsymbol{y_k}}$$
$$+ \boldsymbol{\Phi_{k+1}}\frac{-2\boldsymbol{D_{k+1}D_{jkl}}\Delta\boldsymbol{x_j}\Delta\boldsymbol{z_l}}{\boldsymbol{D_{jkl}}\Delta\boldsymbol{y_{k+1}} + \boldsymbol{D_{k+1}}\Delta\boldsymbol{y_k}} + \boldsymbol{\Phi_{j-1}}\frac{-2\boldsymbol{D_{j-1}D_{jkl}}\Delta\boldsymbol{y_k}\Delta\boldsymbol{z_l}}{\boldsymbol{D_{jkl}}\Delta\boldsymbol{x_{j-1}} + \boldsymbol{D_{j-1}}\Delta\boldsymbol{x_j}} + \boldsymbol{\Phi_{j+1}}\frac{-2\boldsymbol{D_{j+1}D_{jkl}}\Delta\boldsymbol{y_k}\Delta\boldsymbol{z_l}}{\boldsymbol{D_{jkl}}\Delta\boldsymbol{x_{j+1}} + \boldsymbol{D_{j+1}}\Delta\boldsymbol{x_j}} - \boldsymbol{\Phi_{jkl}}\Bigg(\frac{-2\boldsymbol{D_{j+1}D_{jkl}}\Delta\boldsymbol{y_k}\Delta\boldsymbol{z_l}}{\boldsymbol{D_{jkl}}\Delta\boldsymbol{x_{j+1}} + \boldsymbol{D_{j+1}}\Delta\boldsymbol{x_j}}$$
$$+ \frac{-2\boldsymbol{D_{j-1}D_{jkl}}\Delta\boldsymbol{y_k}\Delta\boldsymbol{z_l}}{\boldsymbol{D_{jkl}}\Delta\boldsymbol{x_{j-1}} + \boldsymbol{D_{j-1}}\Delta\boldsymbol{x_j}} + \frac{-2\boldsymbol{D_{k+1}D_{jkl}}\Delta\boldsymbol{x_j}\Delta\boldsymbol{z_l}}{\boldsymbol{D_{jkl}}\Delta\boldsymbol{y_{k+1}} + \boldsymbol{D_{k+1}}\Delta\boldsymbol{y_k}} + \frac{-2\boldsymbol{D_{k-1}D_{jkl}}\Delta\boldsymbol{x_j}\Delta\boldsymbol{z_l}}{\boldsymbol{D_{jkl}}\Delta\boldsymbol{y_{k-1}} + \boldsymbol{D_{k-1}}\Delta\boldsymbol{y_k}} + \frac{-2\boldsymbol{D_{l+1}D_{jkl}}\Delta\boldsymbol{x_j}\Delta\boldsymbol{y_k}}{\boldsymbol{D_{jkl}}\Delta\boldsymbol{z_{l+1}} + \boldsymbol{D_{l+1}}\Delta\boldsymbol{z_l}} + \frac{-2\boldsymbol{D_{l-1}D_{jkl}}\Delta\boldsymbol{x_j}\Delta\boldsymbol{y_k}}{\boldsymbol{D_{jkl}}\Delta\boldsymbol{z_{l-1}} + \boldsymbol{D_{l-1}}\Delta\boldsymbol{z_l}}$$
$$+ \boldsymbol{\sigma_{jkl}}\Delta\boldsymbol{x_j}\Delta\boldsymbol{y_k}\Delta\boldsymbol{z_l}\Bigg) = \boldsymbol{\sigma_s}(\boldsymbol{r})\sum_{\boldsymbol{b=1}}^{\boldsymbol{N}}\boldsymbol{\Phi_b w_b} + \boldsymbol{S}(\boldsymbol{r})$$

Where, j, k and l represent x, y and z directions respectively.

The above equation represents flux definition in cells with interior boundaries/interfaces.

Generic short hands used in the above equation:

$$\Phi_{l-1} = \Phi_{j,k,l-1}$$

$$\Phi_{k-1} = \Phi_{l,j,k-1}$$

$$\Phi_{j-1} = \Phi_{k,l,j-1}$$

and so on.

