The Pennsylvania State University

The Graduate School

Department of Mechanical and Nuclear Engineering

SPHERICAL HARMONICS SOLUTIONS TO SECOND ORDER FORMS OF THE BOLTZMANN TRANSPORT EQUATION USING PARTICLE TRANSPORT CODE SCEPTRE

A Thesis in

Nuclear Engineering

by

Andrew Scott Bielen

© 2008 Andrew Scott Bielen

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Master of Science

May 2008

The thesis of Andrew Scott Bielen was reviewed and approved* by the following:

Yousry Y. Azmy
Professor of Nuclear Engineering
Thesis Advisor

Clifton Drumm
Principle Member of Technical Staff
Sandia National Laboratories

Kostadin Ivanov
Distinguished Professor of Nuclear Engineering

Jack Brenizer
Professor of Mechanical and Nuclear Engineering
Head of the Nuclear Engineering Program

*Signatures are on file in the Graduate School

ABSTRACT

Numerical simulations of basic physical processes are fundamental to how science is performed in this day in age. One of the processes of great interest to nuclear scientists and engineers is particle transport, in which the distribution of fundamental particles is calculated within a given region based on internal source distributions and external boundary conditions. The distribution can then be used to estimate physical effects of interest. The equation describing this process is known as the Boltzmann transport equation. In this work, second order forms of the transport equation are discretized in angle using the spherical harmonics methodology. The coefficients arising from the discretization were found to be consistent with expected values. The resulting discretization is implemented in Sandia National Laboratories' coupled photon-electron charged particle code SCEPTRE, and two test problems are presented. The solution to the equation and the convergence rates of the error with increasing spatial refinement are shown. It was found that the convergence rates for the second test problem were about an order lower than what was expected. The discrepancy is due to possible discontinuities in the first derivative of the solution, errors arising from an averaging procedure carried out on the output, or a lack of knowledge of the estimated convergence rate in the finite elements solver utilized by SCEPTRE.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

The author would like to thank the following people for their invaluable assistance with this work. First, he would like to thank his mentor at Sandia National Laboratories, New Mexico, Dr. Clifton Drumm. Also, thanks are due to his thesis advisor at Penn State, Professor Yousry Y. Azmy. In addition to Dr. Drumm, there are a number of people due for thanks at Sandia: Dr. Leonard Lorence, Dr. Wesley Fan, Dr. Shawn Pautz, and Dr. William Bonhoff. Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under contract DE-AC04-94AL85000. This research was performed under appointment to the U.S. Department of Energy Nuclear Engineering and Health Physics Fellowship Program sponsored by the U.S. Department of Energy's Office of Nuclear Energy, Science, and Technology.

Chapter 1

Introduction

1.1 Background

The rapid advance of computing technology since the Second World War has revolutionized the way that scientific research is performed. In the current climate, there is as much, if not more, emphasis on modeling physical processes computationally as there is on experiments. As such, it is necessary that methodologies be developed to reduce the continuum descriptions of various physical phenomena to discrete forms that allow for solutions on computers. One area of physical modeling that is of great interest to nuclear engineers is the process of particle transport, by which the distribution of subatomic particles within a host medium is ascertained by taking into account the physical interactions that these particles undergo while traveling through this medium. By determining the distribution of these particles, their effects on the host medium can be determined. By and large, nuclear engineers are interested in the transport of neutral particles, such as neutrons and photons. This is to be expected, as these are the primary particles of interest within nuclear reactors. The distribution of neutrons within these devices determines their performance. Much emphasis since the advent of the nuclear age has been placed on the numerical solution of neutral particle transport problems.

However, there exist situations in which the transport of charged particles is also of great interest. For instance, in the course of designing electronic components for

satellites, it is important to be able to predict the effect of cosmic radiation on these components. Charged-particle transport is an area that has not been as fully developed as neutral-particle transport, in part because of the more complicated nature of their model equations. This is due to the fact that charged particles undergo frequent interactions with the atomic electrons of the host medium, necessitating the inclusion of additional terms in the equations to represent this effect. Fortunately, strategies have been developed to deal with these complications, one of which will be discussed in greater detail later in this thesis.

The fundamental equation of general particle transport is known as the Boltzmann transport equation. It is a statement of particle conservation within an infinitesimal volume element in phase space. The dependent variable of the transport equation is the angular flux, defined as the number of particles at a given time per unit time passing through an infinitesimal area element per unit area at a given spatial location while traveling with a given velocity. As implied by the definition of the angular flux, the Boltzmann equation is dependent upon seven independent variables: a time variable, three spatial variables, and three velocity variables (it is common practice to express the velocity variables as a combination of one energy and two directional, or angular, variables). The seven-dimensional phase space makes for an enormous computational challenge; in solving the transport equation for general problems, the number of unknowns can quickly become prohibitively large. This has become less of an issue with the advent of modern supercomputing, but is still a constraint that engineers must keep in mind.

Given the complex nature of the transport equation, analytical solutions for practical problems are few and far between. Thus, it is necessary to derive discrete approximations for the Boltzmann equation that lend themselves to computational solution methods. There are two approaches to achieving this goal. The first, known as the Monte Carlo approach, involves directly simulating the histories of a large number of particles and tallying the results. It does not require discretization, but is not a solution of the Boltzmann equation; rather, it is a simulation of the basic physics involved in the transport process. This approach can be very accurate with a sufficiently large number of particles, but is often too computationally expensive for large applications. The other approach involves directly discretizing the Boltzmann equation. In order to do this, each of the independent variables is discretized. The energy variable is commonly discretized via an energy group structure. The spatial variables can be discretized in a number of ways. This work focuses on the discretization of the angular variables.

There are two common approaches to angular discretization. The scheme that is most commonly implemented is the discrete ordinates (S_N) method. This method involves requiring the transport equation to be satisfied along a given number of directions. The angular integrations that appear in the transport equation representing the production of secondary particles via scattering, fission, et cetera, are approximated by a weighted sum over this finite number of directions. Advantages of the S_N method include simplicity of the derivation of the discrete variable equations and of the development and implementation solution algorithms of high computational efficiency [1]. The second common method for angular discretization is the spherical harmonics

(P_N) method. The angular flux is expanded into a finite series of spherical harmonics, which are complete over the unit sphere. This expansion is substituted into the transport equation, and then the equation is multiplied on both sides by a spherical harmonic of arbitrary order. The resulting equation is next integrated over the unit sphere. Due to the mutual orthogonality of spherical harmonics, this process results in a finite number of equations for the expansion coefficients (flux moments) that depend on the angular variable through the index order of the spherical harmonics function. These equations are then subjected to a closure approximation, and then solved simultaneously for the expansion coefficients. The full angular flux can be recovered as a linear combination of spherical harmonics weighted with the expansion coefficients.

1.2 Motivation

The focus of this work lies in adding spherical harmonics capabilities to Sandia National Laboratories' particle transport code SCEPTRE (Sandia Computational Engine for Particle Transport for Radiation Effects). Originally developed as the coupled electron-photon transport code CEPTRE (Coupled Electron-Photon Transport for Radiation Effects), SCEPTRE has been expanded to allow for the modeling of more general transport problems. SCEPTRE is a massively parallel transport code that solves the time-independent linear Boltzmann equation [2]. It utilizes the energy-group scheme for energy discretization and a finite-elements spatial discretization scheme that enables users to model arbitrary geometries using polygons or polyhedra. The scattering kernel can be expanded into an arbitrarily high order series of Legendre polynomials.

SCEPTRE was initially developed using the S_N method for angular discretization. The solution algorithm utilized in SCEPTRE is unique in that, rather than use the common source-iteration technique, the angular and spatial dependencies are solved simultaneously. In order to do this, a very large matrix is constructed consisting of both angular and spatial unknowns for a given energy group, and the conjugate-gradients iterative method (noted for stability and robustness for symmetric positive-definite (SPD) linear algebraic systems) is used to arrive at a solution. The standard Boltzmann equation is first-order in space, and is not SPD. SCEPTRE gives the user the option of solving several second-order forms of the transport equation that are SPD.

The disadvantage of using the S_N discretization scheme within this context is that all the directions are coupled together via the scattering operator. This means that the angular submatrix for each spatial point is non-sparse, which can lead to longer computing times. In order to deal with this perceived problem, the P_N method has been implemented into the code. The angular unknowns in this approach are the flux moments, which are less tightly coupled than the angular flux unknowns in the S_N formulation. This leads to a sparser angular submatrix and a faster solution time for a comparable number of unknowns. Additionally, for some problems, the number of P_N moments needed may be much less than the number of S_N directions for a comparable degree of accuracy. A final advantage is that the spherical harmonics formulation mitigates ray effects that may be present in the discrete ordinates solution [1].

This thesis is organized as follows: In Chapter 2, a review of the literature is presented, which gives the reader a sense of the context in which the work was

performed. Then, in Chapter 3 the standard first-order Boltzmann transport equation is presented, and in Chapter 4 two of the second-order variations that are solved by SCEPTRE are derived. In Chapter 5, a derivation of the P_N equations for these forms is shown. Following this, in Chapter 6 a discussion of the implementation and verification of the P_N equations within the code is presented. Numerical results and comparisons for a variety of test problems are shown. Finally, in Chapter 7 conclusions and suggestions for future work are given.

Chapter 2

Review of Literature

In order to set the context within which this work was performed, a review of the literature in the relevant subject areas is presented. The first subject area discussed is second order forms of the transport equation. The second is the spherical harmonics methodology of handling the angular dependence present in the Boltzmann equation. Finally, the third area is charged-particle transport.

2.1 Second Order Forms of the Boltzmann Equation

The first general subject area considered is that of second-order forms of the transport equation. First, the even-odd parity (EOP) form of the Boltzmann equation is discussed. Lewis and Miller [1] devote a chapter to describing the even- and odd-parity formulations of the transport equation. Their presentation is limited to the case of isotropic scattering in one and two dimensions. Oliveira [3] extends the scattering treatment to arbitrary-order anisotropic scattering using a variational formulation and applies it to the even-parity transport equation. He then describes the implementation of this formulation in a multigroup spherical harmonics arbitrary-geometry finite elements code called EVENT. After this, he presents the results of some numerical comparisons. First, the EVENT solution is compared with analytic solutions to the transport equation in one-dimensional slab, and then spherical, geometries. Then, he does a comparison study

against a two-dimensional Cartesian problem for which the P_{19} exact kernel solution is known along two lines through the problem. Finally, he presents results for the GODIVA test problem in spherical, *R-Z,* and *X-Y-Z* geometries and compares them with the one-dimensional solution attained from the one-dimensional discrete ordinates code ANISN. In all cases, the EVENT solutions were acceptably close to the benchmark solutions. This work was important because it brought together several elements, such as finite elements, second-order forms of the transport equation, and generalized scattering, which are now present in SCEPTRE.

An alternative second-order formulation, known as the Self Adjoint Angular Flux (SAAF) equation, was first derived in a uniform slab geometry by Pomraning and Clark [4] by differentiating the first-order transport equation and manipulating the result. They developed a variational formulation of the second-order problem, and also used this formulation to develop alternative formulas for the diffusion coefficient and extrapolation distances which they showed to be more accurate than those developed classically. Their paper laid the groundwork for later authors to build upon while developing alternative formulations of the transport problem. Ackroyd [5] came across the SAAF while developing generalized least-squares approximations for the first order transport equation; he found that the SAAF equation is the Euler-Lagrange equation for a generalized least-squares functional in three-dimensional geometry. The work in this paper is set in a heavily detailed mathematical context rendering it very difficult for the casual reader to follow. In contrast, Morel and McGhee [6] showed how to derive the SAAF equation via a simple algebraic procedure from the first-order transport equation. They then presented an extensive comparison of the advantages and disadvantages of the

SAAF formulation relative to the EOP formulation (which will be revisited in Chapter 4), showed two angular discretization schemes for the monoenergetic SAAF equation in slab geometry (one using discrete ordinates and the other using spherical harmonics), and presented computational comparisons between the SAAF and EOP solutions. Their comparisons were done by computing the numerical solution to the S_2 SAAF and EOP equations on a non-uniform spatial mesh in slab geometry, computing the total and cellwise absorption rates, and comparing these with the absorption rates computed from the known analytic solution. Their numerical study indicated that the solution to the SAAF equation yielded comparable numerical accuracy and CPU time to the solution from the EOP formulation, albeit with a slightly higher CPU time due to the larger condition number for the SAAF equation. This paper was very important with respect to the development of the SCEPTRE code because it laid out in a clear format the SAAF equation that was later implemented. The previous authors, along with Roberts [7], next implemented the SAAF (as well as the EOP) formulations in a three-dimensional unstructured mesh finite elements code DANTE. This code offers options for discrete ordinates and both standard and simplified spherical harmonics to handle the angular discretization. DANTE is designed to be executed on massively parallel computer architectures. Their numerical results were mainly confined to timing studies as the mesh size was decreased. They noted that the performance of DANTE was comparable with respect to CPU time to the standard first-order discrete ordinates code ATTILA. This work is significant because the solution techniques employed are similar to those employed in SCEPTRE, with the major difference being that SCEPTRE employs a simultaneous space-angle solution, whereas DANTE uses standard source iteration.

Conceivably, DANTE could be used for comparison studies with SCEPRE. The second-order approaches to the Boltzmann equation were extended to the time-dependent radiative transfer equations describing the interaction of radiation with matter by Morel et al [8]. They discuss a temporal discretization scheme and give two formulations for spatial discretization using finite elements. They present numerical results for one- and two-dimensional problems. These results indicate strong agreement in both cases with established solutions. This work shows the possibility of extending the self-adjoint transport equation beyond solely particle transport. The discrete ordinates formulations for first and second order forms of the Boltzmann transport equation are described by Fan, Drumm and Powell [9]. They present an explicit derivation of both the EOP and SAAF formulations of the transport equation in three-dimensional geometry. They develop a matrix representation of the continuous operators in the transport equation using discrete ordinates angular discretization. A proof of the symmetric positive-definite nature of second-order transport operators is also presented. This work formulates the first- and second-order transport problems very clearly, and serves as a valuable reference for someone interested in learning about these methods.

2.2 Spherical Harmonics Angular Discretization

The spherical harmonics methodology for handling the angular dependence of the transport equation is long-established [10]. Lewis and Miller [1] discuss the spherical harmonics methodology and apply it to one-dimensional slab geometry. Gelbard [11] gives an overview of both the standard and double spherical harmonics approximations in

one dimensional slab, cylindrical and spherical geometries, and pays special attention to discussions of the boundary conditions. Davison [12] presents the spherical harmonics approximation in three-dimensional geometry. Fan [13] presents the equivalence of the spherical harmonics and discrete ordinates approximations in three-dimensional Cartesian geometry. By equivalence, it is meant that it is possible to arrive at the spherical harmonics equations by first starting with the discrete ordinates equations, and vice versa. He also gives a thorough discussion of the properties of spherical harmonics, presents the specific form of the spherical harmonics utilized in SCEPTRE, and derives the spherical harmonic approximation for the first-order transport equation in three-dimensional Cartesian geometry utilizing this formulation. This work was very useful in developing the second-order spherical harmonics approximations to the transport equation that were consistent with the existing version of SCEPTRE.

2.3 Charged Particle Transport

The phenomenon of charged-particle transport is discussed by Duderstadt and Martin [14]. They present the two mechanisms of low-mass charged particle interactions within a host material (these being strong, infrequent collisions with heavy ions and frequent weak collisions with atomic electrons), and discuss how the transport equation is modified accordingly. Morel [15] describes modifications to host cross-sections which allow charged-particle transport problems in one-dimensional slab and spherical geometries to be solved using standard linear Boltzmann solver routines. He provides numerical results from the implementation of these schemes in the linear one-

dimensional discrete ordinates code ONETRAN, and compares them with the charged-particle Monte Carlo code TIGER. The comparisons indicate that the methods developed in [15] were consistent with the Monte Carlo results, and thus the work serves as a basis upon which the methods could be developed further. This work was extended by Morel and Lorence to develop the coupled electron-photon cross section generation code CEPXS [16]. They give a detailed discussion of electron-photon transport physics and show how cross sections can be modified for standard one-dimensional transport solvers. CEPXS is still employed to generate the cross sections used in current computations. The one-dimensional results of [15] and [16] were extended to multidimensional discrete ordinates calculations by Drumm [17]. He derives a method for calculating multigroup electron and photon scattering cross sections by computing a simultaneous solution of the continuous-slowing-down and elastic-scattering portions of the scattering source by the Goldsmit-Saunderson theory. This solution process results in multigroup-Legendre cross sections that are much smaller in magnitude than the true cross sections they represent, and in many cases converge with a low-order Legendre expansion. This approach for generating multigroup-Legendre cross sections was implemented in a modified version of CEPXS called CEPXS-GS. He performs transport calculations employing cross sections from CEPXS-GS in the standard discrete ordinates transport codes DORT, TWODANT, and TORT, and compares the solutions with those obtained from charged-particle Monte Carlo codes TIGER and ACCEPT. The discrete ordinates solutions compared favorably with the Monte Carlo results except in the case of high-energy electrons incident on low-atomic-number materials. He also compares cross sections obtained from CEPXS-GS and compares them with those obtained from CEPXS, and found that the generated cross

sections from CEPXS-GS did not display unwanted oscillatory behavior evident in the standard CEPXS approach. The poor results for high-energy electron beams incident on low-atomic-number materials was rectified by Drumm, Fan, Lorence and Liscum-Powell [18] by means of the Extended Transport Correction (ETC). In this methodology, the scattering cross section is represented by two parts: a forward δ-function in angle and a relatively low-order Legendre expansion. A computation of the ETC cross section moments for highly-forward-peaked elastic scattering cross sections was compared against the standard Legendre expansion approach and was observed to be much closer to the analytic values. The ETC approach was implemented within CEPXS, and the resulting cross sections were employed by the CEPTRE code to generate discrete ordinates transport solutions for high-energy (20 MeV) electron beams incident upon water. The resulting energy- and charge-deposition profiles determined from these solutions were found to be in good agreement with Monte Carlo solutions generated using ACCEPT. Josef and Morel [19] develop simplified spherical harmonics equations for first-order coupled electron-photon transport using CEPXS-generated cross sections, and present numerical results for test problems in one- and two-dimensional geometries. The numerical results were found to compare favorably with Monte Carlo solutions, and thus show that the simplified spherical harmonic treatment is an effective alternative to solving coupled photon-electron transport problems. Drumm [20] describes a solution algorithm for the Boltzmann equation using an unstructured triangular mesh and gives a description of the coupled spatial-angular solution methodology that would be subsequently included into early versions of SCEPTRE. He provides a number of convergence studies in both the angular approximation and the spatial discretization.

This paper serves as an indicator as to the direction that Sandia National Laboratories' deterministic particle transport code development would go. Bonhoff et al [2] gives a basic overview of the original CEPTRE code, describing its solution methodology and giving preliminary numerical results. Liscum-Powell et al [21] describe using a diamond-differencing technique to approximate the energy derivative in the continuous-slowing down term of the charged-particle transport equation and describes the implementation of this approximation in the one-dimensional transport code DOET_1D. The same authors [22] then extend this work to approximate the energy derivative using a linear-discontinuous finite elements technique. They implement this methodology into the SAAF form of the transport equation in one-dimensional planar geometry using the discrete ordinates approximation in angle and a linear-continuous finite elements approximation in space, and conclude by discussing acceleration schemes to speed convergence of the solution.

With the context of this work set, it is now appropriate to move to the description of the transport problem solved by SCEPTRE.

Chapter 3

The Boltzmann Transport Equation

3.1 Derivation of Equation Solved by SCEPTRE

As mentioned above, the fundamental equation upon which transport theory is built is the Boltzmann equation. Before presenting this equation, it is appropriate to define its dependent variable. The particle density $N(\mathbf{r},\hat{\Omega},E,t)$ is distribution that gives the particle density within the host medium; $N(\mathbf{r},\hat{\Omega},E,t)d^3rd\hat{\Omega}dE$ gives the number of particles at time *t* within the volume element d^3r located at point **r** traveling with energy within [*E, E+dE*] and direction within $[\hat{\Omega},\ \hat{\Omega}+d\hat{\Omega}]$. In particle transport problems, it is more customary to solve for the angular flux $\psi(\mathbf{r},\hat{\Omega},E,t)$, which is given as the product of the particle density *N* with the particle velocity *v*, i.e.,

$$\psi(\mathbf{r},\hat{\Omega},E,t) = v(E)N(\mathbf{r},\hat{\Omega},E,t)$$

With this definition in mind, a time-independent transport equation for charged particles can be written as follows [22]:

$$\hat{\Omega}\bullet\nabla\psi(\mathbf{r},\hat{\Omega},E) - \frac{\partial}{\partial E}\left\{SP(\mathbf{r},E)\psi(\mathbf{r},\hat{\Omega},E)\right\} + \sigma(\mathbf{r},E)\psi(\mathbf{r},\hat{\Omega},E) = \int_{4\pi}\int_{0}^{\infty}\sigma_s(\mathbf{r},\hat{\Omega}'\to\hat{\Omega},E'\to E)\psi(\mathbf{r},\hat{\Omega}',E')dE'd\hat{\Omega}' + S(\mathbf{r},\hat{\Omega},E). \quad \textbf{3.1}$$

This equation is defined for all points **r** within a control volume *V,* for all angles $\hat{\Omega}$ lying on the unit sphere and for all energies $E \in [0, \infty]$. In this equation, *SP*(**r,** *E*) is the material stopping power, $\sigma(\mathbf{r}, E)$ is the total macroscopic interaction cross section, $\sigma_s(\mathbf{r}, \hat{\Omega}' \to \hat{\Omega}, E' \to E)$ is the double-differential scattering cross section, and $S(\mathbf{r}, \hat{\Omega}, E)$ denotes the external source. In this equation, the losses of the left hand side are balanced by the production mechanisms on the right hand side. The first term on the left hand side represents particle streaming (net movement out of the volume element), the second represents energy loss of the particles due to electrical interactions with the host medium, and the third represents all collisions. The right hand side's first term encompasses all particles of energy and direction different from *E* and $\hat{\Omega}$ scattering into the energy interval in question, and the second term denotes production mechanisms independent of the solution $\psi(\mathbf{r}, \hat{\Omega}, E, t)$.

The second term on the left hand side is where the charged particle transport equation differs from the standard linear Boltzmann equation describing neutral particle transport. This term, known as the continuous slowing-down (CSD) term, is an approximation that takes into account the loss in energy that charged particles undergo through soft inelastic interactions which do not modify the particles' direction. There are two choices one can make when dealing with this term. The first is to attempt to discretize it directly, which has been done using various schemes ([21], [22]). Alternatively, one can employ the approach first suggested by Morel [15] and later implemented in Sandia National Laboratories' cross-section generation code CEPXS [16], and then extended by Drumm [17] and Drumm et al [18], where the multigroup-

Legendre cross sections are modified to take into account CSD. The advantage to this approach is that one can use standard transport solvers (solving the linear Boltzmann equation) for charged particle transport problems. SCEPTRE has been built with this approach in mind; thus, the equation actually solved by the code has the following structure:

$$\hat{\Omega}\bullet\nabla\psi(\mathbf{r},\hat{\Omega},E)+\sigma(\mathbf{r},E)\psi(\mathbf{r},\hat{\Omega},E)= \int_{4\pi}\int_{0}^{\infty}\sigma_s(\mathbf{r},\hat{\Omega}'\to\hat{\Omega},E'\to E)\psi(\mathbf{r},\hat{\Omega}',E')dE'd\hat{\Omega}'+S(\mathbf{r},\hat{\Omega},E), \quad \textbf{3.2}$$

which is simply the first-order linear Boltzmann equation for neutral particle transport.

3.2 Expansion of the Scattering Term

The scattering cross section is a complicated function of scattering angle and energy. Thus, in order to deal with the scattering term numerically it is customary to represent the scattering cross section as an expansion of orthogonal polynomials. The most common expansion is in Legendre polynomials. First, the scattering cross section is written in terms of the cosine of the angle between the direction of motion of the incident and scattered particles, $\hat{\Omega}$ and $\hat{\Omega}'$, respectively, in the laboratory system:

$$\begin{aligned}\sigma_s(\mathbf{r},\hat{\Omega}'\to\hat{\Omega},E'\to E)&=\sigma_s(\mathbf{r},\hat{\Omega}'\bullet\hat{\Omega},E'\to E)\\&=\sigma_s(\mathbf{r},\mu_0,E'\to E)\\&=\sum_{l=0}^{\infty}\frac{2l+1}{4\pi}\sigma_l(\mathbf{r},E'\to E)P_l(\mu_0)\end{aligned} \quad \textbf{3.3}$$

where the scattering moments $\sigma_l(\mathbf{r},E'\to E)$ are given by:

$$\sigma_l(\mathbf{r}, E' \to E) = 2\pi \int_{-1}^{1} \sigma_s(\mathbf{r}, \mu_0, E' \to E) P_l(\mu_0) d\mu_0 \qquad \textbf{3.4}$$

where the angle-to-angle dependence of the scattering cross section is written in terms of the angle cosine $\mu_0 \equiv \hat{\Omega}' \bullet \hat{\Omega}$.

From a computational point of view, the infinite series in Eq. **3.3** must be truncated to attain the desired level of accuracy of the approximation. In neutral particle transport, a low order of expansion is usually sufficient to accurately capture the typical anisotropy in the neutral particle scattering cross section; however, charged-particle scattering is typically highly anisotropic, and much higher expansion orders are required to give reasonable results.

The scattering expansion is then inserted into the transport equation:

$$\begin{aligned} &\hat{\Omega} \bullet \nabla \psi(\mathbf{r}, \hat{\Omega}, E) + \sigma(\mathbf{r}, E) \psi(\mathbf{r}, \hat{\Omega}, E) = \\ &\sum_{l=0}^{\infty} \frac{2l+1}{4\pi} \int_{4\pi} \int_{0}^{\infty} \sigma_l(\mathbf{r}, E' \to E) P_l(\mu_0) \psi(\mathbf{r}, \hat{\Omega}', E') dE' d\hat{\Omega}' + S(\mathbf{r}, \hat{\Omega}, E) \end{aligned} \qquad \textbf{3.5}$$

Using the Legendre Addition Theorem of spherical harmonics:

$$P_l(\mu_0) = \frac{4\pi}{2l+1} \sum_{m=-l}^{l} Y_{lm}(\hat{\Omega}) Y_{lm}^*(\hat{\Omega}') \qquad \textbf{3.6}$$

where $Y_{lm}(\hat{\Omega})$ is the spherical harmonic of order *l* and *m,* and $Y_{lm}^*(\hat{\Omega}')$ is its complex conjugate, the transport equation is now expressed as:

$$\begin{aligned} &\hat{\Omega} \bullet \nabla \psi(\mathbf{r}, \hat{\Omega}, E) + \sigma(\mathbf{r}, E) \psi(\mathbf{r}, \hat{\Omega}, E) = \\ &\sum_{l=0}^{\infty} \sum_{m=-l}^{l} Y_{lm}(\hat{\Omega}) \int_{4\pi} \int_{0}^{\infty} \sigma_l(\mathbf{r}, E' \to E) Y_{lm}^*(\hat{\Omega}') \psi(\mathbf{r}, \hat{\Omega}', E') dE' d\hat{\Omega}' + S(\mathbf{r}, \hat{\Omega}, E) \end{aligned} \qquad \textbf{3.7}$$

3.3 Energy Discretization

The next step in discretizing the transport equation is to apply an energy discretization. As mentioned above, SCEPTRE makes use of the standard energy group approach. First, it is assumed that it is reasonable to approximate the infinite energy range $E \in [0,\infty]$ with a finite energy range that goes from some minimum energy E_G to a maximum energy E_0. Integration over the whole energy range is thus approximated as:

$$\int_0^{\infty} dE \approx \int_{E_G}^{E_0} dE \qquad \textbf{3.8}$$

The finite range is partitioned into *G* groups, with group *g* consisting of the energy interval from E_g to E_{g-1}. The group flux is defined thusly:

$$\psi_g(\mathbf{r},\hat{\Omega}) \equiv \int_{E_g}^{E_{g-1}} \psi(\mathbf{r},\hat{\Omega},E)dE \qquad \textbf{3.9}$$

Assume that the energy dependence of the flux can be written as the product of a known function of energy $f(E)$ and the group flux defined in Eq. **3.9**:

$$\psi(\mathbf{r},E,\hat{\Omega}) = f(E)\psi_g(\mathbf{r},\hat{\Omega}),\ E_g < E \le E_{g-1}$$

To maintain consistency with Eq. **3.9**, the function $f(E)$ must be normalized as:

$$\int_{E_g}^{E_{g-1}} f(E)dE = 1$$

The function $f(E)$ is known as the spectral weighting function. The choice of the appropriate $f(E)$ is dependent on the physical situation that is being modeled; see [1] for a more thorough discussion.