Appendix E - Derivation of Boundary conditions

Reflecting Boundary:

In the case of reflecting boundary, there is zero net flow through the boundary with reflecting condition. Hence $-D\frac{d\phi}{dx}$ at $x_{j\pm0.5}$ = 0. Therefore, $\Phi_i = \Phi_{i_{\pm1}}$

This means that the coefficient term corresponding to the cell on which the reflecting boundary is placed gets extinguished in the respective flux term. The term associated with that boundary gets extinguished in the central flux (Φ_{ijk}) coefficient term as well.

The same holds for fluxes in all directions with reflecting boundary

An example of the equation form with a reflecting boundary on one of the faces has been given in section 2.2

Vacuum Boundary:

Left boundary

We know that, $J_\alpha = -\frac{{\mu_\alpha}^2}{\sigma}\frac{\partial\phi}{\partial x}$. Upon discretization, the equation takes the following form:

$$J_{\alpha,1/2} = -\frac{{\mu_\alpha}^2}{\sigma_1}\frac{\phi_{\alpha,1} - \phi_{\alpha,1/2}}{\Delta x_1/2}$$

Now, at the boundary,

$$\phi_\alpha = -\frac{\mu_\alpha}{\sigma}\frac{\partial\phi}{\partial x}.$$

Upon multiplying the above equation by μ_α and moving the negative sign to the other side, it is observed that $J_{\alpha,1/2} = -\mu_\alpha \phi_{\alpha,1/2}$

Substituting this relation into the discretized equation to derive an equation for $J_{\alpha,1/2}$ yields:

$$J_{\alpha,i/2} = \frac{-2\mu_\alpha^2 \phi_\alpha}{\sigma_i \Delta x_i + 2\mu_\alpha}$$

The balance equation in cell 1 is:

$$J_{\frac{3}{2}} - J_{\frac{1}{2}} + \sigma_{a1} \Delta x_1 \phi_1 = Q \Delta x_1$$

Where $J_{\alpha,\frac{i}{2}}$ from the equation above is substituted for $J_{\frac{1}{2}}$ (here, the boundary current because vacuum boundary is on the left face) in the above equation. $J_{\frac{3}{2}}$ is an on an interior boundary in this case and its behavior is dictated by the equations obtained for interior boundary in Appendix E.

For the right boundary, one derives the equation for $J_{\frac{3}{2}}$ which is the vacuum boundary current while $J_{\frac{1}{2}}$, here, falls on the interior side of the cell. The exact same procedure is followed as in the case of left boundary. One finds that $J_{\frac{3}{2}} = \frac{-2\mu_\alpha^2 \phi_\alpha}{\sigma_i \Delta x_i + 2\mu_\alpha}$ here as well.

The currents along all three directions at the boundary with vacuum boundary condition are given by the same equation. An example of the vacuum boundary condition equation has been given in Section 2.3 along with an explanation of how the discretized transport equation changes as vacuum boundaries are applied.

Appendix F - Solution File for Comparison Test

x/y/z	S_N-LC	SP_Nx	SP_Ny	SP_Nz
0.5	11.26178	11.91042	11.91042	11.91042
1.5	13.12915	13.11404	13.11404	13.11404
2.5	13.29949	13.29331	13.29331	13.29331
3.5	13.32743	13.32544	13.32544	13.32544
4.5	13.3323	13.3317	13.3317	13.3317
5.5	13.33315	13.33298	13.33298	13.33298
6.5	13.3333	13.33325	13.33325	13.33325
7.5	13.33332	13.33331	13.33331	13.33331
8.5	13.33331	13.33328	13.33328	13.33328
9.5	13.33319	13.3331	13.3331	13.3331
10.5	13.33251	13.33226	13.33226	13.33226
11.5	13.32863	13.32819	13.32819	13.32819
12.5	13.30639	13.30765	13.30765	13.30765
13.5	13.17134	13.19608	13.19608	13.19608
14.5	11.73551	12.49134	12.49134	12.49134
15.5	8.256153	8.778136	8.778136	8.778136
16.5	8.256152	8.778136	8.778136	8.778136
17.5	8.256151	8.778135	8.778135	8.778135
18.5	8.25615	8.778135	8.778135	8.778135
19.5	8.256149	8.778134	8.778134	8.778134
20.5	8.256147	8.778134	8.778134	8.778134
21.5	8.256146	8.778133	8.778133	8.778133
22.5	8.256145	8.778133	8.778133	8.778133
23.5	8.256144	8.778132	8.778132	8.778132
24.5	8.256143	8.778132	8.778132	8.778132
25.5	8.256142	8.778131	8.778131	8.778131
26.5	8.25614	8.778131	8.778131	8.778131
27.5	8.256139	8.77813	8.77813	8.77813
28.5	8.256138	8.77813	8.77813	8.77813
29.5	8.256137	8.778129	8.778129	8.778129
30.5	8.256136	8.778129	8.778129	8.778129
31.5	8.256135	8.778128	8.778128	8.778128
32.5	8.256133	8.778128	8.778128	8.778128
33.5	8.256132	8.778127	8.778127	8.778127
34.5	8.256131	8.778127	8.778127	8.778127
35.5	8.25613	8.778126	8.778126	8.778126
36.5	8.256129	8.778126	8.778126	8.778126
37.5	8.256128	8.778125	8.778125	8.778125

38.5	8.256126	8.778125	8.778125	8.778125
39.5	8.256125	8.778124	8.778124	8.778124
40.5	2.567566	1.352136	1.352136	1.352136
41.5	0.111722	0.146231	0.146231	0.146231
42.5	0.008769	0.016596	0.016596	0.016596
43.5	0.000581	0.001928	0.001928	0.001928
44.5	6.28E-05	0.000334	0.000334	0.000334
45.5	0.000334	0.00098	0.00098	0.00098
46.5	0.005018	0.008273	0.008273	0.008273
47.5	0.063488	0.072902	0.072902	0.072902
48.5	1.487716	0.674555	0.674555	0.674555
49.5	6.517739	8.095034	8.095034	8.095034

Appendix G - MCNP Input Deck

```
c     Created on: Thursday, May 26, 2011 at 10:15
1     2  1.0  4 -5 14 -15 27 -28  imp:n=1 $Source
2     1  1.0  1 -3 11 -13 33 -28  imp:n=1
3     1  1.0  7 -9 11 -13 33 -28  imp:n=1
4     1  1.0  1 -3 17 -19 33 -28  imp:n=1
5     1  1.0  7 -9 17 -19 33 -28  imp:n=1
6     2  1.0  31 -10 32 -20 33 -30 #1 #2 #3 #4 #5 imp:n=1
7     0       10:20:30:-31:-32:-33  imp:n=0

1     px 1
c 2     px 2
3     px 3
4     px 4
5     px 5
c 6     px 6
7     px 7
c 8     px 8
9     px 9
10    px 10
11    py 1
c 12    py 2
13    py 3
14    py 4
15    py 5
c 16    py 6
17    py 7
c 18    py 8
19    py 9
20    py 10
c 21    pz 1
c 22    pz 2
c 23    pz 3
c 24    pz 4
c 25    pz 5
c 26    pz 6
27    pz 7
28    pz 8
c 29    pz 9
```

```
30    pz 10
31    px 0.001
32    py 0.001
33    pz 0.001

c     Material Cards
xs1   92250.12m 1.0 concrete 0 1 1 11 0 0 0.0
m1    92250.12m 1.0
xs2   92250.23m 1.0 dair 0 1 1 11 0 0 0.0
m2    92250.23m 1.0
mgopt f 1
c     Source
sdef   x=d1 y=d2 z=d3
si1    4.0001 4.9999
sp1    0 1
si2    4.0001 4.9999
sp2    0 1
si3    7.0001 7.9999
sp3    0 1
c     Tally cards
FMESH4:n geom=xyz imesh=10 iints=10 jmesh=10 jints=10 kmesh=10 kints=10
        out=ij origin=0.001 0.001 0.001
nps    400000
```