The group source, the group total interaction cross section, and the scattering moment from energy group g' to group g are defined by the following relations:

$$S_g(\mathbf{r},\hat{\Omega}) \equiv \int_{E_g}^{E_{g-1}} S(\mathbf{r},\hat{\Omega},E)dE \qquad \textbf{3.10}$$

$$\sigma_g(\mathbf{r}) \equiv \int_{E_g}^{E_{g-1}} \sigma(\mathbf{r},E) f(E)dE \qquad \textbf{3.11}$$

$$\sigma_{l,g'g}(\mathbf{r}) \equiv \int_{E_{g'}}^{E_{g'-1}} \int_{E_g}^{E_{g-1}} \sigma_l(\mathbf{r},E' \to E) f(E')dEdE' \qquad \textbf{3.12}$$

The transport equation Eq. **3.7** is then integrated with respect to energy from E_g to E_{g-1}.

Making use of the fact that $\int_0^{\infty} dE' \approx \int_{E_G}^{E_0} dE' = \sum_{g'=1}^{G} \int_{E_{g'}}^{E_{g'-1}} dE'$, Eq. **3.7** can be written as:

$$\begin{aligned} &\hat{\Omega}\bullet\nabla\psi_g(\mathbf{r},\hat{\Omega}) + \sigma_g(\mathbf{r})\psi_g(\mathbf{r},\hat{\Omega}) = \\ &\sum_{g'=1}^{G}\sum_{l=0}^{\infty}\sum_{m=-l}^{l} \sigma_{l,g'g}(\mathbf{r})Y_{lm}(\hat{\Omega}) \int_{4\pi} Y_{lm}^*(\hat{\Omega}')\psi_{g'}(\mathbf{r},\hat{\Omega}')d\hat{\Omega}' + S_g(\mathbf{r},\hat{\Omega}) \end{aligned} \qquad \textbf{3.13}$$

The scattering term can be written as the sum of the within-group scattering term and the group-to-group scattering term:

$$\begin{aligned} &\sum_{g'=1}^{G}\sum_{l=0}^{\infty}\sum_{m=-l}^{l} \sigma_{l,g'g}(\mathbf{r})Y_{lm}(\hat{\Omega}) \int_{4\pi} Y_{lm}^*(\hat{\Omega}')\psi_{g'}(\mathbf{r},\hat{\Omega}')d\hat{\Omega}' = \\ &\sum_{l=0}^{\infty}\sum_{m=-l}^{l} \sigma_{l,gg}(\mathbf{r})Y_{lm}(\hat{\Omega}) \int_{4\pi} Y_{lm}^*(\hat{\Omega}')\psi_{g}(\mathbf{r},\hat{\Omega}')d\hat{\Omega}' \\ &+\sum_{g'\neq g}\sum_{l=0}^{\infty}\sum_{m=-l}^{l} \sigma_{l,g'g}(\mathbf{r})Y_{lm}(\hat{\Omega}) \int_{4\pi} Y_{lm}^*(\hat{\Omega}')\psi_{g'}(\mathbf{r},\hat{\Omega}')d\hat{\Omega}' \end{aligned} \qquad \textbf{3.14}$$

By doing this, we can define a source term combining both the external group source and the contributions from other energy groups Q_g:

$$Q_g(\mathbf{r},\hat{\Omega}) \equiv \sum_{g'\neq g}\sum_{l=0}^{\infty}\sum_{m=-l}^{l}\sigma_{l,g'g}(\mathbf{r})Y_{lm}(\hat{\Omega})\int_{4\pi}Y_{lm}^{*}(\hat{\Omega}')\psi_{g'}(\mathbf{r},\hat{\Omega}')d\hat{\Omega}' + S_g(\mathbf{r},\hat{\Omega}) \qquad \textbf{3.15}$$

and Eq. **3.13** can be rewritten as

$$\begin{aligned}&\hat{\Omega}\bullet\nabla\psi_g(\mathbf{r},\hat{\Omega}) + \sigma_g(\mathbf{r})\psi_g(\mathbf{r},\hat{\Omega}) = \\ &\sum_{l=0}^{\infty}\sum_{m=-l}^{l}\sigma_{l,gg}(\mathbf{r})Y_{lm}(\hat{\Omega})\int_{4\pi}Y_{lm}^{*}(\hat{\Omega}')\psi_g(\mathbf{r},\hat{\Omega}')d\hat{\Omega}' + Q_g(\mathbf{r},\hat{\Omega})\end{aligned} \qquad \textbf{3.16}$$

Thus, all contributions to the production terms from scattering events at energies outside of group g are contained within Q_g. The transport equation is now expressed in terms of a single group, does not depend explicitly on ψ_g, and hence from this point forward the group index g will be suppressed, giving the one-group first order Boltzmann equation:

$$\begin{aligned}&\hat{\Omega}\bullet\nabla\psi(\mathbf{r},\hat{\Omega}) + \sigma(\mathbf{r})\psi(\mathbf{r},\hat{\Omega}) = \\ &\sum_{l=0}^{\infty}\sum_{m=-l}^{l}\sigma_l(\mathbf{r})Y_{lm}(\hat{\Omega})\int_{4\pi}Y_{lm}^{*}(\hat{\Omega}')\psi(\mathbf{r},\hat{\Omega}')d\hat{\Omega}' + Q(\mathbf{r},\hat{\Omega})\end{aligned} \qquad \textbf{3.17}$$

A set of one group equations for each energy group, coupled through Q, are solved successively, giving the full energy dependence of the angular flux.

As stated above, the total interaction term gives the rate at which particles in a given group undergo various interactions with the host medium. If the within group scattering term, which gives the scattering interactions that cause a change of direction but no change in energy group, is moved from the right-hand side of Eq. **3.17** to the left hand side, a removal operator R can be defined, giving the rate at which particles are removed from the group via scattering interactions,

$$R\psi(\mathbf{r},\hat{\Omega}) \equiv \sigma(\mathbf{r})\psi(\mathbf{r},\hat{\Omega}) - \sum_{l=0}^{\infty}\sum_{m=-l}^{l}\sigma_l(\mathbf{r})Y_{lm}(\hat{\Omega})\int_{4\pi}Y_{lm}^{*}(\hat{\Omega}')\psi(\mathbf{r},\hat{\Omega}')d\hat{\Omega}' \qquad \textbf{3.18}$$

The transport equation can thus be written succinctly as:

$$\hat{\Omega}\bullet\nabla\psi(\mathbf{r},\hat{\Omega})+R\psi(\mathbf{r},\hat{\Omega})=Q(\mathbf{r},\hat{\Omega}) \qquad \textbf{3.19}$$

3.4 Boundary Conditions for Boltzmann Equation

In order to arrive at a unique solution for a given problem, it is necessary to impose boundary conditions upon the transport equation. In many instances, the angular flux incident upon the problem domain is known, which gives *prescribed incoming flux* boundary conditions:

$$\psi(\mathbf{r}_b,\hat{\Omega})=f(\mathbf{r}_b,\hat{\Omega}) \text{ for } \mathbf{r}_b\in\partial V,\ \mathbf{n}\bullet\hat{\Omega}<0 \qquad \textbf{3.20}$$

where ∂V indicates the boundary of the problem domain V and $\mathbf{n}$ denotes the unit outward normal to ∂V at point r_b. If $f(\mathbf{r},\hat{\Omega})=0,$ there is no incoming flux; this type of boundary condition is referred to as a *vacuum* boundary condition.

The other two common kinds of boundary conditions are *periodic* and *reflective* boundary conditions [1]. In an infinite periodic material array of pitch $\mathbf{r}_p$, the angular flux incoming to the problem domain at a given point in phase space $(\mathbf{r},\hat{\Omega})$ is equal to the outgoing angular flux at a point a distance $\mathbf{r}_p$ away, namely, $(\mathbf{r}_p,\hat{\Omega})$. This gives the periodic boundary condition:

$$\psi(\mathbf{r},\hat{\Omega})=\psi(\mathbf{r}+\mathbf{r}_p,\hat{\Omega}) \qquad \textbf{3.21}$$

Finally, reflective boundary conditions describe spectral reflection of particles back into the host medium at a boundary. These boundary conditions are usually used at lines or planes of symmetry and can greatly reduce the number of unknowns for a given problem. They are given explicitly by

$$\psi(\mathbf{r},\hat{\Omega}) = \psi(\mathbf{r},\hat{\Omega}'),\ \mathbf{n}\bullet\hat{\Omega} = -\mathbf{n}\bullet\hat{\Omega}' \text{ and } (\hat{\Omega}\times\hat{\Omega}')\bullet\mathbf{n} = 0 \qquad \textbf{3.22}$$

The transport problem has now been formally introduced. Next, the specific forms of the Boltzmann equation that are solved in SCEPTRE will be described, and the spherical harmonics equations for these forms will be derived.

Chapter 4

Second Order Forms of the Transport Equation

The first-order form of the Boltzmann equation is the most widespread formulation used for particle transport calculations. However, second-order forms of this equation have also been successfully implemented for particle transport calculations. Generally, the second-order forms are useful in finite-element spatial discretizations because of their symmetric positive-definite property. SCEPTRE has the capability to solve both the Self-Adjoint Angular Flux (SAAF) and the Even/Odd Parity (EOP) formulations of the Boltzmann equation.

4.1 Self-Adjoint Angular Flux Formulation

The SAAF formulation has been in existence for some time ([4], [5]), but was not cast in a general setting until earlier this decade ([6]). It is quite simple to derive. We start with the first-order one-speed transport equation:

$$\hat{\Omega}\bullet\nabla\psi(\mathbf{r},\hat{\Omega}) + R\psi(\mathbf{r},\hat{\Omega}) = Q(\mathbf{r},\hat{\Omega}) \qquad \textbf{4.1}$$

The streaming operator is moved to the right-hand side, and the removal operator inverted, to arrive at an expression for the angular flux:

$$\psi(\mathbf{r},\hat{\Omega}) = R^{-1}\left[Q(\mathbf{r},\hat{\Omega}) - \hat{\Omega}\bullet\nabla\psi(\mathbf{r},\hat{\Omega}) \right] \qquad \textbf{4.2}$$

This expression is then put back into the streaming operator in Eq. **4.1**

$$\hat{\Omega}\bullet\nabla R^{-1}\left[Q(\mathbf{r},\hat{\Omega})-\hat{\Omega}\bullet\nabla\psi(\mathbf{r},\hat{\Omega})\right]+R\psi(\mathbf{r},\hat{\Omega})=Q(\mathbf{r},\hat{\Omega}) \qquad \mathbf{4.3}$$

Next, distribute the streaming operator on the left hand side through and move the source term to the right hand side to arrive at the SAAF form of the Boltzmann transport equation:

$$-\hat{\Omega}\bullet\nabla R^{-1}\hat{\Omega}\bullet\nabla\psi(\mathbf{r},\hat{\Omega})+R\psi(\mathbf{r},\hat{\Omega})=Q(\mathbf{r},\hat{\Omega})-\hat{\Omega}\bullet\nabla R^{-1}Q(\mathbf{r},\hat{\Omega}) \qquad \mathbf{4.4}$$

It remains to derive the explicit form of the inverse removal operator R^{-1} [9]. In order to do this, start with the first-order form with the streaming operator moved to the right-hand side and the within-group scattering operator moved to the left hand side:

$$\begin{aligned}&\sigma(\mathbf{r})\psi(\mathbf{r},\hat{\Omega})-\sum_{l=0}^{\infty}\sum_{m=-l}^{l}\sigma_l(\mathbf{r})Y_{lm}(\hat{\Omega})\int_{4\pi}Y_{lm}^{*}(\hat{\Omega}')\psi(\mathbf{r},\hat{\Omega}')d\hat{\Omega}'\\&=Q(\mathbf{r},\hat{\Omega})-\hat{\Omega}\bullet\nabla\psi(\mathbf{r},\hat{\Omega})\end{aligned} \qquad \mathbf{4.5}$$

This equation is multiplied by $Y_{lm}^{*}(\hat{\Omega})$ and the result is integrated over the solid angle. Making use of the orthogonality property of spherical harmonics, the following equation is obtained:

$$\begin{aligned}&\sigma(\mathbf{r})\int_{4\pi}Y_{lm}^{*}(\hat{\Omega})\psi(\mathbf{r},\hat{\Omega})d\hat{\Omega}-\sigma_l(\mathbf{r})\int_{4\pi}Y_{lm}^{*}(\hat{\Omega})\psi(\mathbf{r},\hat{\Omega})d\hat{\Omega}\\&=\int_{4\pi}Y_{lm}^{*}(\hat{\Omega})\left[Q(\mathbf{r},\hat{\Omega})-\hat{\Omega}\bullet\nabla\psi(\mathbf{r},\hat{\Omega})\right]d\hat{\Omega}\end{aligned} \qquad \mathbf{4.6}$$

Simplifying,

$$\begin{aligned}&\int_{4\pi}Y_{lm}^{*}(\hat{\Omega})\psi(\mathbf{r},\hat{\Omega})d\hat{\Omega}=\\&\left[\sigma(\mathbf{r})-\sigma_l(\mathbf{r})\right]^{-1}\int_{4\pi}Y_{lm}^{*}(\hat{\Omega})\left[Q(\mathbf{r},\hat{\Omega})-\hat{\Omega}\bullet\nabla\psi(\mathbf{r},\hat{\Omega})\right]d\hat{\Omega}\end{aligned} \qquad \mathbf{4.7}$$

Now, this expression is substituted into Eq. **4.5**:

$$\begin{aligned}&\sigma(\mathbf{r})\psi(\mathbf{r},\hat{\Omega})\\&-\sum_{l=0}^{\infty}\sum_{m=-l}^{l}\frac{\sigma_l(\mathbf{r})}{\sigma(\mathbf{r})-\sigma_l(\mathbf{r})}Y_{lm}(\hat{\Omega})\int_{4\pi}Y_{lm}^{*}(\hat{\Omega}')\left[Q(\mathbf{r},\hat{\Omega}')-\hat{\Omega}'\bullet\nabla\psi(\mathbf{r},\hat{\Omega}')\right]d\hat{\Omega}'\\&=Q(\mathbf{r},\hat{\Omega})-\hat{\Omega}\bullet\nabla\psi(\mathbf{r},\hat{\Omega})\end{aligned}\qquad\textbf{4.8}$$

Next, Eq. **4.8** is solved for the angular flux:

$$\begin{aligned}\psi(\mathbf{r},\hat{\Omega})&=\frac{1}{\sigma(\mathbf{r})}\left[Q(\mathbf{r},\hat{\Omega})-\hat{\Omega}\bullet\nabla\psi(\mathbf{r},\hat{\Omega})\right]\\&+\frac{1}{\sigma(\mathbf{r})}\sum_{l=0}^{\infty}\frac{\sigma_l(\mathbf{r})}{\sigma(\mathbf{r})-\sigma_l(\mathbf{r})}\sum_{m=-l}^{l}Y_{lm}(\hat{\Omega})\int_{4\pi}Y_{lm}^{*}(\hat{\Omega}')\left[Q(\mathbf{r},\hat{\Omega}')-\hat{\Omega}'\bullet\nabla\psi(\mathbf{r},\hat{\Omega}')\right]d\hat{\Omega}'\end{aligned}\qquad\textbf{4.9}$$

The explicit form of R^{-1} can be determined by comparing Eq. **4.9** with Eq. **4.2**:

$$R^{-1}f(\mathbf{r},\hat{\Omega})=\frac{f(\mathbf{r},\hat{\Omega})}{\sigma(\mathbf{r})}+\frac{1}{\sigma(\mathbf{r})}\sum_{l=0}^{\infty}\frac{\sigma_l(\mathbf{r})}{\sigma(\mathbf{r})-\sigma_l(\mathbf{r})}\sum_{m=-l}^{l}Y_{lm}(\hat{\Omega})\int_{4\pi}Y_{lm}^{*}(\hat{\Omega}')f(\mathbf{r},\hat{\Omega}')d\hat{\Omega}'\qquad\textbf{4.10}$$

With the inverse removal operator defined as such, the SAAF equation can be completely formulated and solved.

The SAAF form of the Boltzmann equation has many interesting features of note [6], [9]. First of all, the unknown of this equation is the full angular flux, in contrast to the EOP formulation, which will be discussed later. This means that the boundary conditions given in the section describing the first-order transport equation are still applicable. However, because of the second-order spatial operator in the SAAF, an additional boundary condition is required. Morel and McGhee suggest requiring that the first-order equation be satisfied on the boundaries for outgoing directions [6]. This approach has been implemented successfully in SCEPTRE. Additionally, the matrix equations resulting from the application of continuous finite elements is SPD, which

allows for the application of extremely efficient and robust conjugate-gradient iterative solutions. SCEPTRE employs a Krylov subspace method to arrive at a solution.

A major difficulty associated with the SAAF equation (and second-order formulations in general) is the solution of the transport equation in void regions. As seen in Eq. **4.10**, the SAAF equation involves division by the total cross section $\sigma(\mathbf{r})$. In a void, $\sigma(\mathbf{r}) = 0,$ and the equation breaks down. A way to deal with this complication is to treat the void region as a uniform, purely absorbing medium, multiply both sides of the equation by $\sigma(\mathbf{r})$ and take the limit as $\sigma(\mathbf{r}) \to 0$. In this limit, Eq. **4.4** becomes very similar to the standard Laplace equation. However, this equation does not, in general, maintain particle conservation. In the fine-mesh limit, conservation is achieved, but in general this is not the case and can lead to erroneous results.

4.2 Even/Odd ParityFormulation

An alternative second order formulation of the transport problem solved by SCEPTRE is the EOP form [9]. Of the second-order forms, this is the most widespread, primarily for use in transport finite elements codes ([1], [3]). In addition to relative ease of finite element solutions, this formulation also makes use of symmetries in angle to greatly reduce the number of unknowns to be solved for, which reduces computation time considerably.

To derive this form of the EOP equations, we again start with the first-order form of the transport equation:

$$\hat{\Omega}\bullet\nabla\psi(\mathbf{r},\hat{\Omega})+\sigma(\mathbf{r})\psi(\mathbf{r},\hat{\Omega})= \\ \sum_{l=0}^{\infty}\frac{2l+1}{4\pi}\sigma_l(\mathbf{r})\int_{4\pi}P_l(\hat{\Omega}\bullet\hat{\Omega}')\psi(\mathbf{r},\hat{\Omega}')d\hat{\Omega}'+Q(\mathbf{r},\hat{\Omega}) \qquad \textbf{4.11}$$

Next, we define even and odd components of the flux solution ($\psi^e(\mathbf{r},\hat{\Omega})$ and $\psi^o(\mathbf{r},\hat{\Omega})$, respectively):

$$\psi^e(\mathbf{r},\hat{\Omega})\equiv\frac{1}{2}\Big(\psi(\mathbf{r},\hat{\Omega})+\psi(\mathbf{r},-\hat{\Omega})\Big) \qquad \textbf{4.12}$$

$$\psi^o(\mathbf{r},\hat{\Omega})\equiv\frac{1}{2}\Big(\psi(\mathbf{r},\hat{\Omega})-\psi(\mathbf{r},-\hat{\Omega})\Big) \qquad \textbf{4.13}$$

Note that the total flux is the sum of the even and odd components:

$$\psi(\mathbf{r},\hat{\Omega})=\psi^e(\mathbf{r},\hat{\Omega})+\psi^o(\mathbf{r},\hat{\Omega}) \qquad \textbf{4.14}$$

Similarly, we define even and odd source components $Q^e(\mathbf{r},\hat{\Omega})$ and $Q^o(\mathbf{r},\hat{\Omega})$:

$$Q^e(\mathbf{r},\hat{\Omega})\equiv\frac{1}{2}\Big(Q(\mathbf{r},\hat{\Omega})+Q(\mathbf{r},-\hat{\Omega})\Big) \qquad \textbf{4.15}$$

$$Q^o(\mathbf{r},\hat{\Omega})\equiv\frac{1}{2}\Big(Q(\mathbf{r},\hat{\Omega})-Q(\mathbf{r},-\hat{\Omega})\Big) \qquad \textbf{4.16}$$

Since the transport equation holds for any angle $\hat{\Omega}$, it is valid to require Eq. **4.11** to hold for $-\hat{\Omega}$:

$$-\hat{\Omega}\bullet\nabla\psi(\mathbf{r},-\hat{\Omega})+\sigma(\mathbf{r})\psi(\mathbf{r},-\hat{\Omega})= \\ \sum_{l=0}^{\infty}\frac{2l+1}{4\pi}\sigma_l(\mathbf{r})\int_{4\pi}P_l(-\hat{\Omega}\bullet\hat{\Omega}')\psi(\mathbf{r},\hat{\Omega}')d\hat{\Omega}'+Q(\mathbf{r},-\hat{\Omega}) \qquad \textbf{4.17}$$

Next, add Eq. **4.11** and Eq. **4.17**:

$$\hat{\Omega}\bullet\nabla\Big[\psi(\mathbf{r},\hat{\Omega})-\psi(\mathbf{r},-\hat{\Omega})\Big]+\sigma(\mathbf{r})\Big[\psi(\mathbf{r},\hat{\Omega})+\psi(\mathbf{r},-\hat{\Omega})\Big]= \\ \sum_{l=0}^{\infty}\frac{2l+1}{4\pi}\sigma_l(\mathbf{r})\int_{4\pi}\Big[P_l(\hat{\Omega}\bullet\hat{\Omega}')+P_l(-\hat{\Omega}\bullet\hat{\Omega}')\Big]\psi(\mathbf{r},\hat{\Omega}')d\hat{\Omega}'+Q(\mathbf{r},\hat{\Omega})+Q(\mathbf{r},-\hat{\Omega}) \qquad \textbf{4.18}$$

Making use of Eq. **4.12**-**16** and the fact that the Legendre polynomials $P_l(\mu)$ are symmetric for even values of *l* and asymmetric for odd values of *l*, we have:

$$\hat{\Omega}\bullet\nabla\psi^o(\mathbf{r},\hat{\Omega})+\sigma(\mathbf{r})\psi^e(\mathbf{r},\hat{\Omega})= \sum_{l=0,even}^{\infty}\frac{2l+1}{4\pi}\sigma_l(\mathbf{r})\int_{4\pi}P_l(\hat{\Omega}\bullet\hat{\Omega}')\psi^e(\mathbf{r},\hat{\Omega}')d\hat{\Omega}'+Q^e(\mathbf{r},\hat{\Omega}) \quad \textbf{4.19}$$

If instead we subtract Eq. **4.11** and Eq. **4.17** and make use of the same definitions, the following equation is obtained:

$$\hat{\Omega}\bullet\nabla\psi^e(\mathbf{r},\hat{\Omega})+\sigma(\mathbf{r})\psi^o(\mathbf{r},\hat{\Omega})= \sum_{l=1,odd}^{\infty}\frac{2l+1}{4\pi}\sigma_l(\mathbf{r})\int_{4\pi}P_l(\hat{\Omega}\bullet\hat{\Omega}')\psi^o(\mathbf{r},\hat{\Omega}')d\hat{\Omega}'+Q^o(\mathbf{r},\hat{\Omega}) \quad \textbf{4.20}$$

Define even and odd removal operators R_e and R_o, respectively, analogously to the removal operator defined for the full angular flux:

$$R_e\psi^e(\mathbf{r},\hat{\Omega})\equiv\sigma(\mathbf{r})\psi^e(\mathbf{r},\hat{\Omega})-\sum_{l=0,even}^{\infty}\frac{2l+1}{4\pi}\sigma_l(\mathbf{r})\int_{4\pi}P_l(\hat{\Omega}\bullet\hat{\Omega}')\psi^e(\mathbf{r},\hat{\Omega}')d\hat{\Omega}' \quad \textbf{4.21}$$

$$R_o\psi^o(\mathbf{r},\hat{\Omega})\equiv\sigma(\mathbf{r})\psi^o(\mathbf{r},\hat{\Omega})-\sum_{l=1,odd}^{\infty}\frac{2l+1}{4\pi}\sigma_l(\mathbf{r})\int_{4\pi}P_l(\hat{\Omega}\bullet\hat{\Omega}')\psi^o(\mathbf{r},\hat{\Omega}')d\hat{\Omega}' \quad \textbf{4.22}$$

Using these definitions, Eq. **4.19** and Eq. **4.20** can be rewritten as:

$$\hat{\Omega}\bullet\nabla\psi^o(\mathbf{r},\hat{\Omega})+R_e\psi^e(\mathbf{r},\hat{\Omega})=Q^e(\mathbf{r},\hat{\Omega}) \quad \textbf{4.23}$$

$$\hat{\Omega}\bullet\nabla\psi^e(\mathbf{r},\hat{\Omega})+R_o\psi^o(\mathbf{r},\hat{\Omega})=Q^o(\mathbf{r},\hat{\Omega}) \quad \textbf{4.24}$$

To obtain the even-parity equation, solve Eq. **4.24** for $\psi^o(\mathbf{r},\hat{\Omega})$ and substitute the result into **4.23**:

$$\psi^o(\mathbf{r},\hat{\Omega})=R_o^{-1}\left[Q^o(\mathbf{r},\hat{\Omega})-\hat{\Omega}\bullet\nabla\psi^e(\mathbf{r},\hat{\Omega})\right] \quad \textbf{4.25}$$

$$-\hat{\Omega}\bullet\nabla R_o^{-1}\hat{\Omega}\bullet\nabla\psi^e(\mathbf{r},\hat{\Omega})+R_e\psi^e(\mathbf{r},\hat{\Omega})=Q^e(\mathbf{r},\hat{\Omega})-\hat{\Omega}\bullet\nabla R_o^{-1}Q^o(\mathbf{r},\hat{\Omega}) \qquad \textbf{4.26}$$

The odd-parity equation can be obtained in a similar manner by solving Eq. **4.23** for $\psi^e(\mathbf{r},\hat{\Omega})$ and substituting the result into Eq. **4.24**:

$$-\hat{\Omega}\bullet\nabla R_e^{-1}\hat{\Omega}\bullet\nabla\psi^o(\mathbf{r},\hat{\Omega})+R_o\psi^o(\mathbf{r},\hat{\Omega})=Q^o(\mathbf{r},\hat{\Omega})-\hat{\Omega}\bullet\nabla R_e^{-1}Q^e(\mathbf{r},\hat{\Omega}) \qquad \textbf{4.27}$$

The inverse removal operators R_e^{-1} and R_o^{-1} are similar in structure to the inverse removal operator associated with the SAAF equation and can be derived via a similar procedure.

$$\begin{aligned} R_e^{-1}f(\mathbf{r},\hat{\Omega}) &= \frac{f(\mathbf{r},\hat{\Omega})}{\sigma(\mathbf{r})} \\ &+\frac{1}{\sigma(\mathbf{r})}\sum_{l=0,even}^{\infty}\frac{\sigma_l(\mathbf{r})}{\sigma(\mathbf{r})-\sigma_l(\mathbf{r})}\sum_{m=-l}^{l}Y_{lm}(\hat{\Omega})\int_{4\pi}Y_{lm}^*(\hat{\Omega}')f(\mathbf{r},\hat{\Omega}')d\hat{\Omega}' \end{aligned} \qquad \textbf{4.28}$$

$$\begin{aligned} R_o^{-1}f(\mathbf{r},\hat{\Omega}) &= \frac{f(\mathbf{r},\hat{\Omega})}{\sigma(\mathbf{r})} \\ &+\frac{1}{\sigma(\mathbf{r})}\sum_{l=1,odd}^{\infty}\frac{\sigma_l(\mathbf{r})}{\sigma(\mathbf{r})-\sigma_l(\mathbf{r})}\sum_{m=-l}^{l}Y_{lm}(\hat{\Omega})\int_{4\pi}Y_{lm}^*(\hat{\Omega}')f(\mathbf{r},\hat{\Omega}')d\hat{\Omega}' \end{aligned} \qquad \textbf{4.29}$$

This procedure results in the decoupling of the even and odd angular dependence of the angular flux within the host medium. Due to the known symmetries in angle, this reduces the number of unknowns by half for each equation. In other words, instead of solving the transport equation over the whole unit sphere, it is sufficient to solve over only half of the unit sphere, reducing computation time. Depending on the quantities of interest, this can be a great advantage. For instance, in many transport calculations, the *scalar flux* is of interest. The scalar flux is defined as the integral over all angular directions of the angular flux:

$$\phi(\mathbf{r})\equiv\int_{4\pi}\psi(\mathbf{r},\hat{\Omega})d\hat{\Omega} \qquad \textbf{4.30}$$

and is used to calculate reaction density rates and energy deposition per unit volume at a given point within the problem domain. It is easily shown that the scalar flux can be calculated from the even component of the angular flux alone:

$$\begin{aligned}\phi(\mathbf{r},E,t) &\equiv \int_{4\pi} \psi(\mathbf{r},\hat{\Omega})d\hat{\Omega} \\ &= \int_{4\pi} \left[\psi^e(\mathbf{r},\hat{\Omega}) + \psi^o(\mathbf{r},\hat{\Omega})\right] d\hat{\Omega} \\ &= \int_{4\pi} \psi^e(\mathbf{r},\hat{\Omega})d\hat{\Omega}\end{aligned} \qquad \textbf{4.31}$$

since the integral of an odd function over its whole range is zero. Thus for calculations where the scalar flux is ultimately desired, it is only necessary to solve Eq. **4.26**.

If, in contrast, the rate at which particles cross particular boundaries in the problem domain is desired, it is necessary to calculate the *net current*. The net current is a vector quantity that is defined as the integral over the solid angle of the angular flux multiplied by the angular unit vector $\hat{\Omega}$:

$$\mathbf{J}(\mathbf{r}) \equiv \int_{4\pi} \hat{\Omega}\,\psi(\mathbf{r},\hat{\Omega})d\hat{\Omega} \qquad \textbf{4.32}$$

The unit vector $\hat{\Omega}$ is odd in the angular variables. To determine the current from the EOP expression of the angular flux, substitute Eq. **4.14** into Eq. **4.32** and perform the integration:

$$\begin{aligned}\mathbf{J}(\mathbf{r}) &\equiv \int_{4\pi} \hat{\Omega}\,\psi(\mathbf{r},\hat{\Omega})d\hat{\Omega} \\ &= \int_{4\pi} \hat{\Omega}\left[\psi^e(\mathbf{r},\hat{\Omega}) + \psi^o(\mathbf{r},\hat{\Omega})\right] d\hat{\Omega} \\ &= \int_{4\pi} \hat{\Omega}\,\psi^o(\mathbf{r},\hat{\Omega})d\hat{\Omega}\end{aligned} \qquad \textbf{4.33}$$

since the product of an even function and an odd function is odd, resulting in the integral evaluating to zero, while the product of an odd function and an odd function is even. This shows that if the current is required, one only needs to solve Eq. **4.27**. It is also possible to determine the net current from the even-parity flux, simply by using the previous equation and the relation given in Eq. **4.25**:

$$\mathbf{J}(\mathbf{r}) = \int_{4\pi} \hat{\Omega} R_o^{-1} \left[Q^o(\mathbf{r}, \hat{\Omega}) - \hat{\Omega} \bullet \nabla \psi^e(\mathbf{r}, \hat{\Omega}) \right] d\hat{\Omega} \qquad \textbf{4.34}$$

This shows that the quantities of common interest can be computed simply by solving over half the angular range for the even-parity angular flux

4.2.1 Boundary Conditions for EOP

Unlike the SAAF equation, where the boundary conditions are the same as those of the first order transport equation, special boundary conditions must be derived for the EOP equations [9]. In order to do this, start with prescribed boundary conditions:

$$\psi(\mathbf{r}_b, \hat{\Omega}) = f(\mathbf{r}_b, \hat{\Omega}) \text{ for } \mathbf{r}_b \in \partial V, \ \mathbf{n} \bullet \hat{\Omega} < 0 \qquad \textbf{4.35}$$

Write this equation in terms of the even and odd parity fluxes:

$$\psi(\mathbf{r}_b, \hat{\Omega}) = \psi^e(\mathbf{r}_b, \hat{\Omega}) + \psi^o(\mathbf{r}_b, \hat{\Omega}) = f(\mathbf{r}_b, \hat{\Omega}) \ , \ \mathbf{n} \bullet \hat{\Omega} < 0 \qquad \textbf{4.36}$$

Since the EOP formulation is second order in space, the fluxes in the outgoing directions must also be specified.

$$\psi(\mathbf{r}_b, -\hat{\Omega}) = \psi^e(\mathbf{r}_b, \hat{\Omega}) - \psi^o(\mathbf{r}_b, \hat{\Omega}) = f(\mathbf{r}_b, -\hat{\Omega}) \ , \ \mathbf{n} \bullet \hat{\Omega} > 0 \qquad \textbf{4.37}$$

To determine the even-parity conditions, use Eq. **4.25** in the previous two equations:

$$-R_o^{-1}\hat{\Omega}\bullet\nabla\psi^e(\mathbf{r}_b,\hat{\Omega})+\psi^e(\mathbf{r}_b,\hat{\Omega})=f(\mathbf{r}_b,\hat{\Omega})-R_o^{-1}Q^o(\mathbf{r}_b,\hat{\Omega})\ ,\ \mathbf{n}\bullet\hat{\Omega}<0 \qquad \textbf{4.38}$$

$$R_o^{-1}\hat{\Omega}\bullet\nabla\psi^e(\mathbf{r}_b,\hat{\Omega})+\psi^e(\mathbf{r}_b,\hat{\Omega})=f(\mathbf{r}_b,-\hat{\Omega})+R_o^{-1}Q^o(\mathbf{r}_b,\hat{\Omega})\ ,\ \mathbf{n}\bullet\hat{\Omega}>0 \qquad \textbf{4.39}$$

The odd parity boundary conditions can be obtained by solving Eq. **4.24** for $\psi^e(\mathbf{r},\hat{\Omega})$ and substituting the result into the prescribed boundary conditions:

$$-R_e^{-1}\hat{\Omega}\bullet\nabla\psi^o(\mathbf{r}_b,\hat{\Omega})+\psi^o(\mathbf{r}_b,\hat{\Omega})=f(\mathbf{r}_b,\hat{\Omega})-R_e^{-1}Q^e(\mathbf{r}_b,\hat{\Omega})\ ,\ \mathbf{n}\bullet\hat{\Omega}<0 \qquad \textbf{4.40}$$

$$R_e^{-1}\hat{\Omega}\bullet\nabla\psi^o(\mathbf{r}_b,\hat{\Omega})+\psi^o(\mathbf{r}_b,\hat{\Omega})=-f(\mathbf{r}_b,-\hat{\Omega})+R_e^{-1}Q^e(\mathbf{r}_b,\hat{\Omega})\ ,\ \mathbf{n}\bullet\hat{\Omega}>0 \qquad \textbf{4.41}$$

This gives us prescribed flux boundary conditions for the EOP formulation of the transport problem.

4.3 Finite Elements Approach to Second Order Forms

It is appropriate at this point to mention how the finite elements approach is used to perform the spatial discretization of the second order forms of the transport equation. Since this work focused mainly on the angular discretization, the approach for the spatial discretization will not be expounded upon in a great deal of detail; rather, a general framework will be shown for each of the second-order forms. More detailed information on the finite element method can be found in [24]. To allow for compactness in presentation, the following notation will be introduced:

$$\left\langle f(\mathbf{r})\right\rangle=\int_V f(\mathbf{r})dV$$

i.e., the brackets indicate integration over the volume of the problem domain.

4.3.1 Weak Form of the SAAF Equation

To apply the finite element method to the SAAF formulation, it is first necessary to derive the so-called "weak form" of the equation. First, starting with the SAAF equation (Eq. **4.4**):

$$-\hat{\Omega}\bullet\nabla R^{-1}\hat{\Omega}\bullet\nabla\psi(\mathbf{r},\hat{\Omega})+R\psi(\mathbf{r},\hat{\Omega})=Q(\mathbf{r},\hat{\Omega})-\hat{\Omega}\bullet\nabla R^{-1}Q(\mathbf{r},\hat{\Omega})$$

Next, multiplying this equation by a trial function $u(\mathbf{r})$:

$$-u(\mathbf{r})\hat{\Omega}\bullet\nabla R^{-1}\hat{\Omega}\bullet\nabla\psi(\mathbf{r},\hat{\Omega})+u(\mathbf{r})R\psi(\mathbf{r},\hat{\Omega})=u(\mathbf{r})Q(\mathbf{r},\hat{\Omega})-u(\mathbf{r})\hat{\Omega}\bullet\nabla R^{-1}Q(\mathbf{r},\hat{\Omega})$$

Integrating over the problem domain and applying the divergence theorem to distribute the derivative terms leads to the following equation:

$$\begin{aligned}&\left\langle\hat{\Omega}\bullet\nabla u(\mathbf{r})R^{-1}\hat{\Omega}\bullet\nabla\psi(\mathbf{r},\hat{\Omega})\right\rangle+\left\langle u(\mathbf{r})R\psi(\mathbf{r},\hat{\Omega})\right\rangle-\oint_S u(\mathbf{r})R^{-1}\hat{\Omega}\bullet\nabla\psi(\mathbf{r},\hat{\Omega})\hat{\Omega}\bullet\mathbf{n}d\mathbf{s}\\&=\left\langle u(\mathbf{r})Q(\mathbf{r},\hat{\Omega})\right\rangle+\left\langle\hat{\Omega}\bullet\nabla u(\mathbf{r})R^{-1}Q(\mathbf{r},\hat{\Omega})\right\rangle-\oint_S u(\mathbf{r})R^{-1}Q(\mathbf{r},\hat{\Omega})\hat{\Omega}\bullet\mathbf{n}d\mathbf{s}\end{aligned}$$

where $\oint_S\circ\hat{\Omega}\bullet\mathbf{n}d\mathbf{s}$ denotes a surface integral over the boundary of the problem domain.

These surface integrals can be simplified using Eq. **4.2**:

$$\begin{aligned}&\left\langle\hat{\Omega}\bullet\nabla u(\mathbf{r})R^{-1}\hat{\Omega}\bullet\nabla\psi(\mathbf{r},\hat{\Omega})\right\rangle+\left\langle u(\mathbf{r})R\psi(\mathbf{r},\hat{\Omega})\right\rangle+\oint_S u(\mathbf{r})\psi(\mathbf{r},\hat{\Omega})\hat{\Omega}\bullet\mathbf{n}d\mathbf{s}\\&=\left\langle u(\mathbf{r})Q(\mathbf{r},\hat{\Omega})\right\rangle+\left\langle\hat{\Omega}\bullet\nabla u(\mathbf{r})R^{-1}Q(\mathbf{r},\hat{\Omega})\right\rangle\end{aligned}$$

This is the weak form of the SAAF equation. To carry out the finite element discretization, the spatial dependence of the angular flux would be expanded into a finite series of basis functions. The number of terms in this expansion is dependent upon the required degree of accuracy. Orthogonality between the trial function $u(\mathbf{r})$ and the basis functions would result in a set of algebraic equations that would approximate the spatial

dependence of the angular flux distribution. The boundary conditions specified for a given problem are imbedded in the surface integral present in the weak form of the SAAF equation. Suppose a prescribed boundary condition is employed, i.e.,

$$\psi(\mathbf{r}_b,\hat{\Omega}) = f(\mathbf{r}_b,\hat{\Omega}),\ \mathbf{r}_b \in S,\ \mathbf{n}\bullet\hat{\Omega} < 0$$

To implement this boundary condition in a spherical harmonics solution, the boundary integral present in the weak form of the SAAF equation is split to separate the flux for outgoing directions (which are unknown) from the incoming directions (specified by the boundary condition). In this case, the flux on the boundary for incoming directions becomes part of the source term, and the following weak form is employed:

$$\left\langle \hat{\Omega}\bullet\nabla u(\mathbf{r})R^{-1}\hat{\Omega}\bullet\nabla\psi(\mathbf{r},\hat{\Omega})\right\rangle + \left\langle u(\mathbf{r})R\psi(\mathbf{r},\hat{\Omega})\right\rangle + \oint_{S,\hat{\Omega}\bullet\mathbf{n}>0} u(\mathbf{r})\psi(\mathbf{r},\hat{\Omega})\left|\hat{\Omega}\bullet\mathbf{n}\right| d\mathbf{s}$$
$$= \left\langle u(\mathbf{r})Q(\mathbf{r},\hat{\Omega})\right\rangle + \left\langle \hat{\Omega}\bullet\nabla u(\mathbf{r})R^{-1}Q(\mathbf{r},\hat{\Omega})\right\rangle + \oint_{S,\hat{\Omega}\bullet\mathbf{n}<0} u(\mathbf{r})f(\mathbf{r},\hat{\Omega})\left|\hat{\Omega}\bullet\mathbf{n}\right| d\mathbf{s}$$

4.3.2 Weak Form of the EOP Equations

The weak form of the EOP equations can be derived in a completely analogous manner as the weak form of the SAAF equations; i.e., first multiply each equation by a trial function, integrate over space, use the divergence theorem to distribute derivative terms, and then simplify the boundary terms using the appropriate relations. This procedure leads to the weak forms of the even- and odd-parity equations, respectively:

$$\left\langle \hat{\Omega}\bullet\nabla u(\mathbf{r})R_o^{-1}\hat{\Omega}\bullet\nabla\psi^e(\mathbf{r},\hat{\Omega})\right\rangle+\left\langle u(\mathbf{r})R_e\psi^e(\mathbf{r},\hat{\Omega})\right\rangle+\oint_S u(\mathbf{r})\psi^o(\mathbf{r},\hat{\Omega})\hat{\Omega}\bullet\mathbf{n}d\mathbf{s}$$
$$=\left\langle u(\mathbf{r})Q^e(\mathbf{r},\hat{\Omega})\right\rangle+\left\langle \hat{\Omega}\bullet\nabla u(\mathbf{r})R_o^{-1}Q^o(\mathbf{r},\hat{\Omega})\right\rangle$$
$$\left\langle \hat{\Omega}\bullet\nabla u(\mathbf{r})R_e^{-1}\hat{\Omega}\bullet\nabla\psi^o(\mathbf{r},\hat{\Omega})\right\rangle+\left\langle u(\mathbf{r})R_o\psi^o(\mathbf{r},\hat{\Omega})\right\rangle+\oint_S u(\mathbf{r})\psi^e(\mathbf{r},\hat{\Omega})\hat{\Omega}\bullet\mathbf{n}d\mathbf{s}$$
$$=\left\langle u(\mathbf{r})Q^o(\mathbf{r},\hat{\Omega})\right\rangle+\left\langle \hat{\Omega}\bullet\nabla u(\mathbf{r})R_e^{-1}Q^e(\mathbf{r},\hat{\Omega})\right\rangle$$

The coupling between the even- and odd-parity forms at the boundaries of the problem domain is evident in the surface integrals shown above. To deal with this in the weak form, assume the prescribed boundary conditions as given by Eqs. **4.36-37**. Using these boundary conditions, it is determined that:

$$\psi^e(\mathbf{r},\hat{\Omega})=\psi^o(\mathbf{r},\hat{\Omega})+f(\mathbf{r},-\hat{\Omega}),\ \mathbf{n}\bullet\hat{\Omega}>0$$
$$\psi^e(\mathbf{r},\hat{\Omega})=-\psi^o(\mathbf{r},\hat{\Omega})+f(\mathbf{r},\hat{\Omega}),\ \mathbf{n}\bullet\hat{\Omega}<0$$
$$\psi^o(\mathbf{r},\hat{\Omega})=\psi^e(\mathbf{r},\hat{\Omega})-f(\mathbf{r},-\hat{\Omega}),\ \mathbf{n}\bullet\hat{\Omega}>0$$
$$\psi^o(\mathbf{r},\hat{\Omega})=-\psi^e(\mathbf{r},\hat{\Omega})+f(\mathbf{r},\hat{\Omega}),\ \mathbf{n}\bullet\hat{\Omega}<0$$

Using these relations, it is possible to derive weak forms for the even- and odd-parity equations which are not coupled at the boundaries. The weak form of the even-parity equation is given by:

$$\left\langle \hat{\Omega}\bullet\nabla u(\mathbf{r})R_o^{-1}\hat{\Omega}\bullet\nabla\psi^e(\mathbf{r},\hat{\Omega})\right\rangle+\left\langle u(\mathbf{r})R_e\psi^e(\mathbf{r},\hat{\Omega})\right\rangle+\oint_S u(\mathbf{r})\psi^e(\mathbf{r},\hat{\Omega})\left|\hat{\Omega}\bullet\mathbf{n}\right|d\mathbf{s}$$
$$=\left\langle u(\mathbf{r})Q^e(\mathbf{r},\hat{\Omega})\right\rangle+\left\langle \hat{\Omega}\bullet\nabla u(\mathbf{r})R_o^{-1}Q^o(\mathbf{r},\hat{\Omega})\right\rangle+\oint_S u(\mathbf{r})f(\mathbf{r},-\hat{\Omega})\left|\hat{\Omega}\bullet\mathbf{n}\right|d\mathbf{s},\ \hat{\Omega}\bullet\mathbf{n}>0$$
$$\left\langle \hat{\Omega}\bullet\nabla u(\mathbf{r})R_o^{-1}\hat{\Omega}\bullet\nabla\psi^e(\mathbf{r},\hat{\Omega})\right\rangle+\left\langle u(\mathbf{r})R_e\psi^e(\mathbf{r},\hat{\Omega})\right\rangle+\oint_S u(\mathbf{r})\psi^e(\mathbf{r},\hat{\Omega})\left|\hat{\Omega}\bullet\mathbf{n}\right|d\mathbf{s}$$
$$=\left\langle u(\mathbf{r})Q^e(\mathbf{r},\hat{\Omega})\right\rangle+\left\langle \hat{\Omega}\bullet\nabla u(\mathbf{r})R_o^{-1}Q^o(\mathbf{r},\hat{\Omega})\right\rangle+\oint_S u(\mathbf{r})f(\mathbf{r},\hat{\Omega})\left|\hat{\Omega}\bullet\mathbf{n}\right|d\mathbf{s},\ \hat{\Omega}\bullet\mathbf{n}<0$$

The weak form of the odd-parity equation is given by:

$$\left\langle \hat{\Omega}\bullet\nabla u(\mathbf{r})R_e^{-1}\hat{\Omega}\bullet\nabla\psi^o(\mathbf{r},\hat{\Omega})\right\rangle+\left\langle u(\mathbf{r})R_o\psi^o(\mathbf{r},\hat{\Omega})\right\rangle+\oint_S u(\mathbf{r})\psi^o(\mathbf{r},\hat{\Omega})\left|\hat{\Omega}\bullet\mathbf{n}\right|d\mathbf{s}$$

$$=\left\langle u(\mathbf{r})Q^o(\mathbf{r},\hat{\Omega})\right\rangle+\left\langle \hat{\Omega}\bullet\nabla u(\mathbf{r})R_e^{-1}Q^e(\mathbf{r},\hat{\Omega})\right\rangle-\oint_S u(\mathbf{r})f(\mathbf{r},-\hat{\Omega})\left|\hat{\Omega}\bullet\mathbf{n}\right|d\mathbf{s},\ \hat{\Omega}\bullet\mathbf{n}>0$$

$$\left\langle \hat{\Omega}\bullet\nabla u(\mathbf{r})R_e^{-1}\hat{\Omega}\bullet\nabla\psi^o(\mathbf{r},\hat{\Omega})\right\rangle+\left\langle u(\mathbf{r})R_o\psi^o(\mathbf{r},\hat{\Omega})\right\rangle+\oint_S u(\mathbf{r})\psi^o(\mathbf{r},\hat{\Omega})\left|\hat{\Omega}\bullet\mathbf{n}\right|d\mathbf{s}$$

$$=\left\langle u(\mathbf{r})Q^o(\mathbf{r},\hat{\Omega})\right\rangle+\left\langle \hat{\Omega}\bullet\nabla u(\mathbf{r})R_e^{-1}Q^e(\mathbf{r},\hat{\Omega})\right\rangle-\oint_S u(\mathbf{r})f(\mathbf{r},\hat{\Omega})\left|\hat{\Omega}\bullet\mathbf{n}\right|d\mathbf{s},\ \hat{\Omega}\bullet\mathbf{n}<0$$

The finite element method is applied to these equations in an analogous procedure to the SAAF equation.

Now that the second-order transport equations are defined, we will move to deriving the spherical harmonics angular approximations that are solved within the SCEPTRE framework.

Chapter 5

Spherical Harmonics Discretization of Second Order Forms

In the previous chapter, the continuous second-order formulation of the transport equation was defined. Next, our attention is turned towards the heart of this work, which is the spherical harmonics approximation to treat the angular variables.

5.1 Description of Spherical Harmonics Discretization

The spherical harmonics method has been employed since the early days of transport theory as a way to solve for the angular dependence of the angular flux ([10], [12]). The spherical harmonics functions themselves arise from the solution to the angular part of Laplace's equation in the spherical coordinate system. They are complete on the unit sphere, meaning that any function taking values on the unit sphere can be represented as an infinite sum of the spherical harmonics scaled by some constants.

It is assumed that the angular flux can be represented as an infinite series of spherical harmonics and spatial moments, i.e.,

$$\psi(\mathbf{r},\hat{\Omega}) = \sum_{l=0}^{\infty}\sum_{m=-l}^{l} \phi_{l,m}(\mathbf{r}) Y_{l,m}(\hat{\Omega}) \qquad \textbf{5.1}$$

where $\phi_{l,m}(\mathbf{r})$ is the spatial moment corresponding to the l^{th} and m^{th} spherical harmonic and $\hat{\Omega}$ is the unit vector comprising of the polar angle θ and the azimuthal angle φ. These moments are given by

$$\phi_{l,m}(\mathbf{r}) = \int_{4\pi} \psi(\mathbf{r},\hat{\Omega}) Y^{*}_{l,m}(\hat{\Omega}) d\hat{\Omega} \qquad \textbf{5.2}$$

where $Y^{*}_{l,m}(\hat{\Omega})$ is the complex conjugate of $Y_{l,m}(\hat{\Omega})$. In this way, the angular and spatial dependence of the angular flux are separated. It is assumed that an accurate approximation for the angular dependence of the angular flux can be obtained by truncating the infinite series in Eq. **5.1** to a finite series with order *L*; in other words, $\phi_{l,m}(\mathbf{r}) = 0$ for $l > L$. The orthogonality of the spherical harmonics over the unit sphere can be employed to arrive at a finite system of equations for $\phi_{l,m}(\mathbf{r})$, which can then be discretized in space and solved for. The angular flux and related quantities can then be recovered from these spatial moments.

The spherical harmonics are given by the following expression:

$$Y_{l,m}(\hat{\Omega}) = Y_{l,m}(\mu_z, \varphi) = \sqrt{\frac{2l+1}{4\pi}} \sqrt{\frac{(l-m)!}{(l+m)!}} P_l^m(\mu_z) e^{im\varphi} \qquad \textbf{5.3}$$

where μ_z is the cosine of the polar angle θ, φ is the azimuthal angle (see Fig. **5.1** for a description of the angular variables), and $P_l^m(\mu_z)$ is the associated Legendre polynomial, given by:

$$P_l^m(\mu_z) = \begin{cases} (-1)^m (1-\mu_z^2)^{m/2} \dfrac{d^m}{d\mu_z^m} P_l(\mu_z), & m \geq 0 \\ (-1)^m \dfrac{(l-m)!}{(l+m)!} P_l^{-m}(\mu_z), & m < 0 \end{cases}$$

The definition of the spherical harmonics is valid for $l = 0, 1, 2, ..., \infty$ and $-l \leq m \leq l$.