Appendix H - Generic Code Block Used in the SP_N-FVM Code

The following piece of code is the engine of the SP_N-FVM transport code used in this thesis. It is the building block that gets repeated at each step of execution:

```
for ordct=1:ord
    if ordct==1
        D=(X.^-1).*(Wa(ordct)*Wa(ordct));  %%diffusion coefficient -
destroyed and recreated with every iteration of the greater for loop
for count=1:i*j*k
    row=zeros(1,i*j*k);
    if any(count==U)==0 && any(count==Do)==0 %%entry in row vector along z
        row(count-i*j)= -D(count)*D(count-
i*j)*x(count)*y(count)*2/((D(count)*z(count-i*j))+(D(count-i*j)*z(count)));
        row(count+i*j)= -
D(count)*D(count+i*j)*x(count)*y(count)*2/((D(count+i*j)*z(count))+(D(count)*
z(count+i*j)));
        row(count)= (D(count)*D(count-
i*j)*x(count)*y(count)*2/((D(count)*z(count-i*j))+(D(count-
i*j)*z(count))))+(D(count)*D(count+i*j)*x(count)*y(count)*2/((D(count+i*j)*z(
count))+(D(count)*z(count+i*j))));
    elseif any(count==U)==1
        row(count+i*j)= -
D(count)*D(count+i*j)*x(count)*y(count)*2/((D(count+i*j)*z(count))+(D(count)*
z(count+i*j)));
        if BC(5)==0 %%reflected top boundary
            row(count)=
D(count)*D(count+i*j)*x(count)*y(count)*2/((D(count+i*j)*z(count))+(D(count)*
z(count+i*j)));
        else %%vacuum top boundary
            row(count)=
((2*Wa(ordct)*Wa(ordct)*x(count)*y(count))/((X(count)*z(count))+(2*Wa(ordct))
))+(D(count)*D(count+i*j)*x(count)*y(count)*2/((D(count+i*j)*z(count))+(D(cou
nt)*z(count+i*j))));
        end
    else
        row(count-i*j)= -D(count)*D(count-
i*j)*x(count)*y(count)*2/((D(count)*z(count-i*j))+(D(count-i*j)*z(count)));
        if BC(6)==0 %%reflected bottom boundary
            row(count)= D(count)*D(count-
i*j)*x(count)*y(count)*2/((D(count)*z(count-i*j))+(D(count-i*j)*z(count)));
        else %%vacuum bottom boundary
            row(count)=
((2*Wa(ordct)*Wa(ordct)*x(count)*y(count))/((X(count)*z(count))+(2*Wa(ordct))
))+(D(count)*D(count-i*j)*x(count)*y(count)*2/((D(count)*z(count-
i*j))+(D(count-i*j)*z(count))));
        end
    end
    if any(count==E)==0 && any(count==W)==0 %%entry along x direction
        row(count-1)= -D(count-
1)*D(count)*y(count)*z(count)*2/((D(count)*x(count-1))+(D(count-
1)*x(count)));
        row(count+1)= -
D(count)*D(count+1)*y(count)*z(count)*2/((D(count+1)*x(count))+(D(count)*x(co
unt+1)));
```

```
        row(count)= row(count)+(D(count-
1)*D(count)*y(count)*z(count)*2/((D(count)*x(count-1))+(D(count-
1)*x(count))))+(D(count)*D(count+1)*y(count)*z(count)*2/((D(count+1)*x(count)
)+(D(count)*x(count+1))));
    elseif any(count==E)==1
        row(count+1)= -
D(count)*D(count+1)*y(count)*z(count)*2/((D(count+1)*x(count))+(D(count)*x(co
unt+1)));
        if BC(1)==0 %%reflected east boundary

row(count)=row(count)+(D(count)*D(count+1)*y(count)*z(count)*2/((D(count+1)*x
(count))+(D(count)*x(count+1))));
        else %%vacuum east boundary

row(count)=row(count)+((2*Wa(ordct)*Wa(ordct)*z(count)*y(count))/((X(count)*x
(count))+(2*Wa(ordct))))+(D(count)*D(count+1)*y(count)*z(count)*2/((D(count+1
)*x(count))+(D(count)*x(count+1))));
        end
    else
         row(count-1)= -D(count-
1)*D(count)*y(count)*z(count)*2/((D(count)*x(count-1))+(D(count-
1)*x(count)));
         if BC(2)==0 %%reflected west boundary
             row(count)=row(count)+(D(count-
1)*D(count)*y(count)*z(count)*2/((D(count)*x(count-1))+(D(count-
1)*x(count))));
         else %%vacuum west boundary

row(count)=row(count)+((2*Wa(ordct)*Wa(ordct)*z(count)*y(count))/((X(count)*x
(count))+(2*Wa(ordct))))+(D(count-
1)*D(count)*y(count)*z(count)*2/((D(count)*x(count-1))+(D(count-
1)*x(count))));
         end
    end
    if any(count==N)==0 && any(count==S)==0 %%entry along y direction
        row(count-i)= -D(count)*D(count-
i)*x(count)*z(count)*2/((D(count)*y(count-i))+(D(count-i)*y(count)));
        row(count+i)= -
D(count)*D(count+i)*x(count)*z(count)*2/((D(count+i)*y(count))+(D(count)*y(co
unt+i)));
        row(count)=row(count)+(D(count)*D(count-
i)*x(count)*z(count)*2/((D(count)*y(count-i))+(D(count-
i)*y(count))))+(D(count)*D(count+i)*x(count)*z(count)*2/((D(count+i)*y(count)
)+(D(count)*y(count+i))));
    elseif any(count==N)==1
        row(count+i)= -
D(count)*D(count+i)*x(count)*z(count)*2/((D(count+i)*y(count))+(D(count)*y(co
unt+i)));
        if BC(3)==0 %%reflected boundary in north

row(count)=row(count)+(D(count)*D(count+i)*x(count)*z(count)*2/((D(count+i)*y
(count))+(D(count)*y(count+i))));
        else %%vacuum boundary in north

row(count)=row(count)+((2*Wa(ordct)*Wa(ordct)*x(count)*z(count))/((X(count)*y
(count))+(2*Wa(ordct))))+(D(count)*D(count+i)*x(count)*z(count)*2/((D(count+i
)*y(count))+(D(count)*y(count+i))));
```

```
        end
    else
        row(count-i)= -D(count)*D(count-
i)*x(count)*z(count)*2/((D(count)*y(count-i))+(D(count-i)*y(count)));
        if BC(4)==0 %%reflected boundary in south
            row(count)=row(count)+(D(count)*D(count-
i)*x(count)*z(count)*2/((D(count)*y(count-i))+(D(count-i)*y(count))));
        else %%vacuum boundary in south

row(count)=row(count)+((2*Wa(ordct)*Wa(ordct)*x(count)*z(count))/((X(count)*y
(count))+(2*Wa(ordct))))+(D(count)*D(count-
i)*x(count)*z(count)*2/((D(count)*y(count-i))+(D(count-i)*y(count))));
        end
    end
        row(count)=row(count)+(X(count)*x(count)*y(count)*z(count));
        count=count+1;
        CM=[CM;row];
end
```