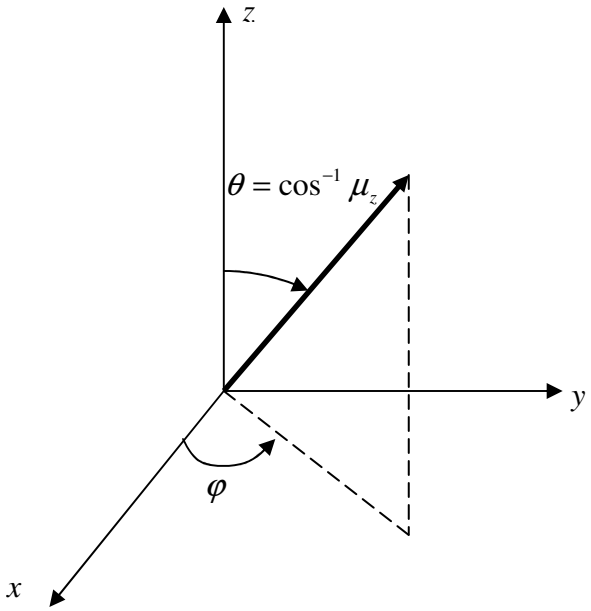

Figure **5.1**: Description of angular coordinate system

The spherical harmonics functions have the property that they are orthonormal over the unit sphere, i.e.,

$$\int_{4\pi} Y_{l,m}(\hat{\Omega}) Y^{*}_{l',m'}(\hat{\Omega}) d\hat{\Omega} = \delta_{ll'}\delta_{mm'}$$

Also, Legendre's Addition Theorem relates the Legendre polynomials in the dot product of two unit vectors to a finite series in their individual spherical harmonics:

$$P_l(\hat{\Omega}\bullet\hat{\Omega}') = \frac{4\pi}{2l+1}\sum_{m=-l}^{l} Y_{l,m}(\hat{\Omega}) Y^{*}_{l,m}(\hat{\Omega}')$$

The exponential present in Eq. **5.3** can be expanded into a combination of sines and cosines:

$$\begin{aligned} Y_{l,m}(\mu_z,\varphi) &= C_{l,m}P_l^m(\mu_z)\left[\cos(m\varphi)+i\sin(m\varphi)\right] \\ &= C_{l,m}P_l^m(\mu_z)\cos(m\varphi)+C_{l,m}P_l^m(\mu_z)\sin(m\varphi) \\ &= Y_{l,m}^c(\mu_z,\varphi)+iY_{l,m}^s(\mu_z,\varphi) \end{aligned} \quad \textbf{5.4}$$

where

$$C_{l,m}=\sqrt{\frac{2l+1}{4\pi}}\sqrt{\varepsilon_m}\sqrt{\frac{(l-m)!}{(l+m)!}}$$

and

$$\varepsilon_m = 2-\delta_{m0}$$

(where δ_{m0} is the Kronecker delta) is a normalization coefficient that ensures that the spherical harmonics are orthonormal when integrated over the unit sphere. SCEPTRE employs a sine/cosine formulation of the spherical harmonics functions [13], which shares the completeness properties of the standard spherical harmonics:

$$Y_{l,m}^c(\mu_z,\varphi)=\begin{cases} C_{l,0}P_l\ (\mu_z), & m=0 \\ C_{l,m}P_l^m(\mu_z)\cos(m\varphi), & m>0 \end{cases} \quad \textbf{5.5}$$

$$Y_{l,m}^s(\mu_z,\varphi)=\begin{cases} 0, & m=0 \\ C_{l,m}P_l^m(\mu_z)\sin(m\varphi), & m>0 \end{cases} \quad \textbf{5.6}$$

for $0\le m\le l$. It can be shown through simple computation that these equations satisfy the orthonormal relations

$$\int_{4\pi} Y_{l,m}^c(\hat{\Omega})Y_{l',m'}^c(\hat{\Omega})d\hat{\Omega}=\delta_{ll'}\delta_{mm'} \quad \textbf{5.7}$$

and

$$\int_{4\pi} Y_{l,m}^s(\hat{\Omega})Y_{l',m'}^s(\hat{\Omega})d\hat{\Omega}=(1-\delta_{m0})\delta_{ll'}\delta_{mm'} \quad \textbf{5.8}$$

Additionally,

$$\int_{4\pi} Y^c_{l,m}(\hat{\Omega}) Y^s_{l',m'}(\hat{\Omega}) d\hat{\Omega} = 0 \qquad \textbf{5.9}$$

It can also be shown that this formulation of the spherical harmonics satisfies the addition theorem given by

$$P_l(\hat{\Omega}\bullet\hat{\Omega}') = \frac{4\pi}{2l+1} \sum_{m=0}^{l} \left[Y^c_{l,m}(\hat{\Omega}) Y^c_{l,m}(\hat{\Omega}') + Y^s_{l,m}(\hat{\Omega}) Y^s_{l,m}(\hat{\Omega}') \right] \qquad \textbf{5.10}$$

which is used in the treatment of the scattering integral.

Instead of expanding the angular flux in terms of the standard spherical harmonics, as in Eq. **5.1**, the flux is expanded into the cosine and sine forms:

$$\psi(\mathbf{r}, \hat{\Omega}) = \sum_{l=0}^{L} \sum_{m=0}^{l} \left[\phi^c_{l,m}(\mathbf{r}) Y^c_{l,m}(\hat{\Omega}) + \phi^s_{l,m}(\mathbf{r}) Y^s_{l,m}(\hat{\Omega}) \right] \qquad \textbf{5.11}$$

where

$$\begin{aligned} \phi^c_{l,m}(\mathbf{r}) &= \int_{4\pi} \psi(\mathbf{r}, \hat{\Omega}) Y^c_{l,m}(\hat{\Omega}) d\hat{\Omega} \\ \phi^s_{l,m}(\mathbf{r}) &= \int_{4\pi} \psi(\mathbf{r}, \hat{\Omega}) Y^s_{l,m}(\hat{\Omega}) d\hat{\Omega} \end{aligned} \qquad \textbf{5.12}$$

To arrive at the spherical harmonics equations for the SAAF formulation of the second-order problem, this expansion is first substituted into the transport equation. By making use of recursion relationships present in the first term on the left hand side of Eq. **4.4** (describing the particle motion), the spatial moments of different orders *l* and *m* are coupled together. Then, the transport equation is multiplied on both sides by $Y^c_{l,m}(\hat{\Omega})$ and integrated over the angular variables. By using the orthonormality of the spherical harmonics, the angular dependence is integrated out, and the resulting set of equations consists of unknowns, namely, the spherical harmonics moments, that are functions only of space. A second set of equations is arrived at by multiplying each side of the transport

equation by $Y_{l,m}^{s}(\hat{\Omega})$ and again integrating over the angular variable. The spherical harmonics approximation to the SAAF equation is the two sets of equations for the spatial moments as described above. The spherical harmonics approximation to the EOP equations can be found by applying the same procedure to Eq. **4.23** and Eq. **4.24**.

5.2 Expansion of Second Order Forms

As shown in the previous chapter, the SAAF formulation of the transport equation using operator notation is given by Eq. **5.13**

$$-\hat{\Omega}\bullet\nabla R^{-1}\hat{\Omega}\bullet\nabla\psi(\mathbf{r},\hat{\Omega})+R\psi(\mathbf{r},\hat{\Omega})=Q(\mathbf{r},\hat{\Omega})-\hat{\Omega}\bullet\nabla R^{-1}Q(\mathbf{r},\hat{\Omega}) \qquad \textbf{5.13}$$

SCEPTRE employs a finite element spatial discretization scheme in three-dimensional Cartesian geometry. The gradient operator in this scheme is written as

$$\nabla\circ=\frac{\partial}{\partial x}\circ\hat{\mathbf{i}}+\frac{\partial}{\partial y}\circ\hat{\mathbf{j}}+\frac{\partial}{\partial z}\circ\hat{\mathbf{k}},$$

where $\hat{\mathbf{i}}$, $\hat{\mathbf{j}}$, and $\hat{\mathbf{k}}$ are the unit vectors in the *x, y* and *z* directions, respectively, defined by a given Cartesian coordinate system. The streaming operator $\hat{\Omega}\bullet\nabla$ is given by

$$\hat{\Omega}\bullet\nabla\circ=\mu_x\frac{\partial}{\partial x}\circ+\mu_y\frac{\partial}{\partial y}\circ+\mu_z\frac{\partial}{\partial z}\circ$$

where

$$\mu_x=\cos\varphi\sin\theta$$
$$\mu_y=\sin\varphi\sin\theta$$
$$\mu_z=\cos\theta$$

We now use these operators in Eq. **5.13** to develop the explicit version of the SAAF equation solved by SCEPTRE. The EOP equations are developed in a similar manner, as the structure of the operators is almost the same as the SAAF operators. For the sake of convenience, we will perform this expansion term-by-term.

The first term on the left hand side of Eq. **5.13** expands as:

$$-\hat{\Omega}\bullet\nabla R^{-1}\hat{\Omega}\bullet\nabla\psi(\mathbf{r},\hat{\Omega}) = -\hat{\Omega}\bullet\nabla R^{-1}\left[\mu_x\frac{\partial}{\partial x}\psi(\mathbf{r},\hat{\Omega}) + \mu_y\frac{\partial}{\partial y}\psi(\mathbf{r},\hat{\Omega}) + \mu_z\frac{\partial}{\partial z}\psi(\mathbf{r},\hat{\Omega})\right]$$

Next, insert the explicit form of the inverse removal operator as given in the previous chapter, with the assumption that the cross sections are constant in space, which is valid because in the spatial discretization employed by SCEPTRE the cross section is constant within each element. We expand this expression term by term with $i \in \{x, y, z\}$:

$$\hat{\Omega}\bullet\nabla R^{-1}\mu_i\frac{\partial}{\partial i}\psi(\mathbf{r},\hat{\Omega}) = \hat{\Omega}\bullet\nabla\left[\frac{\partial}{\partial i}\frac{\mu_i}{\sigma}\psi(\mathbf{r},\hat{\Omega}) + \frac{\partial}{\partial i}\frac{1}{\sigma}\sum_{l=0}^{\infty}\frac{2l+1}{4\pi}\frac{\sigma_l}{\sigma-\sigma_l}\int_{4\pi}P_l(\hat{\Omega}\bullet\hat{\Omega}')\mu_i'\psi(\mathbf{r},\hat{\Omega}')d\hat{\Omega}'\right]$$

Now, apply the streaming operator to the expression in brackets to arrive at the terms of the expansion of the first term of the SAAF equation:

$$\mu_j\frac{\partial}{\partial j}R^{-1}\mu_i\frac{\partial}{\partial i}\psi(\mathbf{r},\hat{\Omega}) =$$
$$\frac{1}{\sigma}\frac{\partial^2}{\partial i\partial j}\mu_i\mu_j\psi(\mathbf{r},\hat{\Omega}) + \frac{1}{\sigma}\mu_j\frac{\partial^2}{\partial i\partial j}\sum_{l=0}^{\infty}\frac{2l+1}{4\pi}\frac{\sigma_l}{\sigma-\sigma_l}\int_{4\pi}P_l(\hat{\Omega}\bullet\hat{\Omega}')\mu_i'\psi(\mathbf{r},\hat{\Omega}')d\hat{\Omega}',$$

$i, j \in \{x, y, z\}$. Thus, the complete first term can be written as

$$-\hat{\Omega}\bullet\nabla R^{-1}\hat{\Omega}\bullet\nabla\psi(\mathbf{r},\hat{\Omega}) = -\sum_{\substack{i\in\{x,y,z\}\\ j\in\{x,y,z\}}}\left[\frac{1}{\sigma}\frac{\partial^2}{\partial i\partial j}\mu_i\mu_j\psi(\mathbf{r},\hat{\Omega}) + \frac{1}{\sigma}\mu_j\frac{\partial^2}{\partial i\partial j}\sum_{l=0}^{\infty}\frac{2l+1}{4\pi}\frac{\sigma_l}{\sigma-\sigma_l}\int_{4\pi}P_l(\hat{\Omega}\bullet\hat{\Omega}')\mu_i'\psi(\mathbf{r},\hat{\Omega}')d\hat{\Omega}'\right] \quad \textbf{5.14}$$

Note that the first term on the left hand side of Eq. **5.13** consists of eighteen subterms resulting from the double summation over the three spatial variables.

The second term on the left hand side of Eq. **5.13** is straightforward to expand:

$$R\psi(\mathbf{r},\hat{\Omega}) = \sigma\psi(\mathbf{r},\hat{\Omega}) - \sum_{l=0}^{\infty}\frac{2l+1}{4\pi}\sigma_l \int_{4\pi} P_l(\hat{\Omega}\bullet\hat{\Omega}')\psi(\mathbf{r},\hat{\Omega}')d\hat{\Omega}' \qquad \textbf{5.15}$$

The first term on the right hand side of Eq. **5.13** requires no expansion. Finally, we expand the second term on the right hand side of Eq. **5.13**:

$$\begin{aligned} -\hat{\Omega}\bullet\nabla R^{-1}Q(\mathbf{r},\hat{\Omega}) &= -\hat{\Omega}\bullet\nabla\left[\frac{Q(\mathbf{r},\hat{\Omega})}{\sigma} + \frac{1}{\sigma}\sum_{l=0}^{\infty}\frac{2l+1}{4\pi}\frac{\sigma_l}{\sigma-\sigma_l}\int_{4\pi} P_l(\hat{\Omega}\bullet\hat{\Omega}')Q(\mathbf{r},\hat{\Omega}')d\hat{\Omega}'\right] \\ &= -\sum_{i\in\{x,y,z\}}\left[\frac{1}{\sigma}\frac{\partial}{\partial i}\mu_i Q(\mathbf{r},\hat{\Omega})\right. \\ &\quad \left. + \frac{1}{\sigma}\frac{\partial}{\partial i}\mu_i\sum_{l=0}^{\infty}\frac{2l+1}{4\pi}\frac{\sigma_l}{\sigma-\sigma_l}\int_{4\pi} P_l(\hat{\Omega}\bullet\hat{\Omega}')Q(\mathbf{r},\hat{\Omega}')d\hat{\Omega}'\right] \end{aligned} \qquad \textbf{5.16}$$

The second term on the right hand side of Eq. **5.13** has a total of six subterms from the summation over the three spatial variables.

Using these expansions, Eq. **5.13** is written as

$$\begin{aligned} &-\sum_{\substack{i\in\{x,y,z\}\\ j\in\{x,y,z\}}}\left[\frac{1}{\sigma}\frac{\partial^2}{\partial i\partial j}\mu_i\mu_j\psi(\mathbf{r},\hat{\Omega}) + \frac{1}{\sigma}\mu_j\frac{\partial^2}{\partial i\partial j}\sum_{l''=0}^{\infty}\frac{2l''+1}{4\pi}\frac{\sigma_{l''}}{\sigma-\sigma_{l''}}\int_{4\pi} P_{l''}(\hat{\Omega}\bullet\hat{\Omega}')\mu_i'\psi(\mathbf{r},\hat{\Omega}')d\hat{\Omega}'\right] \\ &+\sigma\psi(\mathbf{r},\hat{\Omega}) - \sum_{l=0}^{\infty}\frac{2l''+1}{4\pi}\sigma_{l''}\int_{4\pi} P_{l''}(\hat{\Omega}\bullet\hat{\Omega}')\psi(\mathbf{r},\hat{\Omega}')d\hat{\Omega}' \\ &= Q(\mathbf{r},\hat{\Omega}) - \sum_{i\in\{x,y,z\}}\left[\frac{1}{\sigma}\frac{\partial}{\partial i}\mu_i Q(\mathbf{r},\hat{\Omega}) + \frac{1}{\sigma}\frac{\partial}{\partial i}\mu_i\sum_{l''=0}^{\infty}\frac{2l''+1}{4\pi}\frac{\sigma_{l''}}{\sigma-\sigma_{l''}}\int_{4\pi} P_{l''}(\hat{\Omega}\bullet\hat{\Omega}')Q(\mathbf{r},\hat{\Omega}')d\hat{\Omega}'\right] \end{aligned} \qquad \textbf{5.17}$$

We have replaced the index l in the second and the fourth term on the left hand side with l'' for reasons that will become apparent shortly. Eq. **5.17** is the SAAF equation expanded into Cartesian geometry. To arrive at the spherical harmonics equations, we

expand the flux as given in Eq. **5.11**, multiply each term in the equation by either $Y^c_{l,m}(\hat{\Omega})$ or $Y^s_{l,m}(\hat{\Omega})$ and integrate over the angular variables. Again, we will present this manipulation term-by-term for convenience. In the course of this manipulation recursion relations characteristic of the spherical harmonics will be used in order to benefit from the orthonormality of the spherical harmonics.

Start with the first term on the left hand side of Eq. **5.17**, $i, j \in \{x, y, z\}$.

Substitute the flux expansion Eq. **5.11**:

$$\begin{aligned}\frac{1}{\sigma}\frac{\partial^2}{\partial i \partial j}\mu_i\mu_j\psi(\mathbf{r},\hat{\Omega}) &= \frac{1}{\sigma}\frac{\partial^2}{\partial i \partial j}\mu_i\mu_j\sum_{l=0}^{L}\sum_{m=0}^{l}\left[\phi^c_{l,m}(\mathbf{r})Y^c_{l,m}(\hat{\Omega})+\phi^s_{l,m}(\mathbf{r})Y^s_{l,m}(\hat{\Omega})\right] \\ &= \frac{1}{\sigma}\frac{\partial^2}{\partial i \partial j}\sum_{l=0}^{L}\sum_{m=0}^{l}\left[\mu_i\mu_j\phi^c_{l,m}(\mathbf{r})Y^c_{l,m}(\hat{\Omega})+\mu_i\mu_j\phi^s_{l,m}(\mathbf{r})Y^s_{l,m}(\hat{\Omega})\right]\end{aligned}$$

Multiply this term by $Y^{\alpha}_{l',m'}(\hat{\Omega})$ (with $\alpha \in \{c, s\}$)and integrate:

$$\begin{aligned}\int_{4\pi} Y^{\alpha}_{l',m'}(\hat{\Omega})\frac{1}{\sigma}\frac{\partial^2}{\partial i \partial j}\mu_i\mu_j\psi(\mathbf{r},\hat{\Omega})d\hat{\Omega} &= \int_{4\pi} Y^{\alpha}_{l',m'}(\hat{\Omega})\frac{1}{\sigma}\frac{\partial^2}{\partial i \partial j}\sum_{l=0}^{L}\sum_{m=0}^{l}\begin{bmatrix}\mu_i\mu_j\phi^c_{l,m}(\mathbf{r})Y^c_{l,m}(\hat{\Omega}) \\ +\mu_i\mu_j\phi^s_{l,m}(\mathbf{r})Y^s_{l,m}(\hat{\Omega})\end{bmatrix}d\hat{\Omega} \\ &= \frac{1}{\sigma}\frac{\partial^2}{\partial i \partial j}\sum_{l=0}^{L}\sum_{m=0}^{l}\left[\phi^c_{l,m}(\mathbf{r})\int_{4\pi}\mu_i\mu_j Y^c_{l,m}(\hat{\Omega})Y^{\alpha}_{l',m'}(\hat{\Omega})d\hat{\Omega}\right. \\ &\quad \left.+\phi^s_{l,m}(\mathbf{r})\int_{4\pi}\mu_i\mu_j Y^s_{l,m}(\hat{\Omega})Y^{\alpha}_{l',m'}(\hat{\Omega})d\hat{\Omega}\right]\end{aligned}$$

The first term in expanded notation is thus:

$$\begin{aligned}-\sum_{\substack{i\in\{x,y,z\}\\ j\in\{x,y,z\}}}\frac{1}{\sigma}\frac{\partial^2}{\partial i \partial j}\sum_{l=0}^{L}\sum_{m=0}^{l}\left[\phi^c_{l,m}(\mathbf{r})\int_{4\pi}\mu_i\mu_j Y^c_{l,m}(\hat{\Omega})Y^{\alpha}_{l',m'}(\hat{\Omega})d\hat{\Omega}\right. \\ \left.+\phi^s_{l,m}(\mathbf{r})\int_{4\pi}\mu_i\mu_j Y^s_{l,m}(\hat{\Omega})Y^{\alpha}_{l',m'}(\hat{\Omega})d\hat{\Omega}\right]\end{aligned} \qquad \textbf{5.18}$$

Repeating with the second term of Eq. **5.17**:

$$\frac{1}{\sigma}\mu_j\frac{\partial^2}{\partial i\partial j}\sum_{l''=0}^{\infty}\frac{2l''+1}{4\pi}\frac{\sigma_{l''}}{\sigma-\sigma_{l''}}\int_{4\pi}P_{l''}(\hat{\Omega}\bullet\hat{\Omega}')\mu_i'\psi(\mathbf{r},\hat{\Omega}')d\hat{\Omega}'$$

$$=\frac{1}{\sigma}\mu_j\frac{\partial^2}{\partial i\partial j}\sum_{l''=0}^{\infty}\frac{2l''+1}{4\pi}\frac{\sigma_{l''}}{\sigma-\sigma_{l''}}\int_{4\pi}P_{l''}(\hat{\Omega}\bullet\hat{\Omega}')\mu_i'\sum_{l=0}^{L}\sum_{m=0}^{l}\left[\phi_{l,m}^{c}(\mathbf{r})Y_{l,m}^{c}(\hat{\Omega}')+\phi_{l,m}^{s}(\mathbf{r})Y_{l,m}^{s}(\hat{\Omega}')\right]d\hat{\Omega}'$$

$$=\frac{1}{\sigma}\mu_j\frac{\partial^2}{\partial i\partial j}\sum_{l''=0}^{\infty}\frac{2l''+1}{4\pi}\frac{\sigma_{l''}}{\sigma-\sigma_{l''}}\sum_{l=0}^{L}\sum_{m=0}^{l}\int_{4\pi}P_{l''}(\hat{\Omega}\bullet\hat{\Omega}')\left[\phi_{l,m}^{c}(\mathbf{r})\mu_i'Y_{l,m}^{c}(\hat{\Omega}')+\phi_{l,m}^{s}(\mathbf{r})\mu_i'Y_{l,m}^{s}(\hat{\Omega}')\right]d\hat{\Omega}'$$

Using the addition theorem Eq. **5.10**:

$$\frac{1}{\sigma}\mu_j\frac{\partial^2}{\partial i\partial j}\sum_{l''=0}^{\infty}\frac{2l''+1}{4\pi}\frac{\sigma_{l''}}{\sigma-\sigma_{l''}}\sum_{l=0}^{L}\sum_{m=0}^{l}\int_{4\pi}P_{l''}(\hat{\Omega}\bullet\hat{\Omega}')\left[\phi_{l,m}^{c}(\mathbf{r})\mu_i'Y_{l,m}^{c}(\hat{\Omega}')+\phi_{l,m}^{s}(\mathbf{r})\mu_i'Y_{l,m}^{s}(\hat{\Omega}')\right]d\hat{\Omega}'$$

$$=\frac{1}{\sigma}\mu_j\frac{\partial^2}{\partial i\partial j}\sum_{l''=0}^{\infty}\sum_{m''=0}^{l''}\sum_{l=0}^{L}\sum_{m=0}^{l}\frac{\sigma_{l''}}{\sigma-\sigma_{l''}}\int_{4\pi}\left[Y_{l'',m''}^{c}(\hat{\Omega})Y_{l'',m''}^{c}(\hat{\Omega}')+Y_{l'',m''}^{s}(\hat{\Omega})Y_{l'',m''}^{s}(\hat{\Omega}')\right]$$

$$\times\left[\phi_{l,m}^{c}(\mathbf{r})\mu_i'Y_{l,m}^{c}(\hat{\Omega}')+\phi_{l,m}^{s}(\mathbf{r})\mu_i'Y_{l,m}^{s}(\hat{\Omega}')\right]d\hat{\Omega}'$$

$$=\frac{1}{\sigma}\mu_j\frac{\partial^2}{\partial i\partial j}\sum_{l''=0}^{\infty}\sum_{m''=0}^{l''}\sum_{l=0}^{L}\sum_{m=0}^{l}\frac{\sigma_{l''}}{\sigma-\sigma_{l''}}\int_{4\pi}\left[\phi_{l,m}^{c}(\mathbf{r})\mu_i'Y_{l,m}^{c}(\hat{\Omega}')Y_{l'',m''}^{c}(\hat{\Omega})Y_{l'',m''}^{c}(\hat{\Omega}')\right.$$

$$+\phi_{l,m}^{c}(\mathbf{r})\mu_i'Y_{l,m}^{c}(\hat{\Omega}')Y_{l'',m''}^{s}(\hat{\Omega})Y_{l'',m''}^{s}(\hat{\Omega}')+\phi_{l,m}^{s}(\mathbf{r})\mu_i'Y_{l,m}^{s}(\hat{\Omega}')Y_{l'',m''}^{c}(\hat{\Omega})Y_{l'',m''}^{c}(\hat{\Omega}')$$

$$\left.+\phi_{l,m}^{s}(\mathbf{r})\mu_i'Y_{l,m}^{s}(\hat{\Omega}')Y_{l'',m''}^{s}(\hat{\Omega})Y_{l'',m''}^{s}(\hat{\Omega}')\right]d\hat{\Omega}'$$

$$=\frac{1}{\sigma}\mu_j\frac{\partial^2}{\partial i\partial j}\sum_{l''=0}^{\infty}\sum_{m''=0}^{l''}\sum_{l=0}^{L}\sum_{m=0}^{l}\frac{\sigma_{l''}}{\sigma-\sigma_{l''}}\left[\phi_{l,m}^{c}(\mathbf{r})Y_{l'',m''}^{c}(\hat{\Omega})\int_{4\pi}\mu_i'Y_{l,m}^{c}(\hat{\Omega}')Y_{l'',m''}^{c}(\hat{\Omega}')d\hat{\Omega}'\right.$$

$$+\phi_{l,m}^{c}(\mathbf{r})Y_{l'',m''}^{s}(\hat{\Omega})\int_{4\pi}\mu_i'Y_{l,m}^{c}(\hat{\Omega}')Y_{l'',m''}^{s}(\hat{\Omega}')d\hat{\Omega}'+\phi_{l,m}^{s}(\mathbf{r})Y_{l'',m''}^{c}(\hat{\Omega})\int_{4\pi}\mu_i'Y_{l,m}^{s}(\hat{\Omega}')Y_{l'',m''}^{c}(\hat{\Omega}')d\hat{\Omega}'$$

$$\left.+\phi_{l,m}^{s}(\mathbf{r})Y_{l'',m''}^{s}(\hat{\Omega})\int_{4\pi}\mu_i'Y_{l,m}^{s}(\hat{\Omega}')Y_{l'',m''}^{s}(\hat{\Omega}')d\hat{\Omega}'\right]$$

Now multipling by $Y_{l',m'}^{\alpha}(\hat{\Omega})$ (with $\alpha\in\{c,s\}$) and integrating:

$$\int_{4\pi} Y^{\alpha}_{l',m'}(\hat{\Omega})\frac{1}{\sigma}\mu_j\frac{\partial^2}{\partial i\partial j}\sum_{l''=0}^{\infty}\sum_{m''=0}^{l''}\sum_{l=0}^{L}\sum_{m=0}^{l}\frac{\sigma_{l''}}{\sigma-\sigma_{l''}}\Bigg[\phi^c_{l,m}(\mathbf{r})Y^c_{l'',m''}(\hat{\Omega})\int_{4\pi}\mu_i' Y^c_{l,m}(\hat{\Omega}')Y^c_{l'',m''}(\hat{\Omega}')d\hat{\Omega}'$$
$$+\phi^c_{l,m}(\mathbf{r})Y^s_{l'',m''}(\hat{\Omega})\int_{4\pi}\mu_i' Y^c_{l,m}(\hat{\Omega}')Y^s_{l'',m''}(\hat{\Omega}')d\hat{\Omega}'+\phi^s_{l,m}(\mathbf{r})Y^c_{l'',m''}(\hat{\Omega})\int_{4\pi}\mu_i' Y^s_{l,m}(\hat{\Omega}')Y^c_{l'',m''}(\hat{\Omega}')d\hat{\Omega}'$$
$$+\phi^s_{l,m}(\mathbf{r})Y^s_{l'',m''}(\hat{\Omega})\int_{4\pi}\mu_i' Y^s_{l,m}(\hat{\Omega}')Y^s_{l'',m''}(\hat{\Omega}')d\hat{\Omega}'\Bigg]d\hat{\Omega}$$
$$=\frac{1}{\sigma}\frac{\partial^2}{\partial i\partial j}\sum_{l''=0}^{\infty}\sum_{m''=0}^{l''}\sum_{l=0}^{L}\sum_{m=0}^{l}\frac{\sigma_{l''}}{\sigma-\sigma_{l''}}\Bigg[\phi^c_{l,m}(\mathbf{r})\int_{4\pi}\mu_j Y^{\alpha}_{l',m'}(\hat{\Omega})Y^c_{l'',m''}(\hat{\Omega})d\hat{\Omega}\int_{4\pi}\mu_i' Y^c_{l,m}(\hat{\Omega}')Y^c_{l'',m''}(\hat{\Omega}')d\hat{\Omega}'$$
$$+\phi^c_{l,m}(\mathbf{r})\int_{4\pi}\mu_j Y^{\alpha}_{l',m'}(\hat{\Omega})Y^s_{l'',m''}(\hat{\Omega})d\hat{\Omega}\int_{4\pi}\mu_i' Y^c_{l,m}(\hat{\Omega}')Y^s_{l'',m''}(\hat{\Omega}')d\hat{\Omega}'$$
$$+\phi^s_{l,m}(\mathbf{r})\int_{4\pi}\mu_j Y^{\alpha}_{l',m'}(\hat{\Omega})Y^c_{l'',m''}(\hat{\Omega})d\hat{\Omega}\int_{4\pi}\mu_i' Y^s_{l,m}(\hat{\Omega}')Y^c_{l'',m''}(\hat{\Omega}')d\hat{\Omega}'$$
$$+\phi^s_{l,m}(\mathbf{r})\int_{4\pi}\mu_j Y^{\alpha}_{l',m'}(\hat{\Omega})Y^s_{l'',m''}(\hat{\Omega})d\hat{\Omega}\int_{4\pi}\mu_i' Y^s_{l,m}(\hat{\Omega}')Y^s_{l'',m''}(\hat{\Omega}')d\hat{\Omega}'\Bigg]$$

The complete second term is given by

$$-\sum_{\substack{i\in\{x,y,z\}\\ j\in\{x,y,z\}}}\Bigg[\frac{1}{\sigma}\frac{\partial^2}{\partial i\partial j}\sum_{l''=0}^{\infty}\sum_{m''=0}^{l''}\sum_{l=0}^{L}\sum_{m=0}^{l}\frac{\sigma_{l''}}{\sigma-\sigma_{l''}}\Bigg[\phi^c_{l,m}(\mathbf{r})\int_{4\pi}\mu_j Y^{\alpha}_{l',m'}(\hat{\Omega})Y^c_{l'',m''}(\hat{\Omega})d\hat{\Omega}$$
$$\times\int_{4\pi}\mu_i' Y^c_{l,m}(\hat{\Omega}')Y^c_{l'',m''}(\hat{\Omega}')d\hat{\Omega}'$$
$$+\phi^c_{l,m}(\mathbf{r})\int_{4\pi}\mu_j Y^{\alpha}_{l',m'}(\hat{\Omega})Y^s_{l'',m''}(\hat{\Omega})d\hat{\Omega}\int_{4\pi}\mu_i' Y^c_{l,m}(\hat{\Omega}')Y^s_{l'',m''}(\hat{\Omega}')d\hat{\Omega}' \qquad \textbf{5.19}$$
$$+\phi^s_{l,m}(\mathbf{r})\int_{4\pi}\mu_j Y^{\alpha}_{l',m'}(\hat{\Omega})Y^c_{l'',m''}(\hat{\Omega})d\hat{\Omega}\int_{4\pi}\mu_i' Y^s_{l,m}(\hat{\Omega}')Y^c_{l'',m''}(\hat{\Omega}')d\hat{\Omega}'$$
$$+\phi^s_{l,m}(\mathbf{r})\int_{4\pi}\mu_j Y^{\alpha}_{l',m'}(\hat{\Omega})Y^s_{l'',m''}(\hat{\Omega})d\hat{\Omega}\int_{4\pi}\mu_i' Y^s_{l,m}(\hat{\Omega}')Y^s_{l'',m''}(\hat{\Omega}')d\hat{\Omega}'\Bigg]\Bigg]$$

Proceeding with the third term:

$$\sigma\psi(\mathbf{r},\hat{\Omega})=\sigma\sum_{l=0}^{L}\sum_{m=0}^{l}\Big[\phi^c_{l,m}(\mathbf{r})Y^c_{l,m}(\hat{\Omega})+\phi^s_{l,m}(\mathbf{r})Y^s_{l,m}(\hat{\Omega})\Big]$$

Multipling by $Y^{\alpha}_{l',m'}(\hat{\Omega})$ (with $\alpha\in\{c,s\}$) and integrating:

$$\int_{4\pi} Y^{\alpha}_{l',m'}(\hat{\Omega})\sigma\sum_{l=0}^{L}\sum_{m=0}^{l}\left[\phi^{c}_{l,m}(\mathbf{r})Y^{c}_{l,m}(\hat{\Omega})+\phi^{s}_{l,m}(\mathbf{r})Y^{s}_{l,m}(\hat{\Omega})\right]d\hat{\Omega}$$
$$=\sigma\sum_{l=0}^{L}\sum_{m=0}^{l}\left[\phi^{c}_{l,m}(\mathbf{r})\int_{4\pi} Y^{\alpha}_{l',m'}(\hat{\Omega})Y^{c}_{l,m}(\hat{\Omega})d\hat{\Omega}+\phi^{s}_{l,m}(\mathbf{r})\int_{4\pi} Y^{\alpha}_{l',m'}(\hat{\Omega})Y^{s}_{l,m}(\hat{\Omega})d\hat{\Omega}\right]$$

The third term is thus

$$\sigma\sum_{l=0}^{L}\sum_{m=0}^{l}\left[\phi^{c}_{l,m}(\mathbf{r})\int_{4\pi} Y^{\alpha}_{l',m'}(\hat{\Omega})Y^{c}_{l,m}(\hat{\Omega})d\hat{\Omega}+\phi^{s}_{l,m}(\mathbf{r})\int_{4\pi} Y^{\alpha}_{l',m'}(\hat{\Omega})Y^{s}_{l,m}(\hat{\Omega})d\hat{\Omega}\right] \qquad \textbf{5.20}$$

Next, the fourth term yields:

$$\sum_{l=0}^{\infty}\frac{2l''+1}{4\pi}\sigma_{l''}\int_{4\pi} P_{l''}(\hat{\Omega}\bullet\hat{\Omega}')\psi(\mathbf{r},\hat{\Omega}')d\hat{\Omega}'$$
$$=\sum_{l=0}^{\infty}\frac{2l''+1}{4\pi}\sigma_{l''}\int_{4\pi} P_{l''}(\hat{\Omega}\bullet\hat{\Omega}')\sum_{l=0}^{L}\sum_{m=0}^{l}\left[\phi^{c}_{l,m}(\mathbf{r})Y^{c}_{l,m}(\hat{\Omega}')+\phi^{s}_{l,m}(\mathbf{r})Y^{s}_{l,m}(\hat{\Omega}')\right]d\hat{\Omega}'$$
$$=\sum_{l=0}^{\infty}\sum_{l=0}^{L}\sum_{m=0}^{l}\frac{2l''+1}{4\pi}\sigma_{l''}\int_{4\pi} P_{l''}(\hat{\Omega}\bullet\hat{\Omega}')\left[\phi^{c}_{l,m}(\mathbf{r})Y^{c}_{l,m}(\hat{\Omega}')+\phi^{s}_{l,m}(\mathbf{r})Y^{s}_{l,m}(\hat{\Omega}')\right]d\hat{\Omega}'$$

Utilizing the addition theorem Eq. **5.10**:

$$\sum_{l''=0}^{\infty}\frac{2l''+1}{4\pi}\sigma_{l''}\int_{4\pi} P_{l''}(\hat{\Omega}\bullet\hat{\Omega}')\psi(\mathbf{r},\hat{\Omega}')d\hat{\Omega}'$$
$$=\sum_{l''=0}^{\infty}\sum_{m''=0}^{l''}\sum_{l=0}^{L}\sum_{m=0}^{l}\sigma_{l''}\int_{4\pi}\left[Y^{c}_{l'',m''}(\hat{\Omega})Y^{c}_{l'',m''}(\hat{\Omega}')+Y^{s}_{l'',m''}(\hat{\Omega})Y^{s}_{l'',m''}(\hat{\Omega}')\right]\left[\phi^{c}_{l,m}(\mathbf{r})Y^{c}_{l,m}(\hat{\Omega}')+\phi^{s}_{l,m}(\mathbf{r})Y^{s}_{l,m}(\hat{\Omega}')\right]d\hat{\Omega}'$$
$$=\sum_{l''=0}^{\infty}\sum_{m''=0}^{l''}\sum_{l=0}^{L}\sum_{m=0}^{l}\sigma_{l''}\int_{4\pi}\left[Y^{c}_{l'',m''}(\hat{\Omega})Y^{c}_{l'',m''}(\hat{\Omega}')\phi^{c}_{l,m}(\mathbf{r})Y^{c}_{l,m}(\hat{\Omega}')+Y^{s}_{l'',m''}(\hat{\Omega})Y^{s}_{l'',m''}(\hat{\Omega}')\phi^{c}_{l,m}(\mathbf{r})Y^{c}_{l,m}(\hat{\Omega}')\right.$$
$$\left.+Y^{c}_{l'',m''}(\hat{\Omega})Y^{c}_{l'',m''}(\hat{\Omega}')\phi^{s}_{l,m}(\mathbf{r})Y^{s}_{l,m}(\hat{\Omega}')+Y^{s}_{l'',m''}(\hat{\Omega})Y^{s}_{l'',m''}(\hat{\Omega}')\phi^{s}_{l,m}(\mathbf{r})Y^{s}_{l,m}(\hat{\Omega}')\right]d\hat{\Omega}'$$
$$=\sum_{l''=0}^{\infty}\sum_{m''=0}^{l''}\sum_{l=0}^{L}\sum_{m=0}^{l}\sigma_{l''}\left[\phi^{c}_{l,m}(\mathbf{r})Y^{c}_{l'',m''}(\hat{\Omega})\int_{4\pi} Y^{c}_{l'',m''}(\hat{\Omega}')Y^{c}_{l,m}(\hat{\Omega}')d\hat{\Omega}'+\phi^{c}_{l,m}(\mathbf{r})Y^{s}_{l'',m''}(\hat{\Omega})\int_{4\pi} Y^{s}_{l'',m''}(\hat{\Omega}')Y^{c}_{l,m}(\hat{\Omega}')d\hat{\Omega}'\right.$$
$$\left.\phi^{s}_{l,m}(\mathbf{r})Y^{c}_{l'',m''}(\hat{\Omega})\int_{4\pi} Y^{c}_{l'',m''}(\hat{\Omega}')Y^{s}_{l,m}(\hat{\Omega}')d\hat{\Omega}'+\phi^{s}_{l,m}(\mathbf{r})Y^{s}_{l'',m''}(\hat{\Omega})\int_{4\pi} Y^{s}_{l'',m''}(\hat{\Omega}')Y^{s}_{l,m}(\hat{\Omega}')d\hat{\Omega}'\right]$$

By making use of the orthonormality relations Eqs. **5.7-9**, this term can be condensed to:

$$\sum_{l=0}^{L}\sum_{m=0}^{l}\sigma_{l}\left[\phi^{c}_{l,m}(\mathbf{r})Y^{c}_{l,m}(\hat{\Omega})+\phi^{s}_{l,m}(\mathbf{r})Y^{s}_{l,m}(\hat{\Omega})\right]$$

Now, multipling by $Y^{\alpha}_{l',m'}(\hat{\Omega})$ (with $\alpha \in \{c,s\}$) and integrating:

$$\int_{4\pi} Y^{\alpha}_{l',m'}(\hat{\Omega}) \sum_{l=0}^{L} \sum_{m=0}^{l} \sigma_l \left[\phi^c_{l,m}(\mathbf{r}) Y^c_{l,m}(\hat{\Omega}) + \phi^s_{l,m}(\mathbf{r}) Y^s_{l,m}(\hat{\Omega}) \right] d\hat{\Omega}$$

This gives us the expanded fourth term:

$$\sum_{l=0}^{L} \sum_{m=0}^{l} \sigma_l \left[\phi^c_{l,m}(\mathbf{r}) \int_{4\pi} Y^{\alpha}_{l',m'}(\hat{\Omega}) Y^c_{l,m}(\hat{\Omega}) d\hat{\Omega} + \phi^s_{l,m}(\mathbf{r}) \int_{4\pi} Y^{\alpha}_{l',m'}(\hat{\Omega}) Y^s_{l,m}(\hat{\Omega}) d\hat{\Omega} \right] \qquad \textbf{5.21}$$

We next move to expanding the source term on the right hand side of Eq. **5.17**. We define the source expansion similarly to the flux expansion:

$$Q(\mathbf{r},\hat{\Omega}) = \sum_{l=0}^{L} \sum_{m=0}^{l} \left[q^c_{l,m}(\mathbf{r}) Y^c_{l,m}(\hat{\Omega}) + q^s_{l,m}(\mathbf{r}) Y^s_{l,m}(\hat{\Omega}) \right] \qquad \textbf{5.22}$$

where

$$\begin{aligned} q^c_{l,m}(\mathbf{r}) &= \int_{4\pi} Q(\mathbf{r},\hat{\Omega}) Y^c_{l,m}(\hat{\Omega}) d\hat{\Omega} \\ q^s_{l,m}(\mathbf{r}) &= \int_{4\pi} Q(\mathbf{r},\hat{\Omega}) Y^s_{l,m}(\hat{\Omega}) d\hat{\Omega} \end{aligned} \qquad \textbf{5.23}$$

The first term on the right hand side of Eq. **5.17** is simply the flux expansion Eq. **5.22**. We multiply this term by $Y^{\alpha}_{l',m'}(\hat{\Omega})$ (with $\alpha \in \{c,s\}$) and integrate over the unit sphere:

$$\int_{4\pi} Y^{\alpha}_{l',m'}(\hat{\Omega}) \sum_{l=0}^{L} \sum_{m=0}^{l} \left[q^c_{l,m}(\mathbf{r}) Y^c_{l,m}(\hat{\Omega}) + q^s_{l,m}(\mathbf{r}) Y^s_{l,m}(\hat{\Omega}) \right] d\hat{\Omega}$$

which gives us the first term on the right hand side of Eq. **5.17**:

$$\sum_{l=0}^{L} \sum_{m=0}^{l} \left[q^c_{l,m}(\mathbf{r}) \int_{4\pi} Y^{\alpha}_{l',m'}(\hat{\Omega}) Y^c_{l,m}(\hat{\Omega}) d\hat{\Omega} + q^s_{l,m}(\mathbf{r}) \int_{4\pi} Y^{\alpha}_{l',m'}(\hat{\Omega}) Y^s_{l,m}(\hat{\Omega}) d\hat{\Omega} \right] \qquad \textbf{5.24}$$

The second term, with $i \in \{x, y, z\}$, is:

$$\frac{1}{\sigma}\frac{\partial}{\partial i}\mu_i Q(\mathbf{r},\hat{\Omega}) = \frac{1}{\sigma}\frac{\partial}{\partial i}\mu_i \sum_{l=0}^{L}\sum_{m=0}^{l}\left[q_{l,m}^{c}(\mathbf{r}) Y_{l,m}^{c}(\hat{\Omega}) + q_{l,m}^{s}(\mathbf{r}) Y_{l,m}^{s}(\hat{\Omega}) \right]$$
$$= \frac{1}{\sigma}\frac{\partial}{\partial i}\sum_{l=0}^{L}\sum_{m=0}^{l}\left[q_{l,m}^{c}(\mathbf{r})\mu_i Y_{l,m}^{c}(\hat{\Omega}) + q_{l,m}^{s}(\mathbf{r})\mu_i Y_{l,m}^{s}(\hat{\Omega}) \right]$$

Multipling this term by $Y_{l',m'}^{\alpha}(\hat{\Omega})$ (with $\alpha \in \{c,s\}$) and integrating:

$$\int_{4\pi} Y_{l',m'}^{\alpha}(\hat{\Omega})\frac{1}{\sigma}\frac{\partial}{\partial i}\sum_{l=0}^{L}\sum_{m=0}^{l}\left[q_{l,m}^{c}(\mathbf{r})\mu_i Y_{l,m}^{c}(\hat{\Omega}) + q_{l,m}^{s}(\mathbf{r})\mu_i Y_{l,m}^{s}(\hat{\Omega}) \right] d\hat{\Omega}$$
$$= \frac{1}{\sigma}\frac{\partial}{\partial i}\sum_{l=0}^{L}\sum_{m=0}^{l}\left[q_{l,m}^{c}(\mathbf{r})\int_{4\pi}\mu_i Y_{l',m'}^{\alpha}(\hat{\Omega}) Y_{l,m}^{c}(\hat{\Omega}) d\hat{\Omega} + q_{l,m}^{s}(\mathbf{r})\int_{4\pi}\mu_i Y_{l',m'}^{\alpha}(\hat{\Omega}) Y_{l,m}^{s}(\hat{\Omega}) d\hat{\Omega} \right]$$

The second term on the right hand side is thus written as:

$$-\sum_{i\in\{x,y,z\}}\frac{1}{\sigma}\frac{\partial}{\partial i}\sum_{l=0}^{L}\sum_{m=0}^{l}\left[q_{l,m}^{c}(\mathbf{r})\int_{4\pi}\mu_i Y_{l',m'}^{\alpha}(\hat{\Omega}) Y_{l,m}^{c}(\hat{\Omega}) d\hat{\Omega} + q_{l,m}^{s}(\mathbf{r})\int_{4\pi}\mu_i Y_{l',m'}^{\alpha}(\hat{\Omega}) Y_{l,m}^{s}(\hat{\Omega}) d\hat{\Omega} \right] \quad \textbf{5.25}$$

Finally, we expand the third term on the right hand side:

$$\frac{1}{\sigma}\frac{\partial}{\partial i}\mu_i \sum_{l''=0}^{\infty}\frac{2l''+1}{4\pi}\frac{\sigma_{l''}}{\sigma-\sigma_{l''}}\int_{4\pi} P_{l''}(\hat{\Omega}\bullet\hat{\Omega}') Q(\mathbf{r},\hat{\Omega}') d\hat{\Omega}'$$
$$= \frac{1}{\sigma}\frac{\partial}{\partial i}\mu_i \sum_{l''=0}^{\infty}\frac{2l''+1}{4\pi}\frac{\sigma_{l''}}{\sigma-\sigma_{l''}}\int_{4\pi} P_{l''}(\hat{\Omega}\bullet\hat{\Omega}')\sum_{l=0}^{L}\sum_{m=0}^{l}\left[q_{l,m}^{c}(\mathbf{r}) Y_{l,m}^{c}(\hat{\Omega}) + q_{l,m}^{s}(\mathbf{r}) Y_{l,m}^{s}(\hat{\Omega}) \right] d\hat{\Omega}'$$
$$= \frac{1}{\sigma}\frac{\partial}{\partial i}\mu_i \sum_{l''=0}^{\infty}\sum_{l=0}^{L}\sum_{m=0}^{l}\frac{2l''+1}{4\pi}\frac{\sigma_{l''}}{\sigma-\sigma_{l''}}\int_{4\pi} P_{l''}(\hat{\Omega}\bullet\hat{\Omega}')\left[q_{l,m}^{c}(\mathbf{r}) Y_{l,m}^{c}(\hat{\Omega}) + q_{l,m}^{s}(\mathbf{r}) Y_{l,m}^{s}(\hat{\Omega}) \right] d\hat{\Omega}'$$

Using the addition theorem:

$$\frac{1}{\sigma}\frac{\partial}{\partial i}\mu_i\sum_{l''=0}^{\infty}\sum_{l=0}^{L}\sum_{m=0}^{l}\frac{\sigma_{l''}}{\sigma-\sigma_{l''}}\int_{4\pi}\left[Y^c_{l'',m''}(\hat{\Omega})Y^c_{l'',m''}(\hat{\Omega}')+Y^s_{l'',m''}(\hat{\Omega})Y^s_{l'',m''}(\hat{\Omega}')\right]$$
$$\times\left[q^c_{l,m}(\mathbf{r})Y^c_{l,m}(\hat{\Omega}')+q^s_{l,m}(\mathbf{r})Y^s_{l,m}(\hat{\Omega}')\right]d\hat{\Omega}'$$
$$=\frac{1}{\sigma}\frac{\partial}{\partial i}\mu_i\sum_{l''=0}^{\infty}\sum_{m''=0}^{l''}\sum_{l=0}^{L}\sum_{m=0}^{l}\frac{\sigma_{l''}}{\sigma-\sigma_{l''}}\int_{4\pi}\left[Y^c_{l'',m''}(\hat{\Omega})Y^c_{l'',m''}(\hat{\Omega}')q^c_{l,m}(\mathbf{r})Y^c_{l,m}(\hat{\Omega}')\right.$$
$$+Y^s_{l'',m''}(\hat{\Omega})Y^s_{l'',m''}(\hat{\Omega}')q^c_{l,m}(\mathbf{r})Y^c_{l,m}(\hat{\Omega}')+Y^c_{l'',m''}(\hat{\Omega})Y^c_{l'',m''}(\hat{\Omega}')q^s_{l,m}(\mathbf{r})Y^s_{l,m}(\hat{\Omega}')$$
$$\left.+Y^s_{l'',m''}(\hat{\Omega})Y^s_{l'',m''}(\hat{\Omega}')q^s_{l,m}(\mathbf{r})Y^s_{l,m}(\hat{\Omega}')\right]d\hat{\Omega}'$$
$$=\frac{1}{\sigma}\frac{\partial}{\partial i}\mu_i\sum_{l''=0}^{\infty}\sum_{m''=0}^{l''}\sum_{l=0}^{L}\sum_{m=0}^{l}\frac{\sigma_{l''}}{\sigma-\sigma_{l''}}\left[q^c_{l,m}(\mathbf{r})Y^c_{l'',m''}(\hat{\Omega})\int_{4\pi}Y^c_{l'',m''}(\hat{\Omega}')Y^c_{l,m}(\hat{\Omega}')d\hat{\Omega}'\right.$$
$$+q^c_{l,m}(\mathbf{r})Y^s_{l'',m''}(\hat{\Omega})\int_{4\pi}Y^s_{l'',m''}(\hat{\Omega}')Y^c_{l,m}(\hat{\Omega}')d\hat{\Omega}'+q^s_{l,m}(\mathbf{r})Y^c_{l'',m''}(\hat{\Omega})\int_{4\pi}Y^c_{l'',m''}(\hat{\Omega}')Y^s_{l,m}(\hat{\Omega}')d\hat{\Omega}'$$
$$\left.+q^s_{l,m}(\mathbf{r})Y^s_{l'',m''}(\hat{\Omega})\int_{4\pi}Y^s_{l'',m''}(\hat{\Omega}')Y^s_{l,m}(\hat{\Omega}')d\hat{\Omega}'\right]$$

When the orthonormality relations are applied, this reduces to:

$$\frac{1}{\sigma}\frac{\partial}{\partial i}\mu_i\sum_{l=0}^{L}\sum_{m=0}^{l}\frac{\sigma_l}{\sigma-\sigma_l}\left[q^c_{l,m}(\mathbf{r})Y^c_{l,m}(\hat{\Omega})+q^s_{l,m}(\mathbf{r})Y^s_{l,m}(\hat{\Omega})\right]$$

We again multiply by $Y^{\alpha}_{l',m'}(\hat{\Omega})$ (with $\alpha\in\{c,s\}$) and integrate over the angular variables:

$$\int_{4\pi}Y^{\alpha}_{l',m'}(\hat{\Omega})\frac{1}{\sigma}\frac{\partial}{\partial i}\mu_i\sum_{l=0}^{L}\sum_{m=0}^{l}\frac{\sigma_l}{\sigma-\sigma_l}\left[q^c_{l,m}(\mathbf{r})Y^c_{l,m}(\hat{\Omega})+q^s_{l,m}(\mathbf{r})Y^s_{l,m}(\hat{\Omega})\right]d\hat{\Omega}$$
$$=\frac{1}{\sigma}\frac{\partial}{\partial i}\sum_{l=0}^{L}\sum_{m=0}^{l}\frac{\sigma_l}{\sigma-\sigma_l}\left[q^c_{l,m}(\mathbf{r})\int_{4\pi}\mu_iY^{\alpha}_{l',m'}(\hat{\Omega})Y^c_{l,m}(\hat{\Omega})d\hat{\Omega}+q^s_{l,m}(\mathbf{r})\int_{4\pi}\mu_iY^{\alpha}_{l',m'}(\hat{\Omega})Y^s_{l,m}(\hat{\Omega})d\hat{\Omega}\right]$$

The complete third term on the right hand side is obtained by summing over $i\in\{x,y,z\}$:

$$-\sum_{i\in\{x,y,z\}}\frac{1}{\sigma}\frac{\partial}{\partial i}\sum_{l=0}^{L}\sum_{m=0}^{l}\frac{\sigma_l}{\sigma-\sigma_l}\left[q_{l,m}^{c}(\mathbf{r})\int_{4\pi}\mu_i Y_{l',m'}^{\alpha}(\hat{\Omega})Y_{l,m}^{c}(\hat{\Omega})d\hat{\Omega}\right.$$
$$\left.+q_{l,m}^{s}(\mathbf{r})\int_{4\pi}\mu_i Y_{l',m'}^{\alpha}(\hat{\Omega})Y_{l,m}^{s}(\hat{\Omega})d\hat{\Omega}\right] \qquad \textbf{5.26}$$

The terms Eqs. **5.18-21** and Eqs. **5.24-26** are one step away from giving the spherical harmonics discretization of the SAAF equation. We wish to make use of the orthonormality relations Eqs. **5.7-9** in order to arrive at the simplest possible form of these equations. By using these relations, we can explicitly perform the integrals appearing in Eqs. **5.18-21** and Eqs. **5.24-26**, and by extension collapse the summations over *l* and *m* present in these expressions to a summation over only a relatively few moments.

Fortunately, closed-form recursion relationships for $\mu_i Y_{l,m}^{\alpha}(\hat{\Omega})$, $i\in\{x,y,z\}$ and $\alpha\in\{c,s\}$, do exist and allow us to write the spherical harmonics expansion of Eq. **5.17** as a coupled set of equations for purely spatially-dependent moments. We will next derive these recursion relationships for both $\mu_i Y_{l,m}^{\alpha}(\hat{\Omega})$ (necessary to simplify Eq. **5.19**, Eq. **5.25** and Eq. **5.26**), as well as $\mu_i\mu_j Y_{l,m}^{\alpha}(\hat{\Omega})$ (necessary to simplify Eq. **5.18**).