Note: The above piece of code is not the complete code, it is just a part of the SP_N-FVM code.

Appendix I - Error Tables

Z	*SP6 Average error*	*SP 8 Average error*	*SP10 Average error*	*SP12 Average error*
0.5	*0.275225749*	*0.238378191*	*0.228244879*	*0.228855987*
1.5	*0.198378427*	*0.154506316*	*0.135788728*	*0.130426116*
2.5	*0.163711307*	*0.115610888*	*0.088894565*	*0.076695596*
3.5	*0.147954753*	*0.101260245*	*0.069265767*	*0.050515631*
4.5	*0.13447954*	*0.096165926*	*0.064146911*	*0.041529485*
5.5	*0.126072025*	*0.100556759*	*0.073945239*	*0.051919759*
6.5	*0.11494605*	*0.100941315*	*0.081988427*	*0.064278982*
7.5	*0.152916472*	*0.145995822*	*0.133082483*	*0.120677126*
8.5	*0.070468965*	*0.058088336*	*0.035101807*	*0.012968121*
9.5	*0.105022132*	*0.07475521*	*0.041501755*	*0.014870856*

z	*SP14 Average error*	*SP16 Average error*	*SP18 Average error*	*SP20 Average Error*
0.5	*0.232652579*	*0.236522483*	*0.239495917*	*0.241492003*
1.5	*0.131066304*	*0.133718937*	*0.136530429*	*0.138815693*
2.5	*0.07285438*	*0.073111445*	*0.074957689*	*0.077106759*
3.5	*0.041161623*	*0.037585269*	*0.037141544*	*0.038151652*
4.5	*0.027261828*	*0.019185917*	*0.015224264*	*0.013768841*