5.3 Recursion Relations for $\mu_i Y_{l,m}^{\alpha}(\hat{\Omega})$

We begin our derivation of the spherical harmonics relationships by recalling four basic trigonometric identities:

$$\cos m\varphi\cos\varphi=\frac{1}{2}\left[\cos(m+1)\varphi+\cos(m-1)\varphi\right] \quad \textbf{5.27}$$

$$\cos m\varphi\sin\varphi=\frac{1}{2}\left[\sin(m+1)\varphi-\sin(m-1)\varphi\right] \quad \textbf{5.28}$$

$$\sin m\varphi\cos\varphi=\frac{1}{2}\left[\sin(m+1)\varphi+\sin(m-1)\varphi\right] \quad \textbf{5.29}$$

$$\sin m\varphi\sin\varphi=\frac{1}{2}\left[\cos(m-1)\varphi-\cos(m+1)\varphi\right] \quad \textbf{5.30}$$

Also, it will be useful to recall the angular cosine definitions given above:

$$\mu_x=\cos\varphi\sin\theta$$
$$\mu_y=\sin\varphi\sin\theta$$
$$\mu_z=\cos\theta$$

We can express μ_x and μ_y in terms of μ_z:

$$\mu_x=\cos\varphi\sqrt{1-\mu_z^2}$$
$$\mu_y=\sin\varphi\sqrt{1-\mu_z^2}$$

Finally, it will be necessary to make use of the recursion relationships that exist for the associated Legendre polynomials that make up part of the spherical harmonics [13]:

$$\sqrt{1-\mu_z^2}P_l^m(\mu_z)=\frac{1}{2l+1}\Big[(l-m+2)(l-m+1)P_{l+1}^{m-1}(\mu_z) -(l+m)(l+m-1)P_{l-1}^{m-1}(\mu_z)\Big] \quad \textbf{5.31}$$

$$\sqrt{1-\mu_z^2}P_l^m(\mu_z)=\frac{1}{2l+1}\Big[P_{l-1}^{m+1}(\mu_z)-P_{l+1}^{m+1}(\mu_z)\Big] \quad \textbf{5.32}$$

$$\mu_z P_l^m(\mu_z)=\frac{1}{2l+1}\Big[(l+m)P_{l-1}^m(\mu_z)+(l-m+1)P_{l+1}^m(\mu_z)\Big] \quad \textbf{5.33}$$

With these definitions and relationships in mind, the recursion formulas for $\mu_i Y_{l,m}^{\alpha}(\hat{\Omega})$, $i\in\{x,y,z\}$ and $\alpha\in\{c,s\}$, are next presented.

We start with $\mu_x Y^c_{l,m}(\hat{\Omega})$:

$$\begin{aligned}\mu_x Y^c_{l,m}(\hat{\Omega}) &= \sqrt{1-\mu_z^2}\cos\varphi C_{l,m} P_l^m(\mu_z)\cos m\varphi \\ &= \sqrt{1-\mu_z^2}\cos\varphi\sqrt{\frac{2l+1}{4\pi}}\sqrt{\varepsilon_m}\sqrt{\frac{(l-m)!}{(l+m)!}}P_l^m(\mu_z)\cos m\varphi \\ &= \sqrt{\frac{2l+1}{4\pi}}\sqrt{\varepsilon_m}\sqrt{\frac{(l-m)!}{(l+m)!}}\sqrt{1-\mu_z^2}P_l^m(\mu_z)\cos\varphi\cos m\varphi\end{aligned}$$

Using Eq. **5.27**:

$$\begin{aligned}\mu_x Y^c_{l,m}(\hat{\Omega}) &= \sqrt{\frac{2l+1}{4\pi}}\sqrt{\varepsilon_m}\sqrt{\frac{(l-m)!}{(l+m)!}}\sqrt{1-\mu_z^2}P_l^m(\mu_z)\frac{1}{2}\left[\cos(m+1)\varphi+\cos(m-1)\varphi\right] \\ &= \sqrt{\frac{2l+1}{4\pi}}\sqrt{\varepsilon_m}\sqrt{\frac{(l-m)!}{(l+m)!}}\frac{1}{2}\left[\sqrt{1-\mu_z^2}P_l^m(\mu_z)\cos(m+1)\varphi+\sqrt{1-\mu_z^2}P_l^m(\mu_z)\cos(m-1)\varphi\right]\end{aligned}$$

Next, using recursion relations Eq. **5.31** and Eq. **5.32**:

$$\mu_x Y^c_{l,m}(\hat{\Omega}) = \sqrt{\frac{2l+1}{4\pi}}\sqrt{\varepsilon_m}\sqrt{\frac{(l-m)!}{(l+m)!}}\frac{1}{2}\begin{bmatrix} \frac{1}{2l+1}\left[P^{m+1}_{l-1}(\mu_z) - P^{m+1}_{l+1}(\mu_z)\right]\cos(m+1)\varphi \\ +\frac{1}{2l+1}\left[(l-m+2)(l-m+1)P^{m-1}_{l+1}(\mu_z)\right. \\ \left.-(l+m)(l+m-1)P^{m-1}_{l-1}(\mu_z)\right]\cos(m-1)\varphi \end{bmatrix}$$

$$= \frac{1}{2}\begin{bmatrix} \frac{1}{2l+1}\sqrt{\frac{2l+1}{4\pi}}\sqrt{\varepsilon_m}\sqrt{\frac{(l-m)!}{(l+m)!}}P^{m+1}_{l-1}(\mu_z)\cos(m+1)\varphi \\ -\frac{1}{2l+1}\sqrt{\frac{2l+1}{4\pi}}\sqrt{\varepsilon_m}\sqrt{\frac{(l-m)!}{(l+m)!}}P^{m+1}_{l+1}(\mu_z)\cos(m+1)\varphi \\ +\frac{1}{2l+1}\sqrt{\frac{2l+1}{4\pi}}\sqrt{\varepsilon_m}\sqrt{\frac{(l-m)!}{(l+m)!}}(l-m+2)(l-m+1)P^{m-1}_{l+1}(\mu_z)\cos(m-1)\varphi \\ -\frac{1}{2l+1}\sqrt{\frac{2l+1}{4\pi}}\sqrt{\varepsilon_m}\sqrt{\frac{(l-m)!}{(l+m)!}}(l+m)(l+m-1)P^{m-1}_{l-1}(\mu_z)\cos(m-1)\varphi \end{bmatrix}$$

$$= \frac{\sqrt{\varepsilon_m}}{2}\begin{bmatrix} \sqrt{\frac{1}{2l+1}}\sqrt{\frac{1}{4\pi}}\sqrt{\frac{(l-m)!}{(l+m)!}}P^{m+1}_{l-1}(\mu_z)\cos(m+1)\varphi \\ -\sqrt{\frac{1}{2l+1}}\sqrt{\frac{1}{4\pi}}\sqrt{\frac{(l-m)!}{(l+m)!}}P^{m+1}_{l+1}(\mu_z)\cos(m+1)\varphi \\ +\sqrt{\frac{1}{2l+1}}\sqrt{\frac{1}{4\pi}}\sqrt{\frac{(l-m)!}{(l+m)!}}(l-m+2)(l-m+1)P^{m-1}_{l+1}(\mu_z)\cos(m-1)\varphi \\ -\sqrt{\frac{1}{2l+1}}\sqrt{\frac{1}{4\pi}}\sqrt{\frac{(l-m)!}{(l+m)!}}(l+m)(l+m-1)P^{m-1}_{l-1}(\mu_z)\cos(m-1)\varphi \end{bmatrix}$$

Then, multiplying each term by $C_{l',m'} / C_{l',m'}$, where l', m' are the indices of the associated Legendre polynomial present in the term:

$$\mu_x Y_{l,m}^{c}(\hat{\Omega}) = \frac{\sqrt{\varepsilon_m}}{2}\left[\begin{array}{l}
\sqrt{\frac{1}{2l+1}}\frac{C_{l-1,m+1}}{C_{l-1,m+1}}\sqrt{\frac{1}{4\pi}}\sqrt{\frac{(l-m)!}{(l+m)!}}P_{l-1}^{m+1}(\mu_z)\cos(m+1)\varphi \\
-\sqrt{\frac{1}{2l+1}}\frac{C_{l+1,m+1}}{C_{l+1,m+1}}\sqrt{\frac{1}{4\pi}}\sqrt{\frac{(l-m)!}{(l+m)!}}P_{l+1}^{m+1}(\mu_z)\cos(m+1)\varphi \\
+\sqrt{\frac{1}{2l+1}}\frac{C_{l+1,m-1}}{C_{l+1,m-1}}\sqrt{\frac{1}{4\pi}}\sqrt{\frac{(l-m)!}{(l+m)!}}(l-m+2)(l-m+1)P_{l+1}^{m-1}(\mu_z)\cos(m-1)\varphi \\
-\sqrt{\frac{1}{2l+1}}\frac{C_{l-1,m-1}}{C_{l-1,m-1}}\sqrt{\frac{1}{4\pi}}\sqrt{\frac{(l-m)!}{(l+m)!}}(l+m)(l+m-1)P_{l-1}^{m-1}(\mu_z)\cos(m-1)\varphi
\end{array}\right]$$

Expanding $C_{l',m'}$ present in the denominator:

$$\mu_x Y_{l,m}^{c}(\hat{\Omega}) = \frac{\sqrt{\varepsilon_m}}{2}\left[\begin{array}{l}
\sqrt{\frac{1}{2l+1}}\frac{C_{l-1,m+1}}{\sqrt{\frac{2l-1}{4\pi}}\sqrt{\varepsilon_{m+1}}\sqrt{\frac{(l-m-2)!}{(l+m)!}}}\sqrt{\frac{1}{4\pi}}\sqrt{\frac{(l-m)!}{(l+m)!}}P_{l-1}^{m+1}(\mu_z)\cos(m+1)\varphi \\
-\sqrt{\frac{1}{2l+1}}\frac{C_{l+1,m+1}}{\sqrt{\frac{2l+3}{4\pi}}\sqrt{\varepsilon_{m+1}}\sqrt{\frac{(l-m)!}{(l+m+2)!}}}\sqrt{\frac{1}{4\pi}}\sqrt{\frac{(l-m)!}{(l+m)!}}P_{l+1}^{m+1}(\mu_z)\cos(m+1)\varphi \\
+\sqrt{\frac{1}{2l+1}}\frac{C_{l+1,m-1}}{\sqrt{\frac{2l+3}{4\pi}}\sqrt{\varepsilon_{m-1}}\sqrt{\frac{(l-m+2)!}{(l+m)!}}} \\
\times\sqrt{\frac{1}{4\pi}}\sqrt{\frac{(l-m)!}{(l+m)!}}(l-m+2)(l-m+1)P_{l+1}^{m-1}(\mu_z)\cos(m-1)\varphi \\
-\sqrt{\frac{1}{2l+1}}\frac{C_{l-1,m-1}}{\sqrt{\frac{2l-1}{4\pi}}\sqrt{\varepsilon_{m-1}}\sqrt{\frac{(l-m)!}{(l+m-2)!}}} \\
\times\sqrt{\frac{1}{4\pi}}\sqrt{\frac{(l-m)!}{(l+m)!}}(l+m)(l+m-1)P_{l-1}^{m-1}(\mu_z)\cos(m-1)\varphi
\end{array}\right]$$

$$= \frac{\sqrt{\varepsilon_m}}{2} \left[\begin{array}{l} + \dfrac{\sqrt{\dfrac{1}{2l+1}}\sqrt{\dfrac{1}{4\pi}}\sqrt{\dfrac{(l-m)!}{(l+m)!}}}{\sqrt{\dfrac{2l-1}{4\pi}}\sqrt{\varepsilon_{m+1}}\sqrt{\dfrac{(l-m-2)!}{(l+m)!}}} C_{l-1,m+1} P_{l-1}^{m+1}(\mu_z)\cos(m+1)\varphi \\ - \dfrac{\sqrt{\dfrac{1}{2l+1}}\sqrt{\dfrac{1}{4\pi}}\sqrt{\dfrac{(l-m)!}{(l+m)!}}}{\sqrt{\dfrac{2l+3}{4\pi}}\sqrt{\varepsilon_{m+1}}\sqrt{\dfrac{(l-m)!}{(l+m+2)!}}} C_{l+1,m+1} P_{l+1}^{m+1}(\mu_z)\cos(m+1)\varphi \\ + \dfrac{\sqrt{\dfrac{1}{2l+1}}\sqrt{\dfrac{1}{4\pi}}\sqrt{\dfrac{(l-m)!}{(l+m)!}}(l-m+2)(l-m+1)}{\sqrt{\dfrac{2l+3}{4\pi}}\sqrt{\varepsilon_{m-1}}\sqrt{\dfrac{(l-m+2)!}{(l+m)!}}} C_{l+1,m-1} P_{l+1}^{m-1}(\mu_z)\cos(m-1)\varphi \\ - \dfrac{\sqrt{\dfrac{1}{2l+1}}\sqrt{\dfrac{1}{4\pi}}\sqrt{\dfrac{(l-m)!}{(l+m)!}}(l+m)(l+m-1)}{\sqrt{\dfrac{2l-1}{4\pi}}\sqrt{\varepsilon_{m-1}}\sqrt{\dfrac{(l-m)!}{(l+m-2)!}}} C_{l-1,m-1} P_{l-1}^{m-1}(\mu_z)\cos(m-1)\varphi \end{array} \right]$$

Simplifying each term:

$$\mu_x Y_{l,m}^c(\hat{\Omega}) = \frac{\sqrt{\varepsilon_m}}{2} \left[\begin{array}{l} + \dfrac{1}{\sqrt{\varepsilon_{m+1}}} \sqrt{\dfrac{(l-m)(l-m-1)}{(2l-1)(2l+1)}} Y_{l-1,m+1}^c(\hat{\Omega}) \\ - \dfrac{1}{\sqrt{\varepsilon_{m+1}}} \sqrt{\dfrac{(l+m+1)(l+m+2)}{(2l+1)(2l+3)}} Y_{l+1,m+1}^c(\hat{\Omega}) \\ + \dfrac{1}{\sqrt{\varepsilon_{m-1}}} \sqrt{\dfrac{(l-m+1)(l-m+2)}{(2l+1)(2l+3)}} Y_{l+1,m-1}^c(\hat{\Omega}) \\ - \dfrac{1}{\sqrt{\varepsilon_{m-1}}} \sqrt{\dfrac{(l+m)(l+m-1)}{(2l-1)(2l+1)}} Y_{l-1,m-1}^c(\hat{\Omega}) \end{array} \right]$$

The recursion relation for $\mu_x Y_{l,m}^c(\hat{\Omega})$ is thus given by:

$$\mu_x Y_{l,m}^c(\hat{\Omega}) = \frac{1}{2}\left[\begin{array}{l} -\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m-1}}}\sqrt{\frac{(l+m)(l+m-1)}{(2l-1)(2l+1)}}Y_{l-1,m-1}^c(\hat{\Omega}) \\ +\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m+1}}}\sqrt{\frac{(l-m)(l-m-1)}{(2l-1)(2l+1)}}Y_{l-1,m+1}^c(\hat{\Omega}) \\ +\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m-1}}}\sqrt{\frac{(l-m+1)(l-m+2)}{(2l+1)(2l+3)}}Y_{l+1,m-1}^c(\hat{\Omega}) \\ -\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m+1}}}\sqrt{\frac{(l+m+1)(l+m+2)}{(2l+1)(2l+3)}}Y_{l+1,m+1}^c(\hat{\Omega}) \end{array}\right] \quad \textbf{5.34}$$

We follow a similar procedure to derive the recursion relation for $\mu_y Y_{l,m}^c(\hat{\Omega})$. The derivation is basically similar; the only difference lies in using trigonometric identity Eq. **5.28** instead of Eq. **5.27**. For this reason, we simply present the result:

$$\mu_y Y_{l,m}^c(\hat{\Omega}) = \frac{1}{2}\left[\begin{array}{l} +\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m-1}}}\sqrt{\frac{(l+m)(l+m-1)}{(2l-1)(2l+1)}}Y_{l-1,m-1}^s(\hat{\Omega}) \\ +\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m+1}}}\sqrt{\frac{(l-m)(l-m-1)}{(2l-1)(2l+1)}}Y_{l-1,m+1}^s(\hat{\Omega}) \\ -\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m-1}}}\sqrt{\frac{(l-m+2)(l-m+1)}{(2l+1)(2l+3)}}Y_{l+1,m-1}^s(\hat{\Omega}) \\ -\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m+1}}}\sqrt{\frac{(l+m+1)(l+m+2)}{(2l+1)(2l+3)}}Y_{l+1,m+1}^s(\hat{\Omega}) \end{array}\right] \quad \textbf{5.35}$$

Note that in contrast to Eq. **5.34**, the recursion relation for $\mu_y Y_{l,m}^c(\hat{\Omega})$ is cast in terms of the sine components of the spherical harmonics rather than the cosine terms. This will turn out to be significant in the spherical harmonics equations because it will lead to the coupling between the cosine and sine flux moments of certain indices. One might think at first glance that the integrals performed to obtain the P_N discretization might result in a decoupling of these moments, but this is not the case.

Next we present the derivation of the recursion formula for $\mu_z Y^c_{l,m}(\hat{\Omega})$:

$$\begin{aligned}\mu_z Y^c_{l,m}(\hat{\Omega}) &= \mu_z C_{l,m} P^m_l(\mu_z)\cos m\phi \\ &= \mu_z \sqrt{\frac{2l+1}{4\pi}}\sqrt{\varepsilon_m}\sqrt{\frac{(l-m)!}{(l+m)!}}P^m_l(\mu_z)\cos m\phi \\ &= \sqrt{\frac{2l+1}{4\pi}}\sqrt{\varepsilon_m}\sqrt{\frac{(l-m)!}{(l+m)!}}\mu_z P^m_l(\mu_z)\cos m\phi\end{aligned}$$

Using the Legendre recursion relation Eq. **5.33**:

$$\mu_z Y^c_{l,m}(\hat{\Omega}) = \sqrt{\frac{2l+1}{4\pi}}\sqrt{\varepsilon_m}\sqrt{\frac{(l-m)!}{(l+m)!}}\frac{1}{2l+1}\Big[(l-m+1)P^m_{l+1}(\mu_z)+(l+m)P^m_{l-1}(\mu_z)\Big]\cos m\phi$$

$$= \begin{bmatrix} \dfrac{1}{2l+1}\sqrt{\dfrac{2l+1}{4\pi}}\sqrt{\varepsilon_m}\sqrt{\dfrac{(l-m)!}{(l+m)!}}(l-m+1)P^m_{l+1}(\mu_z)\cos m\phi \\ +\dfrac{1}{2l+1}\sqrt{\dfrac{2l+1}{4\pi}}\sqrt{\varepsilon_m}\sqrt{\dfrac{(l-m)!}{(l+m)!}}(l+m)P^m_{l-1}(\mu_z)\cos m\phi \end{bmatrix}$$

Then, multiplying both terms by $C_{l',m'}/C_{l',m'}$, where l',m' are the indices of the associated Legendre polynomial present in the term:

$$\mu_z Y^c_{l,m}(\hat{\Omega}) = \begin{bmatrix} +\dfrac{C_{l+1,m}}{C_{l+1,m}}\dfrac{1}{2l+1}\sqrt{\dfrac{2l+1}{4\pi}}\sqrt{\varepsilon_m}\sqrt{\dfrac{(l-m)!}{(l+m)!}}(l-m+1)P^m_{l+1}(\mu_z)\cos m\phi \\ +\dfrac{C_{l-1,m}}{C_{l-1,m}}\dfrac{1}{2l+1}\sqrt{\dfrac{2l+1}{4\pi}}\sqrt{\varepsilon_m}\sqrt{\dfrac{(l-m)!}{(l+m)!}}(l+m)P^m_{l-1}(\mu_z)\cos m\phi \end{bmatrix}$$

Expanding $C_{l',m'}$ present in denominator and simplifying:

$$\mu_z Y_{l,m}^c(\hat{\Omega}) = \begin{bmatrix} + \dfrac{\dfrac{1}{2l+1}\sqrt{\dfrac{2l+1}{4\pi}}\sqrt{\varepsilon_m}\sqrt{\dfrac{(l-m)!}{(l+m)!}}(l-m+1)}{\sqrt{\dfrac{2l+3}{4\pi}}\sqrt{\varepsilon_m}\sqrt{\dfrac{(l-m+1)!}{(l+m+1)!}}} C_{l+1,m} P_{l+1}^m(\mu_z)\cos m\phi \\ + \dfrac{\dfrac{1}{2l+1}\sqrt{\dfrac{2l+1}{4\pi}}\sqrt{\varepsilon_m}\sqrt{\dfrac{(l-m)!}{(l+m)!}}(l+m)}{\sqrt{\dfrac{2l-1}{4\pi}}\sqrt{\varepsilon_m}\sqrt{\dfrac{(l-m-1)!}{(l+m-1)!}}} C_{l-1,m} P_{l-1}^m(\mu_z)\cos m\phi \end{bmatrix}$$

$$= \begin{bmatrix} + \sqrt{\dfrac{(l-m)(l+m)}{(2l-1)(2l+1)}} C_{l-1,m} P_{l-1}^m(\mu_z)\cos m\phi \\ + \sqrt{\dfrac{(l-m+1)(l+m+1)}{(2l+1)(2l+3)}} C_{l+1,m} P_{l+1}^m(\mu_z)\cos m\phi \end{bmatrix}$$

This gives us the recursion relationship we sought:

$$\mu_z Y_{l,m}^c(\hat{\Omega}) = \left[+ \sqrt{\frac{(l-m)(l+m)}{(2l-1)(2l+1)}} Y_{l-1,m}^c(\hat{\Omega}) + \sqrt{\frac{(l-m+1)(l+m+1)}{(2l+1)(2l+3)}} Y_{l+1,m}^c(\hat{\Omega}) \right] \quad \textbf{5.36}$$

In Eqs. **5.34-36**, we have the recursion relations for $\mu_i Y_{l,m}^c(\hat{\Omega})$, $i \in \{x, y, z\}$.

The procedure to derive the corresponding recursion relations for $\mu_i Y_{l,m}^s(\hat{\Omega})$, $i \in \{x, y, z\}$ is analogous to that for the cosine relations. The only difference lies in using Eq. **5.29** instead of Eq. **5.27** to derive $\mu_x Y_{l,m}^s(\hat{\Omega})$ and Eq. **5.30** instead of Eq. **5.28** to derive $\mu_y Y_{l,m}^s(\hat{\Omega})$. Therefore, it is sufficient to simply present the results from this derivation:

$$\mu_x Y_{l,m}^s(\hat{\Omega}) = \frac{1}{2}\left[\begin{array}{l} -\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m-1}}}\sqrt{\frac{(l+m)(l+m-1)}{(2l-1)(2l+1)}}Y_{l-1,m-1}^s(\hat{\Omega}) \\ +\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m+1}}}\sqrt{\frac{(l-m)(l-m-1)}{(2l-1)(2l+1)}}Y_{l-1,m+1}^s(\hat{\Omega}) \\ +\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m-1}}}\sqrt{\frac{(l-m+1)(l-m+2)}{(2l+1)(2l+3)}}Y_{l+1,m-1}^s(\hat{\Omega}) \\ -\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m+1}}}\sqrt{\frac{(l+m+1)(l+m+2)}{(2l+1)(2l+3)}}Y_{l+1,m+1}^s(\hat{\Omega}) \end{array}\right] \quad \textbf{5.37}$$

$$\mu_y Y_{l,m}^s(\hat{\Omega}) = \frac{1}{2}\left[\begin{array}{l} -\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m-1}}}\sqrt{\frac{(l+m)(l+m-1)}{(2l-1)(2l+1)}}Y_{l-1,m-1}^c(\hat{\Omega}) \\ -\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m+1}}}\sqrt{\frac{(l-m)(l-m-1)}{(2l-1)(2l+1)}}Y_{l-1,m+1}^c(\hat{\Omega}) \\ +\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m-1}}}\sqrt{\frac{(l-m+2)(l-m+1)}{(2l+1)(2l+3)}}Y_{l+1,m-1}^c(\hat{\Omega}) \\ +\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m+1}}}\sqrt{\frac{(l+m+1)(l+m+2)}{(2l+1)(2l+3)}}Y_{l+1,m+1}^c(\hat{\Omega}) \end{array}\right] \quad \textbf{5.38}$$

$$\mu_z Y_{l,m}^s(\hat{\Omega}) = \left[+\sqrt{\frac{(l-m)(l+m)}{(2l-1)(2l+1)}}Y_{l-1,m}^s(\hat{\Omega}) + \sqrt{\frac{(l-m+1)(l+m+1)}{(2l+1)(2l+3)}}Y_{l+1,m}^s(\hat{\Omega})\right] \quad \textbf{5.39}$$

Recursion relations Eq. **5.34-39** are used to simplify terms Eq. **5.19** and Eqs. **5.25-26** in the transport equation. Within each of these terms, integrals of the form

$$\int_{4\pi} \mu_i Y_{l,m}^{\alpha}(\hat{\Omega}) Y_{l',m'}^{\beta}(\hat{\Omega}) d\hat{\Omega}$$

$i \in \{x, y, z\}$, $\alpha, \beta \in \{c, s\}$, are present. In order to carry out these integrations, the appropriate recursion relation is substituted into this integral and the orthogonality of the spherical harmonics is employed. For example, let $i = x$ and $\alpha = \beta = c$. The integral in question is thus:

$$\int_{4\pi} \mu_x Y^c_{l,m}(\hat{\Omega}) Y^c_{l',m'}(\hat{\Omega}) d\hat{\Omega}$$

Substituting recursion formula Eq. **5.34**:

$$\int_{4\pi} \mu_x Y^c_{l,m}(\hat{\Omega}) Y^c_{l',m'}(\hat{\Omega}) d\hat{\Omega} = \int_{4\pi} \frac{1}{2} \left[\begin{array}{l} -\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m-1}}} \sqrt{\frac{(l+m)(l+m-1)}{(2l-1)(2l+1)}} Y^c_{l-1,m-1}(\hat{\Omega}) \\ +\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m+1}}} \sqrt{\frac{(l-m)(l-m-1)}{(2l-1)(2l+1)}} Y^c_{l-1,m+1}(\hat{\Omega}) \\ +\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m-1}}} \sqrt{\frac{(l-m+1)(l-m+2)}{(2l+1)(2l+3)}} Y^c_{l+1,m-1}(\hat{\Omega}) \\ -\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m+1}}} \sqrt{\frac{(l+m+1)(l+m+2)}{(2l+1)(2l+3)}} Y^c_{l+1,m+1}(\hat{\Omega}) \end{array} \right] Y^c_{l',m'}(\hat{\Omega}) d\hat{\Omega}$$

Finally, employing the orthogonality of the spherical harmonics Eq. **5.7** to determine the value of the integral:

$$\int_{4\pi} \mu_x Y^c_{l,m}(\hat{\Omega}) Y^c_{l',m'}(\hat{\Omega}) d\hat{\Omega} = \begin{cases} -\frac{1}{2}\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m-1}}} \sqrt{\frac{(l+m)(l+m-1)}{(2l-1)(2l+1)}}, & l' = l-1,\ m' = m-1 \\ \frac{1}{2}\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m+1}}} \sqrt{\frac{(l-m)(l-m-1)}{(2l-1)(2l+1)}}, & l' = l-1,\ m' = m+1 \\ \frac{1}{2}\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m-1}}} \sqrt{\frac{(l-m+1)(l-m+2)}{(2l+1)(2l+3)}}, & l' = l+1,\ m' = m-1 \\ -\frac{1}{2}\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m+1}}} \sqrt{\frac{(l+m+1)(l+m+2)}{(2l+1)(2l+3)}}, & l' = l+1,\ m' = m+1 \\ 0, & \text{otherwise} \end{cases} \quad \textbf{5.40}$$

We can repeat this procedure for all combinations of *i,* α and β. These integrals are listed in Appendix A.

5.4 Recursion Relations for $\mu_i\mu_j Y^{\alpha}_{l,m}(\hat{\Omega})$

It remains to determine the values of the integrals appearing in term Eq. **5.18**. To do this, the recursion relations for terms of the form $\mu_i\mu_j Y^{\alpha}_{l,m}(\hat{\Omega})$, $i, j \in \{x, y, z\}$, $\alpha, \beta \in \{c, s\}$, must be determined. This can be done by using relations Eq. **5.34-39** as a starting point and applying a recursion formula twice. For example, take $i = j = x$ and $\alpha = c$. The recursion formula for $\mu_x\mu_x Y^{c}_{l,m}(\hat{\Omega})$ is given by

$$\mu_x\mu_x Y^{c}_{l,m}(\hat{\Omega}) = \mu_x \frac{1}{2}\begin{bmatrix} -\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m-1}}}\sqrt{\frac{(l+m)(l+m-1)}{(2l-1)(2l+1)}}Y^{c}_{l-1,m-1}(\hat{\Omega}) \\ +\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m+1}}}\sqrt{\frac{(l-m)(l-m-1)}{(2l-1)(2l+1)}}Y^{c}_{l-1,m+1}(\hat{\Omega}) \\ +\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m-1}}}\sqrt{\frac{(l-m+1)(l-m+2)}{(2l+1)(2l+3)}}Y^{c}_{l+1,m-1}(\hat{\Omega}) \\ -\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m+1}}}\sqrt{\frac{(l+m+1)(l+m+2)}{(2l+1)(2l+3)}}Y^{c}_{l+1,m+1}(\hat{\Omega}) \end{bmatrix}$$

$$= \frac{1}{2}\begin{bmatrix} -\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m-1}}}\sqrt{\frac{(l+m)(l+m-1)}{(2l-1)(2l+1)}}\mu_x Y^{c}_{l-1,m-1}(\hat{\Omega}) \\ +\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m+1}}}\sqrt{\frac{(l-m)(l-m-1)}{(2l-1)(2l+1)}}\mu_x Y^{c}_{l-1,m+1}(\hat{\Omega}) \\ +\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m-1}}}\sqrt{\frac{(l-m+1)(l-m+2)}{(2l+1)(2l+3)}}\mu_x Y^{c}_{l+1,m-1}(\hat{\Omega}) \\ -\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m+1}}}\sqrt{\frac{(l+m+1)(l+m+2)}{(2l+1)(2l+3)}}\mu_x Y^{c}_{l+1,m+1}(\hat{\Omega}) \end{bmatrix}$$

$$\mu_x \mu_x Y^c_{l,m}(\hat{\Omega}) = \frac{1}{2}\left[\begin{array}{l}
-\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m-1}}}\sqrt{\frac{(l+m)(l+m-1)}{(2l-1)(2l+1)}}\frac{1}{2}\left[\begin{array}{l}
-\frac{\sqrt{\varepsilon_{m-1}}}{\sqrt{\varepsilon_{m-2}}}\sqrt{\frac{(l+m-2)(l+m-3)}{(2l-3)(2l-1)}}Y^c_{l-2,m-2}(\hat{\Omega}) \\
+\frac{\sqrt{\varepsilon_{m-1}}}{\sqrt{\varepsilon_m}}\sqrt{\frac{(l-m)(l-m-1)}{(2l-3)(2l-1)}}Y^c_{l-2,m}(\hat{\Omega}) \\
+\frac{\sqrt{\varepsilon_{m-1}}}{\sqrt{\varepsilon_{m-2}}}\sqrt{\frac{(l-m+1)(l-m+2)}{(2l-1)(2l+1)}}Y^c_{l,m-2}(\hat{\Omega}) \\
-\frac{\sqrt{\varepsilon_{m-1}}}{\sqrt{\varepsilon_m}}\sqrt{\frac{(l+m-1)(l+m)}{(2l-1)(2l+1)}}Y^c_{l,m}(\hat{\Omega})
\end{array}\right] \\
+\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m+1}}}\sqrt{\frac{(l-m)(l-m-1)}{(2l-1)(2l+1)}}\frac{1}{2}\left[\begin{array}{l}
-\frac{\sqrt{\varepsilon_{m+1}}}{\sqrt{\varepsilon_m}}\sqrt{\frac{(l+m)(l+m-1)}{(2l-3)(2l-1)}}Y^c_{l-2,m}(\hat{\Omega}) \\
+\frac{\sqrt{\varepsilon_{m+1}}}{\sqrt{\varepsilon_{m+2}}}\sqrt{\frac{(l-m-2)(l-m-3)}{(2l-3)(2l-1)}}Y^c_{l-2,m+2}(\hat{\Omega}) \\
+\frac{\sqrt{\varepsilon_{m+1}}}{\sqrt{\varepsilon_m}}\sqrt{\frac{(l-m-1)(l-m)}{(2l-1)(2l+1)}}Y^c_{l,m}(\hat{\Omega}) \\
-\frac{\sqrt{\varepsilon_{m+1}}}{\sqrt{\varepsilon_{m+2}}}\sqrt{\frac{(l+m+1)(l+m+2)}{(2l-1)(2l+1)}}Y^c_{l,m+2}(\hat{\Omega})
\end{array}\right] \\
+\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m-1}}}\sqrt{\frac{(l-m+1)(l-m+2)}{(2l+1)(2l+3)}}\frac{1}{2}\left[\begin{array}{l}
-\frac{\sqrt{\varepsilon_{m-1}}}{\sqrt{\varepsilon_{m-2}}}\sqrt{\frac{(l+m)(l+m-1)}{(2l+1)(2l+3)}}Y^c_{l,m-2}(\hat{\Omega}) \\
+\frac{\sqrt{\varepsilon_{m-1}}}{\sqrt{\varepsilon_m}}\sqrt{\frac{(l-m+2)(l-m+1)}{(2l+1)(2l+3)}}Y^c_{l,m}(\hat{\Omega}) \\
+\frac{\sqrt{\varepsilon_{m-1}}}{\sqrt{\varepsilon_{m-2}}}\sqrt{\frac{(l-m+3)(l-m+4)}{(2l+3)(2l+5)}}Y^c_{l+2,m-2}(\hat{\Omega}) \\
-\frac{\sqrt{\varepsilon_{m-1}}}{\sqrt{\varepsilon_m}}\sqrt{\frac{(l+m+1)(l+m+2)}{(2l+3)(2l+5)}}Y^c_{l+2,m}(\hat{\Omega})
\end{array}\right] \\
-\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m+1}}}\sqrt{\frac{(l+m+1)(l+m+2)}{(2l+1)(2l+3)}}\frac{1}{2}\left[\begin{array}{l}
-\frac{\sqrt{\varepsilon_{m+1}}}{\sqrt{\varepsilon_m}}\sqrt{\frac{(l+m+2)(l+m+1)}{(2l+1)(2l+3)}}Y^c_{l,m}(\hat{\Omega}) \\
+\frac{\sqrt{\varepsilon_{m+1}}}{\sqrt{\varepsilon_{m+2}}}\sqrt{\frac{(l-m)(l-m-1)}{(2l+1)(2l+3)}}Y^c_{l,m+2}(\hat{\Omega}) \\
+\frac{\sqrt{\varepsilon_{m+1}}}{\sqrt{\varepsilon_m}}\sqrt{\frac{(l-m+1)(l-m+2)}{(2l+3)(2l+5)}}Y^c_{l+2,m}(\hat{\Omega}) \\
-\frac{\sqrt{\varepsilon_{m+1}}}{\sqrt{\varepsilon_{m+2}}}\sqrt{\frac{(l+m+3)(l+m+4)}{(2l+3)(2l+5)}}Y^c_{l+2,m+2}(\hat{\Omega})
\end{array}\right]
\end{array}\right]$$

$$\mu_x\mu_x Y^c_{l,m}(\hat{\Omega}) = \frac{1}{4}\left[\begin{array}{l}
+\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m-2}}}\sqrt{\frac{(l+m)(l+m-1)(l+m-2)(l+m-3)}{(2l-3)(2l-1)(2l-1)(2l+1)}}Y^c_{l-2,m-2}(\hat{\Omega}) \\
-\sqrt{\frac{(l-m)(l-m-1)(l+m)(l+m-1)}{(2l-3)(2l-1)(2l-1)(2l+1)}}Y^c_{l-2,m}(\hat{\Omega}) \\
-\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m-2}}}\sqrt{\frac{(l-m+1)(l-m+2)(l+m)(l+m-1)}{(2l-1)(2l-1)(2l+1)(2l+1)}}Y^c_{l,m-2}(\hat{\Omega}) \\
+\sqrt{\frac{(l+m)(l+m)(l+m-1)(l+m-1)}{(2l-1)(2l-1)(2l+1)(2l+1)}}Y^c_{l,m}(\hat{\Omega}) \\
-\sqrt{\frac{(l-m)(l-m-1)(l+m)(l+m-1)}{(2l-3)(2l-1)(2l-1)(2l+1)}}Y^c_{l-2,m}(\hat{\Omega}) \\
+\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m+2}}}\sqrt{\frac{(l-m)(l-m-1)(l-m-2)(l-m-3)}{(2l-3)(2l-1)(2l-1)(2l+1)}}Y^c_{l-2,m+2}(\hat{\Omega}) \\
+\sqrt{\frac{(l-m)(l-m)(l-m-1)(l-m-1)}{(2l-1)(2l-1)(2l+1)(2l+1)}}Y^c_{l,m}(\hat{\Omega}) \\
-\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m+2}}}\sqrt{\frac{(l-m)(l-m-1)(l+m+1)(l+m+2)}{(2l-1)(2l-1)(2l+1)(2l+1)}}Y^c_{l,m+2}(\hat{\Omega}) \\
-\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m-2}}}\sqrt{\frac{(l-m+1)(l-m+2)(l+m)(l+m-1)}{(2l+1)(2l+1)(2l+3)(2l+3)}}Y^c_{l,m-2}(\hat{\Omega}) \\
+\sqrt{\frac{(l-m+1)(l-m+2)(l-m+2)(l-m+1)}{(2l+1)(2l+1)(2l+3)(2l+3)}}Y^c_{l,m}(\hat{\Omega}) \\
+\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m-2}}}\sqrt{\frac{(l-m+1)(l-m+2)(l-m+3)(l-m+4)}{(2l+1)(2l+3)(2l+3)(2l+5)}}Y^c_{l+2,m-2}(\hat{\Omega}) \\
-\sqrt{\frac{(l-m+1)(l-m+2)(l+m+1)(l+m+2)}{(2l+1)(2l+3)(2l+3)(2l+5)}}Y^c_{l+2,m}(\hat{\Omega}) \\
+\sqrt{\frac{(l+m+1)(l+m+1)(l+m+2)(l+m+2)}{(2l+1)(2l+1)(2l+3)(2l+3)}}Y^c_{l,m}(\hat{\Omega}) \\
-\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m+2}}}\sqrt{\frac{(l+m+1)(l+m+2)(l-m)(l-m-1)}{(2l+1)(2l+1)(2l+3)(2l+3)}}Y^c_{l,m+2}(\hat{\Omega}) \\
-\sqrt{\frac{(l-m+1)(l-m+2)(l+m+1)(l+m+2)}{(2l+1)(2l+3)(2l+3)(2l+5)}}Y^c_{l+2,m}(\hat{\Omega}) \\
+\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m+2}}}\sqrt{\frac{(l+m+1)(l+m+2)(l+m+3)(l+m+4)}{(2l+1)(2l+3)(2l+3)(2l+5)}}Y^c_{l+2,m+2}(\hat{\Omega})
\end{array}\right]$$

$$\mu_x\mu_x Y_{l,m}^c(\hat{\Omega}) = \frac{1}{4}\left[\begin{array}{l}
+\dfrac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m-2}}}\sqrt{\dfrac{(l+m)(l+m-1)(l+m-2)(l+m-3)}{(2l-3)(2l-1)(2l-1)(2l+1)}}Y_{l-2,m-2}^c(\hat{\Omega}) \\
-2\sqrt{\dfrac{(l-m)(l-m-1)(l+m)(l+m-1)}{(2l-3)(2l-1)(2l-1)(2l+1)}}Y_{l-2,m}^c(\hat{\Omega}) \\
+\dfrac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m+2}}}\sqrt{\dfrac{(l-m)(l-m-1)(l-m-2)(l-m-3)}{(2l-3)(2l-1)(2l-1)(2l+1)}}Y_{l-2,m+2}^c(\hat{\Omega}) \\
-\dfrac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m-2}}}\sqrt{\dfrac{(l-m+1)(l-m+2)(l+m)(l+m-1)}{(2l-1)(2l-1)(2l+1)(2l+1)}}Y_{l,m-2}^c(\hat{\Omega}) \\
-\dfrac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m-2}}}\sqrt{\dfrac{(l-m+1)(l-m+2)(l+m)(l+m-1)}{(2l+1)(2l+1)(2l+3)(2l+3)}}Y_{l,m-2}^c(\hat{\Omega}) \\
+\dfrac{(l+m)(l+m-1)}{(2l-1)(2l+1)}Y_{l,m}^c(\hat{\Omega})+\dfrac{(l-m)(l-m-1)}{(2l-1)(2l+1)}Y_{l,m}^c(\hat{\Omega}) \\
+\dfrac{(l-m+1)(l-m+2)}{(2l+1)(2l+3)}Y_{l,m}^c(\hat{\Omega})+\dfrac{(l+m+1)(l+m+2)}{(2l+1)(2l+3)}Y_{l,m}^c(\hat{\Omega}) \\
-\dfrac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m+2}}}\sqrt{\dfrac{(l-m)(l-m-1)(l+m+1)(l+m+2)}{(2l-1)(2l-1)(2l+1)(2l+1)}}Y_{l,m+2}^c(\hat{\Omega}) \\
-\dfrac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m+2}}}\sqrt{\dfrac{(l+m+1)(l+m+2)(l-m)(l-m-1)}{(2l+1)(2l+1)(2l+3)(2l+3)}}Y_{l,m+2}^c(\hat{\Omega}) \\
+\dfrac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m-2}}}\sqrt{\dfrac{(l-m+1)(l-m+2)(l-m+3)(l-m+4)}{(2l+1)(2l+3)(2l+3)(2l+5)}}Y_{l+2,m-2}^c(\hat{\Omega}) \\
-2\sqrt{\dfrac{(l-m+1)(l-m+2)(l+m+1)(l+m+2)}{(2l+1)(2l+3)(2l+3)(2l+5)}}Y_{l+2,m}^c(\hat{\Omega}) \\
+\dfrac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m+2}}}\sqrt{\dfrac{(l+m+1)(l+m+2)(l+m+3)(l+m+4)}{(2l+1)(2l+3)(2l+3)(2l+5)}}Y_{l+2,m+2}^c(\hat{\Omega})
\end{array}\right]$$

$$\mu_x\mu_x Y^c_{l,m}(\hat{\Omega}) = \frac{1}{4}\left[\begin{array}{l}
+\dfrac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m-2}}}\sqrt{\dfrac{(l+m)(l+m-1)(l+m-2)(l+m-3)}{(2l-3)(2l-1)(2l-1)(2l+1)}}Y^c_{l-2,m-2}(\hat{\Omega}) \\
-2\sqrt{\dfrac{(l-m)(l-m-1)(l+m)(l+m-1)}{(2l-3)(2l-1)(2l-1)(2l+1)}}Y^c_{l-2,m}(\hat{\Omega}) \\
+\dfrac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m+2}}}\sqrt{\dfrac{(l-m)(l-m-1)(l-m-2)(l-m-3)}{(2l-3)(2l-1)(2l-1)(2l+1)}}Y^c_{l-2,m+2}(\hat{\Omega}) \\
-2\dfrac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m-2}}}\dfrac{\sqrt{(l-m+1)(l-m+2)(l+m)(l+m-1)}}{(2l-1)(2l+3)}Y^c_{l,m-2}(\hat{\Omega}) \\
+4\dfrac{l^2+l+m^2-1}{(2l-1)(2l+3)}Y^c_{l,m}(\hat{\Omega}) \\
-2\dfrac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m+2}}}\dfrac{\sqrt{(l+m+1)(l+m+2)(l-m)(l-m-1)}}{(2l-1)(2l+3)}Y^c_{l,m+2}(\hat{\Omega}) \\
+\dfrac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m-2}}}\sqrt{\dfrac{(l-m+1)(l-m+2)(l-m+3)(l-m+4)}{(2l+1)(2l+3)(2l+3)(2l+5)}}Y^c_{l+2,m-2}(\hat{\Omega}) \\
-2\sqrt{\dfrac{(l-m+1)(l-m+2)(l+m+1)(l+m+2)}{(2l+1)(2l+3)(2l+3)(2l+5)}}Y^c_{l+2,m}(\hat{\Omega}) \\
+\dfrac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m+2}}}\sqrt{\dfrac{(l+m+1)(l+m+2)(l+m+3)(l+m+4)}{(2l+1)(2l+3)(2l+3)(2l+5)}}Y^c_{l+2,m+2}(\hat{\Omega})
\end{array}\right]$$

A similar procedure is used to determine the recursion relations for the other combinations of *i, j,* α and β. These are tabulated in Appendix A.

With these relations derived, it is now possible to evaluate integrals of the form

$$\int_{4\pi} \mu_i\mu_j Y^{\alpha}_{l,m}(\hat{\Omega}) Y^{\beta}_{l',m'}(\hat{\Omega}) d\hat{\Omega}$$

that are present in Eq. **5.18**. As an illustration, consider $i = j = x$ and $\alpha = \beta = c$. First, we substitute the recursion relation for $\mu_x\mu_x Y^c_{l,m}(\hat{\Omega})$ into the integral

$$\int_{4\pi} \mu_x \mu_x Y^c_{l,m}(\hat{\Omega}) Y^c_{l',m'}(\hat{\Omega}) d\hat{\Omega} =$$

$$\int_{4\pi} \frac{1}{4} \left[\begin{array}{l}
+\dfrac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m-2}}} \sqrt{\dfrac{(l+m)(l+m-1)(l+m-2)(l+m-3)}{(2l-3)(2l-1)(2l-1)(2l+1)}} Y^c_{l-2,m-2}(\hat{\Omega}) \\
-2\sqrt{\dfrac{(l-m)(l-m-1)(l+m)(l+m-1)}{(2l-3)(2l-1)(2l-1)(2l+1)}} Y^c_{l-2,m}(\hat{\Omega}) \\
+\dfrac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m+2}}} \sqrt{\dfrac{(l-m)(l-m-1)(l-m-2)(l-m-3)}{(2l-3)(2l-1)(2l-1)(2l+1)}} Y^c_{l-2,m+2}(\hat{\Omega}) \\
-2\dfrac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m-2}}} \dfrac{\sqrt{(l-m+1)(l-m+2)(l+m)(l+m-1)}}{(2l-1)(2l+3)} Y^c_{l,m-2}(\hat{\Omega}) \\
+4\dfrac{l^2+l+m^2-1}{(2l-1)(2l+3)} Y^c_{l,m}(\hat{\Omega}) \\
-2\dfrac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m+2}}} \dfrac{\sqrt{(l+m+1)(l+m+2)(l-m)(l-m-1)}}{(2l-1)(2l+3)} Y^c_{l,m+2}(\hat{\Omega}) \\
+\dfrac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m-2}}} \sqrt{\dfrac{(l-m+1)(l-m+2)(l-m+3)(l-m+4)}{(2l+1)(2l+3)(2l+3)(2l+5)}} Y^c_{l+2,m-2}(\hat{\Omega}) \\
-2\sqrt{\dfrac{(l-m+1)(l-m+2)(l+m+1)(l+m+2)}{(2l+1)(2l+3)(2l+3)(2l+5)}} Y^c_{l+2,m}(\hat{\Omega}) \\
+\dfrac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m+2}}} \sqrt{\dfrac{(l+m+1)(l+m+2)(l+m+3)(l+m+4)}{(2l+1)(2l+3)(2l+3)(2l+5)}} Y^c_{l+2,m+2}(\hat{\Omega})
\end{array} \right] \times Y^c_{l',m'}(\hat{\Omega}) d\hat{\Omega}$$

This integral evaluates to

$$\int_{4\pi} \mu_x \mu_x Y^c_{l,m}(\hat{\Omega}) Y^c_{l',m'}(\hat{\Omega}) d\hat{\Omega} =$$

$$\begin{cases}
\frac{1}{4}\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m-2}}}\sqrt{\frac{(l+m)(l+m-1)(l+m-2)(l+m-3)}{(2l-3)(2l-1)(2l-1)(2l+1)}}, & l'=l-2,\ m'=m-2 \\
-\frac{1}{2}\sqrt{\frac{(l-m)(l-m-1)(l+m)(l+m-1)}{(2l-3)(2l-1)(2l-1)(2l+1)}}, & l'=l-2,\ m'=m \\
+\frac{1}{4}\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m+2}}}\sqrt{\frac{(l-m)(l-m-1)(l-m-2)(l-m-3)}{(2l-3)(2l-1)(2l-1)(2l+1)}}, & l'=l-2,\ m'=m+2 \\
-\frac{1}{2}\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m-2}}}\frac{\sqrt{(l-m+1)(l-m+2)(l+m)(l+m-1)}}{(2l-1)(2l+3)}, & l'=l,\ m'=m-2 \\
\frac{l^2+l+m^2-1}{(2l-1)(2l+3)}, & l'=l,\ m'=m \\
-\frac{1}{2}\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m+2}}}\frac{\sqrt{(l+m+1)(l+m+2)(l-m)(l-m-1)}}{(2l-1)(2l+3)}, & l'=l,\ m'=m+2 \\
\frac{1}{4}\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m-2}}}\sqrt{\frac{(l-m+1)(l-m+2)(l-m+3)(l-m+4)}{(2l+1)(2l+3)(2l+3)(2l+5)}}, & l'=l+2,\ m'=m-2 \\
-\frac{1}{2}\sqrt{\frac{(l-m+1)(l-m+2)(l+m+1)(l+m+2)}{(2l+1)(2l+3)(2l+3)(2l+5)}}, & l'=l+2,\ m'=m \\
\frac{1}{4}\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m+2}}}\sqrt{\frac{(l+m+1)(l+m+2)(l+m+3)(l+m+4)}{(2l+1)(2l+3)(2l+3)(2l+5)}}, & l'=l+2,\ m'=m+2 \\
0, & \text{otherwise}
\end{cases}$$

Analogous integrals appearing in Eq. **5.18** that involve other combinations of *x, y, z, c,* and *s* are presented in Appendix A.

5.5 Implementation in SCEPTRE

By using the recursion relations, the transport equation truncated at a finite-order angular expansion can now be written as a finite set of spatially-dependent flux moments in which a given moment with indices *l* and *m* is coupled with (at most) the flux moments

whose indices range from $l-2$ to $l+2$ and from $m-2$ to $m+2$. To implement these recursion relations in SCEPTRE, two subroutines were created; one returns the values of the first integral considered:

$$\int_{4\pi} \mu_i Y_{l,m}^{\alpha}(\hat{\Omega}) Y_{l',m'}^{\beta}(\hat{\Omega}) d\hat{\Omega}$$

accepting as input parameters *i*, the direction of the angular cosine μ_i, the indices *l, m, l′* and *m′*, and α and β, the phase of the spherical harmonics appearing in the integral. The other subroutine returns the values of the integral

$$\int_{4\pi} \mu_i \mu_j Y_{l,m}^{\alpha}(\hat{\Omega}) Y_{l',m'}^{\beta}(\hat{\Omega}) d\hat{\Omega}$$

This subroutine accepts as input parameters those listed above, with an additional parameter *j* to specify the direction of the second angular cosine μ_j. SCEPTRE uses these moment values to construct the transport matrix that is the discrete representation of the continuous operator on the left-hand side of equation Eq. **5.13**, as well as the source terms on the right-hand side. The subroutines created to calculate the moment values are provided in Appendix B. This matrix equation is then solved using a Krylov subspace method, and the flux and particle current values of interest can be extracted from the converged solution comprising of the angular moments of the angular flux.

In the next chapter, the numerical tests used to verify that the recursion formulas were derived correctly will be described. Then, two benchmark problems used to verify that the spherical harmonics methodology is functioning correctly will be described, and numerical results presented and discussed.

Chapter 6

Numerical Results

The previous chapter presented the derivation of the spherical harmonics equations for second order forms of the Boltzmann equation and showed which quantities were necessary to be computed in order to implement this solution technique into the transport code SCEPTRE. The code uses the values of integrals Eqs. **5.41** and **5.42** to construct the matrix used to solve the angular and spatial dependences simultaneously, as described in the first chapter. It is obviously necessary to perform a series of checks to ensure that the integrals are computed correctly, and that the subroutines are implemented correctly into the code. To this end, two phases of verification were carried out. First, the values of the moment integrals for a number of indices were calculated and compared with two independent sets of reference values. This ensured that the individual integrals were being calculated correctly. Then, two benchmark problems were run to solve the transport equation in two-dimensional geometry. The computed flux values were compared with reference values to verify that the subroutines were implemented into SCEPTRE successfully and to evaluate the accuracy of the resulting solution relative to reference solutions for the employed benchmarks. The results of these calculations are now presented.

6.1 Verification of Moment Value Calculation

6.1.1 Verification Against *Mathematica*

In order to ensure the correctness of the derived spherical harmonics moments, a two-level comparison scheme was implemented before the subroutines were included in SCEPTRE. The first level consisted of calculating the values of the spherical harmonics moments via *Mathematica*. These values were then implemented into a driver routine designed to test the analytically derived moments. This driver routine computed the difference between the analytical moment and the moment computed by *Mathematica* to 14 digits of precision for each given value of *l, l′, m* and *m′*. The maximum difference for each μ_i, $i \in \{x, y, z\}$ and $\alpha, \beta \in \{c, s\}$ for the integrals of the form $\int_{4\pi} \mu_i Y_{l,m}^{\alpha}(\hat{\Omega}) Y_{l',m'}^{\beta}(\hat{\Omega}) d\hat{\Omega}$ and for each combination of $\mu_i \mu_j$, $i, j \in \{x, y, z\}$ and $\alpha, \beta \in \{c, s\}$ for the integrals of the form $\int_{4\pi} \mu_i \mu_j Y_{l,m}^{\alpha}(\hat{\Omega}) Y_{l',m'}^{\beta}(\hat{\Omega}) d\hat{\Omega}$ was reported, as well as the maximum overall difference. These calculations were performed for $l = 5, 7$ and 9 to ensure that a thorough check of the moments was carried out. The results of these calculations indicated that the derived formulas for the moment integrals were computed correctly, as the maximum reported difference was on the order of 10^{-15}. This indicates agreement to all 14 digits of the moment values calculated by *Mathematica*.