5.5	*0.03558401*	*0.024338237*	*0.017062329*	*0.012642698*
6.5	*0.04967287*	*0.038414889*	*0.030110376*	*0.024176214*
7.5	*0.110358671*	*0.102252375*	*0.096050315*	*0.091360053*
8.5	*-0.004850428*	*-0.018019237*	*-0.027268135*	*-0.033538236*
9.5	*-0.004850428*	*-0.015841863*	*-0.023054543*	*-0.02702767*

Appendix J - Sample (SP_{12}) Flux and Error Profiles

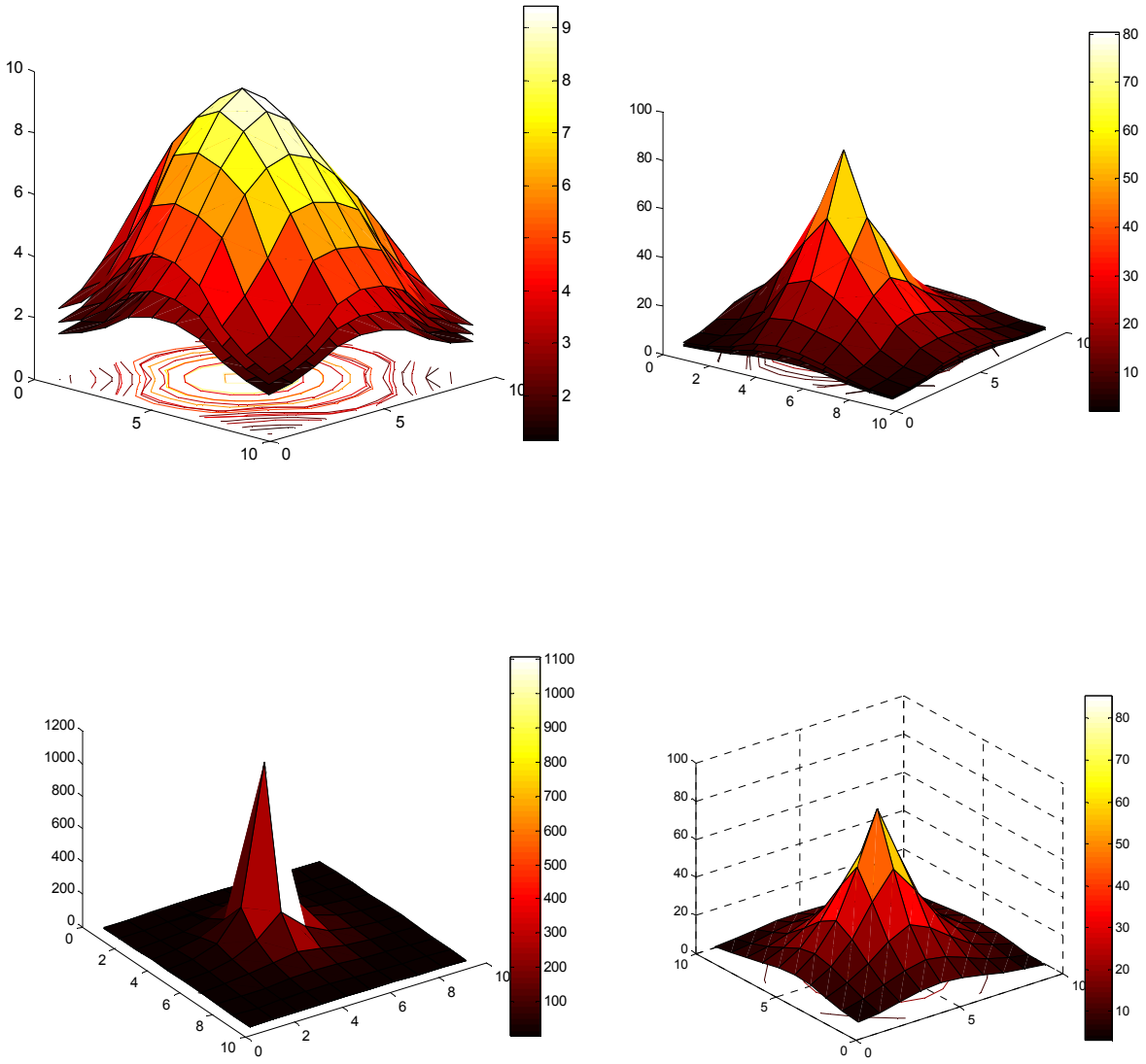


Figure 15

Plots of figure 14 represent flux profiles in various slices of the system. Going left to right, the plots represent flux profiles in the first three slices (z = 0.5, 1.5 and 2.5 cm) the next picture represents that at z = 3.5, 4.5, and 5.5 cm. The third plot represents slices at z = 6.5, 7.5, and 8.5 cm. The last plot represents the top most slice.

Fluxes (x = 0.5 to 4.5 cm) - X - across; Y - down

Slice 05

1.486831855	1.70092835	2.08123917	2.62405	2.8279742
1.700928345	1.83006742	2.37857071	3.2857416	3.42347384
2.081239169	2.37857071	2.98317966	3.8326502	3.94748148
2.62404998	3.2857416	3.83265017	4.1744385	4.3020845
2.827974197	3.42347384	3.94748148	4.3020845	4.43246992
2.787315505	3.36853462	3.86892064	4.194625	4.31211553
2.518999111	3.15009314	3.63555373	3.8856195	3.98020728
1.947120913	2.22899713	2.74442775	3.4151086	3.50329374
1.543249547	1.64055811	2.0941518	2.8396293	2.95174984
1.31200856	1.47163484	1.77818298	2.2349595	2.4053846

Fluxes (x = 5.5 to 9.5 cm)

2.787315505	2.518999111	1.947120913	1.54324955	1.3120086
3.368534618	3.150093136	2.228997129	1.64055811	1.4716348
3.868920636	3.635553733	2.744427755	2.0941518	1.778183
4.194624989	3.885619471	3.415108556	2.83962934	2.2349595
4.312115527	3.980207278	3.503293743	2.95174984	2.4053846
4.197772947	3.886404355	3.439684519	2.90988342	2.3752498
3.886404355	3.628566994	3.252537639	2.7347826	2.1559937
3.439684519	3.252537639	2.532402499	1.97130923	1.6747186
2.909883424	2.734782605	1.971309229	1.47941809	1.3455107
2.375249775	2.155993677	1.674718591	1.34551068	1.1672262

Relative errors

x = 0.5 to 4.5 cm

1.502477269	1.2594541	0.16337352	-0.062599	-0.0175753
1.092826931	1.44639179	0.5788584	0.0145781	0.0115931
0.245789237	0.33098911	0.36973668	0.0276386	-0.0538587
-0.004303732	0.04724164	-0.0086444	-0.073995	0.01328276
-0.01916693	0.00110356	-0.0307625	-0.04356	0.00092131
-0.040631554	-0.0375973	0.0280523	-0.032134	0.01171359
-0.073301846	-0.022108	-0.0268991	-0.021873	0.03040235
0.034701485	0.24680586	0.18186913	-0.050219	0.01323596
0.514985969	0.99735818	0.49715304	-0.074204	0.00992833
1.09330889	1.172081	0.37490005	-0.016377	-0.0967157

x = 5.5 to 9.5 cm

-0.054082087	-0.026033371	0.040203708	0.52112779	1.3950372
0.043061127	-0.005030563	0.300413478	0.98764458	1.8364381
-0.039581608	0.020016703	0.27663419	0.53424121	0.1817564
0.051078984	-0.01182079	0.08645853	0.0171288	-0.083723
-0.000450729	0.084138163	0.097502473	0.08020382	0.0078827
0.001494198	-0.007151963	-0.023463299	-0.0247834	0.0151512
0.034512372	0.02190126	-0.037705064	-0.029995	-0.065877
0.01157671	0.037303469	0.188072977	0.29885884	0.1088259
0.005277547	-0.020856658	0.233736897	1.0642083	0.9545705
-0.051151052	0.028735824	0.083213572	0.79256585	1.4201048

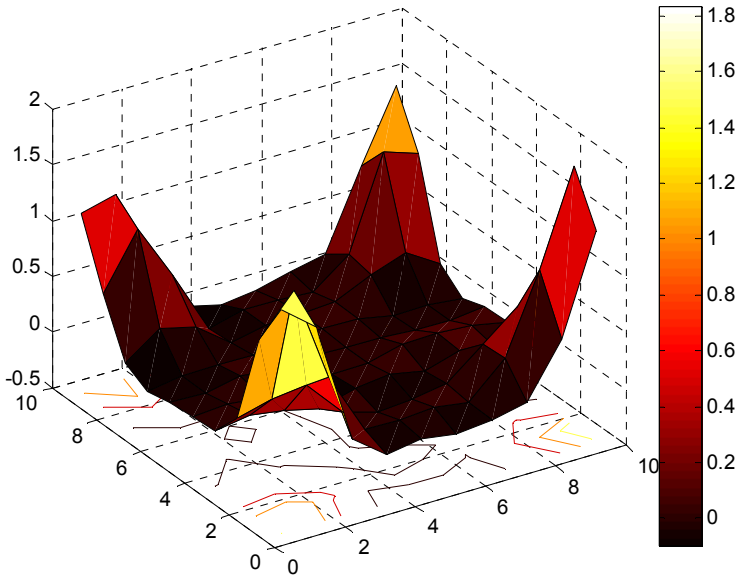


Figure 16

Figure 15 represents relative errors in the system slice at z = 0.5 cm. As stated in Section 4, the errors are very low almost all over the system slice but incredibly high at and around corners of the slice.

Appendix K- Future Work

The shortcomings of the SP_N-FVM code used in this thesis are.

- The code consumes large amounts of memory, more memory than even a Monte Carlo code like MCNP. This makes it difficult to large problems and makes it very difficult to use finer meshes for such problems.
- The code is very slow when it comes to generating the coefficient matrices.
- The code does not take advantage of the easily parallelizable block diagonal coefficient matrix as it does not invert the individual blocks separately to cut computation costs and inverts one giant uneconomical matrix instead
- The code does not have a cross section library and requires the user to specify cross sections.
- The code has a very primitive user interface often making it difficult to define complicated problem geometries.
- The code does not use an adaptive mesh and can only solve problems defined on Cartesian coordinates.
- The code does not incorporate anisotropic scattering of particles.
- The code assumes steady state
- The code cannot handle multiple energy group calculations and is constrained to one energy group only.
- The code is not yet ready to handle transient sources - dispersed radiation sources.

All the above shortcomings will be addressed in future work.

Bibliography

- Edward Larsen, G. T. (2002). *Simplified Pn Approximation to the Equations of Radiative Heat Transfer and Applications.* Elviser Science.
- Glasstone, G. B. (1970). *Nuclear Reactor Theory.* Malabar: Robert E. Krienger Publishing Company.
- Hamilton, J. D. (1976). *Nuclear Reactor Analysis.* Ann Arbor: John Wiley and Sons.
- Knupp, K. S. (2000). *Verification of Code Using the Method of Manufactured Coefficients.* Albuquerque: Sandia National Labs.
- R.Coilini, G. C. (2002). Simplified Pn and An Methods in Neutron Transport. *Progress in Nuclear Energy* , 1-28.
- E. E. Lewis (2008). Fundamentals of Nuclear Reactor Physics. Elviser Science.
- W. Kirschenmann, L. P. (2009). Massively Parallel Solving of 3D Simplified Pn Equations on Graphics Processing Units. *Conference on Mathematics, Computational Methods and Reactor Physics* (pp. 1-12). LaGrange Park : American Nuclear Society.
- W.F.Miller, E. L. (1993). *Computational Methods for Neutron Transport.* LaGrange Park: American Nuclear Society.
-
- Palmer, T. (2011, February). NE 654 Computational Particle Transport. Corvallis, Oregon, USA.