6.1.2 Verification Against Numerical Quadrature Integration

Once it was established that the moments computed via the analytical formulas were consistent with those computed numerically via *Mathematica*, the second level of the comparison strategy was implemented. In this stage the analytical values of the integrals Eqs. **5.41-42** were compared with the estimated values of the moments computed via a discrete ordinates approximation. The discrete ordinates approximation is a numerical technique used to approximate angular integrals within a given domain using a quadrature formula [1]. The function $f(\hat{\Omega})$ is discretized in angle and the integral over angle is replaced by the following formula:

$$\int_{4\pi} f(\hat{\Omega}) d\hat{\Omega} \approx \sum_{n=1}^{N_d} w_n f(\hat{\Omega}_n)$$

where N_d is the number of angles chosen in the angular discretization and w_n is the weight associated with angle $\hat{\Omega}_n$ in the quadrature formula. Prior to the introduction of the subroutines computing the exact integrals of the spherical harmonics moments derived in the present work, SCEPTRE relied on using an S_{20} angular quadrature of the fully symmetric type to approximate the values of these integrals. In this quadrature, there are $N(N+2) = 20 \times 22 = 440$ discrete directions. The integrals approximated were thus

$$\int_{4\pi} \mu_i Y^{\alpha}_{l,m}(\hat{\Omega}) Y^{\beta}_{l',m'}(\hat{\Omega}) d\hat{\Omega} \approx \sum_{n=1}^{440} w_n \mu_{i,n} Y^{\alpha}_{l,m}(\hat{\Omega}_n) Y^{\beta}_{l',m'}(\hat{\Omega}_n)$$

and

$$\int_{4\pi} \mu_i \mu_j Y_{l,m}^{\alpha}(\hat{\Omega}) Y_{l',m'}^{\beta}(\hat{\Omega}) d\hat{\Omega} \approx \sum_{n=1}^{440} w_n \mu_{i,n} \mu_{j,n} Y_{l,m}^{\alpha}(\hat{\Omega}_n) Y_{l',m'}^{\beta}(\hat{\Omega}_n)$$

with i, j, α and β having the usual meanings. One would expect that the integrals are approximated very well with such a refined quadrature.

To perform the comparison, first, a Legendre order l was specified. Then, the difference between the exact values of Eqs. **4.41-42** computed by the newly implemented subroutines and the approximate values produced by the quadrature formulas as originally implemented in SCEPTRE was determined for each dimension i for the integrals of the form $\int_{4\pi} \mu_i Y_{l,m}^{\alpha}(\hat{\Omega}) Y_{l',m'}^{\beta}(\hat{\Omega}) d\hat{\Omega}$, and each combination of dimensions i and j for the integrals of the form $\int_{4\pi} \mu_i \mu_j Y_{l,m}^{\alpha}(\hat{\Omega}) Y_{l',m'}^{\beta}(\hat{\Omega}) d\hat{\Omega}$, for each l' ranging from $l-2$ to $l+2$ and each m' ranging from $m-2$ to $m+2$. The maximum difference between each of these individual moments was reported for each given l. In addition to the maximum differences, for each dimension (in the first case) or combination of dimensions (in the second case), the L_1 norm (maximum column sum) and L_∞ norm (maximum row sum) of the matrix containing the differences between the quadrature approximation and exact values of the moment integrals was reported. In both the first and second order cases, differences and norms were calculated for each value of l from $l = 3$ to $l = 8$. The values of l from $l = 0$ through $l = 2$ were not included in this comparison because P_2 calculations using the discrete ordinates approximation of the angular integrals were successfully run with SCEPTRE; it was of more interest to evaluate the approximation strategy for higher-order moments.

The comparisons indicated agreement between the exact values and the quadrature approximations out to 14 digits of precision for values of l up to 5 for the first integral and up to 4 for the second integral. For values of l greater than these numbers, the maximum differences and both of the norms for each dimension were comparable to the values of the moment integrals themselves. The strong agreement for lower values of l indicated that the exact values are consistent with the quadrature approximation. The large difference between the approximations and the exact values for the larger values of l is troubling at first consideration, but when viewed with the good agreement of the derived expressions with those computed by *Mathematica*, the most logical conclusion that can be drawn is that the quadrature approximation of the moment integrals breaks down at the values of l given above. This reinforces the importance of deploying the exact moment calculations; without them, it would be inaccurate to use SCEPTRE to perform high-order spherical harmonics calculations.

6.2 Test Problems

After gaining confidence that the moment integral calculations were correctly implemented into the subroutines, it was necessary to ensure that the subroutines were integrated with the rest of the SCEPTRE code successfully. In addition, these exercises are designed to evaluate the accuracy of SCEPTRE's solution with the newly installed subroutines. To accomplish this, SCEPTRE was exercised on two simple test problems. The first problem incorporated the Method of Manufactured Solutions to generate a comparison with an analytic solution, while the second relied on comparison with

numerical data. In both cases, the one-speed transport equation was solved on a square geometry.

6.2.1 Method of Manufactured Solutions

It was decided that the first test of the spherical harmonics routines within SCEPTRE should be carried out by performing a comparison with an analytic solution. The analytic solution is generated via the Method of Manufactured Solutions (MMS). The procedure for MMS is conceptually very simple. One chooses an arbitrary, but reasonable, analytic representation of the flux solution over the spatial and angular domains. This solution is inserted into the Boltzmann equation, and the corresponding source and boundary conditions can be determined easily since this does not involve operator inversions. The source distribution and boundary conditions are then implemented into the computer code, and a resulting numerical flux solution is generated. This numerical flux distribution is compared with the original stipulated flux shape, and an error analysis can be undertaken to verify that the numerical solution is converging as expected to the pre-selected analytic solution. The correct implementation of the spherical harmonics subroutines is implied if the numerical solution converges with the expected asymptotic order.

6.2.1.1 Problem Description

The MMS test problem considered is adapted from one given in [24]. We begin by considering the unit square (Figure **6.1**) $0 \le x \le 1,\ 0 \le y \le 1$:

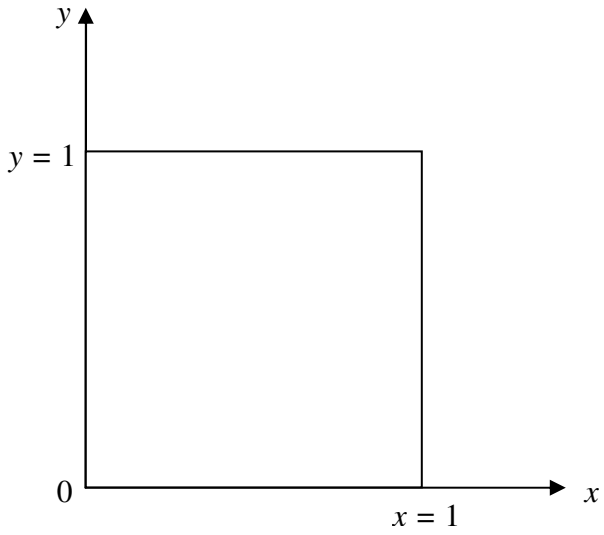

Figure **6.1**: Geometry for MMS Problem

Within the unit square, we assume a unit total cross section $\sigma = 1$ with an absorption cross section $\sigma_a = 0.5$. The scattering cross section considered is linearly anisotropic and has the form $\sigma_s(\mu_0) = \frac{1}{2\pi}\left(\frac{1}{4} + \frac{\mu_0}{4}\right)$, where μ_0 denotes the cosine of the scattering angle. The relationship between μ_0 and the directions of travel pre- and post-collision θ, ϕ and θ', ϕ', respectively, is given as:

$$\mu_0 = \mu_z \mu_z' + \sin\theta \sin\theta' \cos(\phi - \phi')$$

where $\mu_z = \cos\theta$. The Legendre moments of the scattering cross section are calculated as:

$$\sigma_{s0} = 2\pi \int_{-1}^{1} \sigma_s(\mu_0) P_0(\mu_0) d\mu_0$$
$$= 2\pi \int_{-1}^{1} \frac{1}{2\pi} \left(\frac{1}{4} + \frac{\mu_0}{4} \right) (1) d\mu_0$$
$$= \frac{1}{2}$$

and

$$\sigma_{s1} = 2\pi \int_{-1}^{1} \sigma_s(\mu_0) P_1(\mu_0) d\mu_0$$
$$= 2\pi \int_{-1}^{1} \frac{1}{2\pi} \left(\frac{1}{4} + \frac{\mu_0}{4} \right) \mu_0 d\mu_0$$
$$= \frac{1}{6}$$

The flux solution we consider is given as:

$$\psi(x, y, \mu_z) = \frac{e^{-x} e^{-y}}{1 + \mu_z^2} \qquad \textbf{6.1}$$

Note in this case that $\psi(x, y, \mu_z) = \psi(x, y, -\mu_z)$, so the even parity flux $\psi^e(x, y, \mu_z)$ $= \psi(x, y, \mu_z)$ and the odd parity flux $\psi^o(x, y, \mu_z) = 0$. The angular dependence of the flux solution Eq. **6.1** was chosen such that it could not be exactly represented by a polynomial expansion (which would result in an exact flux solution after a finite number of terms). A simple exponential spatial dependence was stipulated such that the spatial derivatives appearing in the transport equation would not evaluate to zero. The first-order one-speed Boltzmann equation in two-dimensional Cartesian geometry (assuming reflection about the *z*-axis), using the form of the scattering cross section defined above, is given by:

$$\mu_x \frac{\partial \psi}{\partial x} + \mu_y \frac{\partial \psi}{\partial y} + \psi(x, y, \mu_z) - \int_0^{2\pi} \int_{-1}^{1} \frac{1}{2\pi} \left(\frac{1}{4} + \frac{\mu_0}{4} \right) \psi(x, y, \mu_z') d\mu_z' d\phi' = Q(x, y, \mu_x, \mu_y, \mu_z)$$

In order to determine the form of the source term in this equation, it is necessary to evaluate the derivatives and integrals on the left hand side using Eq. **6.1**. This was done using the *Mathematica* software package. The resulting source distribution is given by the following expression:

$$Q(x, y, \mu_x, \mu_y, \mu_z) = e^{-x} e^{-y} \frac{1 - \frac{\pi}{8} - \frac{\pi}{8} \mu_z^2 - \mu_x - \mu_y}{1 + \mu_z^2} \qquad \textbf{6.2}$$

Recall the definition of μ_x and μ_y as functions of the polar and azimuthal angles:

$$\mu_x(\theta, \varphi) = \cos\varphi \sin\theta$$
$$\mu_y(\theta, \varphi) = \sin\varphi \sin\theta$$

With this definition, the following relations can be determined using the symmetry properties of the trigonometric functions:

$$\mu_x(-\theta, -\varphi) = \cos(-\varphi)\sin(-\theta) = -\cos\varphi \sin\theta = -\mu_x(\theta, \varphi)$$
$$\mu_y(-\theta, -\varphi) = \sin(-\varphi)\sin(-\theta) = \sin\varphi \sin\theta = \mu_y(\theta, \varphi)$$
$$\mu_z(-\theta, -\varphi) = \cos(-\theta) = \cos\theta = \mu_z(\theta, \varphi)$$

Using these expressions, it is now possible to evaluate the even- and odd-parity source moments. The general expression for the even parity source is given by:

$$Q^e(\mathbf{r}, \hat{\Omega}) = \frac{1}{2} \left(Q(\mathbf{r}, \hat{\Omega}) + Q(\mathbf{r}, -\hat{\Omega}) \right)$$

In the present notation, this is equivalent to evaluating

$$Q^e(x,y,\mu_x,\mu_y,\mu_z)=\frac{1}{2}\Big(Q(x,y,\mu_x,\mu_y,\mu_z)+Q(x,y,-\mu_x,\mu_y,\mu_z)\Big)$$
$$=\frac{e^{-x}e^{-y}}{2(1+\mu_z^2)}\left(1-\frac{\pi}{8}-\frac{\pi}{8}\mu_z^2-\mu_x-\mu_y+1-\frac{\pi}{8}-\frac{\pi}{8}\mu_z^2+\mu_x-\mu_y\right)$$

The even-parity source distribution thus reduces to:

$$Q^e(x,y,\mu_x,\mu_y,\mu_z)=\frac{e^{-x}e^{-y}}{1+\mu_z^2}\left(1-\frac{\pi}{8}-\frac{\pi}{8}\mu_z^2-\mu_y\right) \qquad \textbf{6.3}$$

Likewise, we can find the odd-parity source distribution beginning from the definition:

$$Q^o(\mathbf{r},\hat{\Omega})=\frac{1}{2}\Big(Q(\mathbf{r},\hat{\Omega})-Q(\mathbf{r},-\hat{\Omega})\Big)$$

In the present notation, this expression becomes:

$$Q^o(x,y,\mu_x,\mu_y,\mu_z)=\frac{1}{2}\Big(Q(x,y,\mu_x,\mu_y,\mu_z)-Q(x,y,-\mu_x,\mu_y,\mu_z)\Big)$$
$$=\frac{e^{-x}e^{-y}}{2(1+\mu_z^2)}\left(1-\frac{\pi}{8}-\frac{\pi}{8}\mu_z^2-\mu_x-\mu_y-1+\frac{\pi}{8}+\frac{\pi}{8}\mu_z^2-\mu_x+\mu_y\right)$$

and the form of the odd-parity source distribution is determined to be:

$$Q^o(x,y,\mu_x,\mu_y,\mu_z)=-\frac{e^{-x}e^{-y}\mu_x}{1+\mu_z^2} \qquad \textbf{6.4}$$

The even-parity solver was executed to generate the numerical solution of the angular flux distribution for this problem. The even-parity flux incoming and outgoing angular flux boundary conditions on each face in Figure **6.1** are found using Eq. **6.1**, and are given by:

$$\psi^e\left(0,y,\mu_x,\mu_y\right)=\frac{e^{-y}}{2-\mu_x^2-\mu_y^2},\ \mu_x>0 \text{ (Incoming flux at } x=0)$$

$$\psi^e\left(0,y,\mu_x,\mu_y\right)=\frac{e^{-y}}{2-\mu_x^2-\mu_y^2},\ \mu_x<0 \text{ (Outgoing flux at } x=0)$$

$$\psi^e\left(1,y,\mu_x,\mu_y\right)=\frac{e^{1-y}}{2-\mu_x^2-\mu_y^2},\ \mu_x<0 \text{ (Incoming flux at } x=1)$$

$$\psi^e\left(1,y,\mu_x,\mu_y\right)=\frac{e^{1-y}}{2-\mu_x^2-\mu_y^2},\ \mu_x>0 \text{ (Outgoing flux at } x=1)$$

$$\psi^e\left(x,0,\mu_x,\mu_y\right)=\frac{e^{-x}}{2-\mu_x^2-\mu_y^2},\ \mu_y>0 \text{ (Incoming flux at } y=0)$$

$$\psi^e\left(x,0,\mu_x,\mu_y\right)=\frac{e^{-x}}{2-\mu_x^2-\mu_y^2},\ \mu_y<0 \text{ (Outgoing flux at } y=0)$$

$$\psi^e\left(x,1,\mu_x,\mu_y\right)=\frac{e^{1-x}}{2-\mu_x^2-\mu_y^2},\ \mu_y<0 \text{ (Incoming flux at } y=1)$$

$$\psi^e\left(x,1,\mu_x,\mu_y\right)=\frac{e^{1-x}}{2-\mu_x^2-\mu_y^2},\ \mu_y>0 \text{ (Outgoing flux at } y=1) \qquad \textbf{6.5}$$

In Eq. **6.5**, we have made use of the fact that $\mu_z^2=1-\mu_x^2-\mu_y^2$. Supplying SCEPTRE with the source distributions, Eqs. **6.3-4**, and boundary conditions, Eq. **6.5**, it is now possible to compute a numerical solution for the flux distribution.

6.2.1.2 Numerical Results

In order to perform this computation, a quadrilateral mesh of varying size was imposed over the unit square. There are *n* equally sized finite elements along a given direction, each with side length $h = 1/n$. Therefore, in a given mesh, there are n^2 equally sized finite elements in total. Figure **6.2** shows a sample mesh.

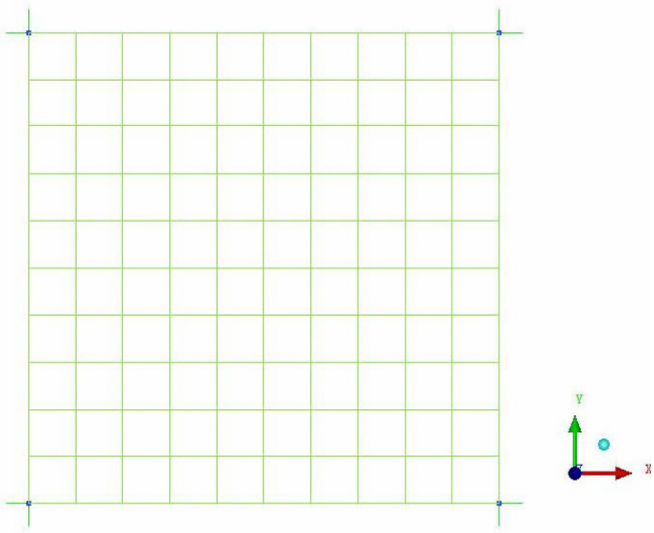

Figure **6.2**: Sample quadrilateral mesh for MMS problem

There were two sequences of runs done for this problem. The first was performed using a simple linear finite elements scheme where there were four computational nodes for each quadrilateral located at its corners. When the angular expansion is of sufficiently high order, this spatial discretization scheme should show a convergence rate of 2 [24]. The second series of runs were done using a finite elements scheme employing eight computational nodes on a given quadrilateral: one on each corner, and one at the midpoint of each side. The theoretical convergence rate of this scheme is 3 [24]. For each sequence of runs, the mesh size was varied; in the first sequence, the mesh was varied from $h = 1$ (one finite element) to $h = 0.0625$ (1024 finite elements), with the mesh size halved for each subsequent mesh. Then, for each spatial mesh, the order of angular approximation was varied in increments of 2 from P_2 to P_{10}. This produced a total of 25 separate runs. For the second sequence of runs, three meshes were used, from $h = 1$ to $h = 0.25$, again halving the mesh size between each pair of consecutive meshes.

Fewer meshes were necessary because of the heightened accuracy of the spatial discretization stencil. Again, the angular approximation was varied in increments of 2, where the angular approximation order varied from P_2 to P_{12}. The cutoff for the angular approximation was chosen when the error norm displayed linear behavior with decreasing mesh size on a log-log scale indicating a power-law asymptotic convergence.

The following procedure was followed for each run. First, the even- and odd-parity fixed source distributions and boundary conditions were input into SCEPTRE. Then, the order of the P_N angular discretization and the appropriate mesh were specified. The even-parity transport solver was executed to generate a pointwise angular flux solution, and the L_2 error norm was calculated to give an integral representation of the difference between the numerical solution and the analytic solution. This norm is defined as:

$$\left\|\psi-\psi_h\right\|_2=\left(\int_{4\pi}\int_{V}\left|\psi-\psi_h\right|^2 dV d\hat{\Omega}\right)^{1/2}$$

where ψ denotes the MMS analytic flux, Eq. **6.1**, ψ_h denotes the numerical solution on the mesh with cell edge-length *h*, and the integral over *V* indicates integration over the unit square. The integral was performed numerically using a fifth-order-accurate technique in space and an S_{20} quadrature to integrate in angle.

The results for the first sequence of runs are presented in Figure **6.3**

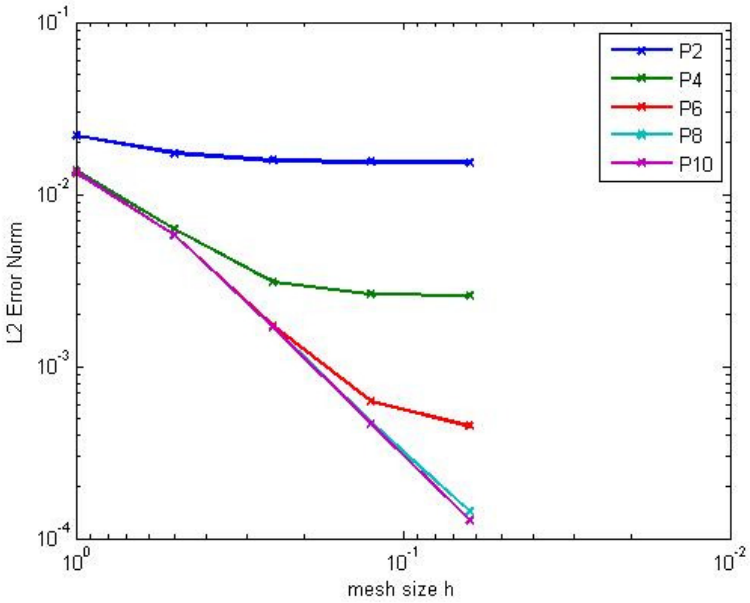

Figure **6.3**: Error norms for MMS Problem run with quadrilateral mesh with 4 nodes per finite element

The results from the second sequence of runs, done on a quadrilateral mesh with 8 nodes per finite element, are shown in Figure **6.4**

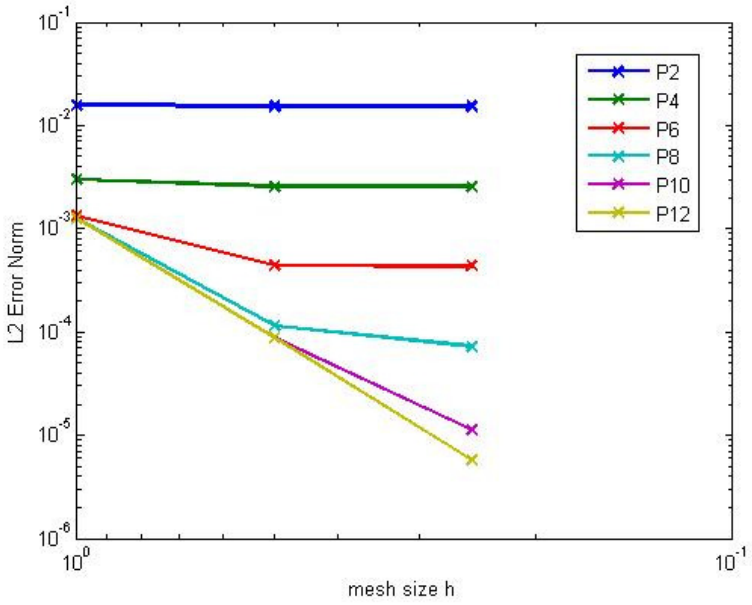

Figure **6.4**: Error norms for MMS Problem run with quadrilateral mesh with 8 nodes per finite element

6.2.1.3 Discussion of Results

From Figure **6.3** , it is clear that convergence in angle has been attained for a P_{10} angular approximation. To determine the rate of convergence in space, a power curve was fit to the error norms on the three finest meshes for the P_{10} series of runs. These points were chosen because they gave the best representation of the asymptotic behavior of the error norm as $h \to 0$. The rate of convergence is given by the exponent of the power curve, and was determined to be 1.8577. This value compares well with the theoretical convergence rate of 2.

In the second sequence of runs, angular convergence is attained with a P_{12} angular approximation. The rate of convergence was again estimated with a power curve fit on this data series, and was determined to be 3.8835. This convergence rate is actually better than the predicted value of 3. A similar sequence of runs was performed on this mesh using a discrete ordinates angular approximation; the convergence rate for this sequence of runs was also close to a value of 4 [25]. This leads to the conjecture that for some problems the quadrilateral mesh with 8 nodes per finite element exhibits superconvergence.

The MMS approach was employed to test the spherical harmonics flux solver implemented in SCEPTRE. The convergence rates in space of the L_2 error norm between the analytic and numerical solution were calculated for the data series showing convergence in angle on two separate finite element types. These convergence rates were found to be in reasonable agreement with the theoretically predicted convergence rates for the mesh with 4 nodes per finite element. The convergence rate resulting from the sequence of runs done with 8 nodes per finite element was found to be in reasonable agreement with a similar sequence of runs using discrete ordinates. In both the spherical harmonics and discrete ordinates solutions, the observed convergence rate was about one order higher than was expected, leading to the conjecture that for this particular problem the finite elements scheme implemented may show superconvergence. Thus, we have a preliminary indication that the spherical harmonics solver is functioning correctly in SCEPTRE.

6.2.2 Azmy Benchmark Problem

6.2.2.1 Problem Description

The second test problem was adapted from a problem developed to test spatial approximations in nodal transport methods [26]. The one-speed transport equation is solved in a square region of side length 10 that has been divided into 4 quadrants; see Figure **6.5**. Within the whole domain, isotropic scattering is considered. In the lower left quadrant (region I), there is a unit volumetric source. Region I is characterized by a unit total cross section and a scattering ratio of 0.5. The lower right, upper left, and upper right quadrants (denoted regions II, III, and IV, respectively, in Figure **6.5**) are source-free and have a total cross section equal to 2 and a scattering cross section of 0.1. Reflective boundary conditions are specified on the left and bottom faces and vacuum boundary conditions are specified on the right and top faces. The problem geometry is shown in Figure **6.5**, and the nuclear and source data are summarized in Table **6.1**.

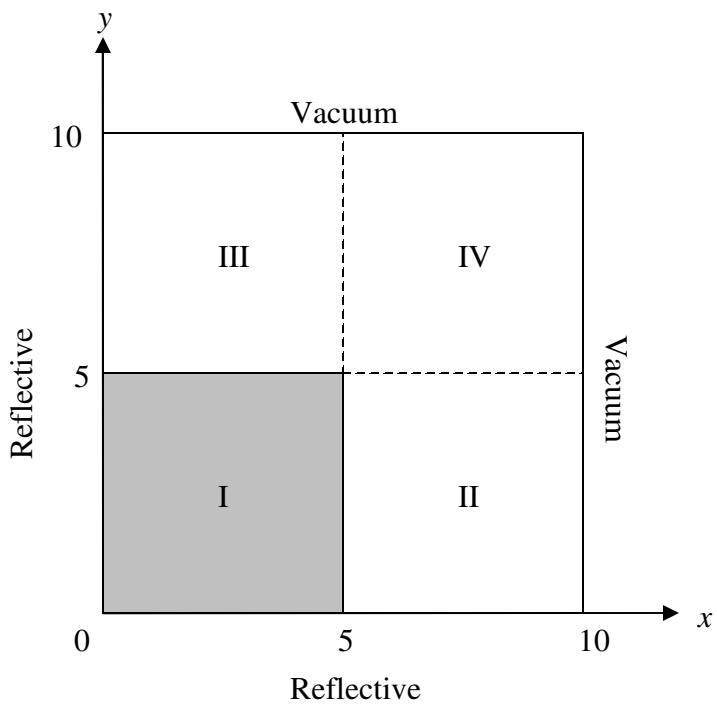

Figure **6.5**: Geometry specification and boundary conditions for second test problem

Table **6.1**: Nuclear and source data for second test problem

Region	σ_t	σ_s	q
I	1	0.5	1
II	2	0.1	0
III	2	0.1	0
IV	2	0.1	0

6.2.2.2 Numerical Results

For this problem, a quadrilateral mesh with nodes at the corners and midpoints of each of the finite elements was imposed on the problem geometry (i.e., 8 nodes per element). Figure **6.6** illustrates a sample mesh for this problem.

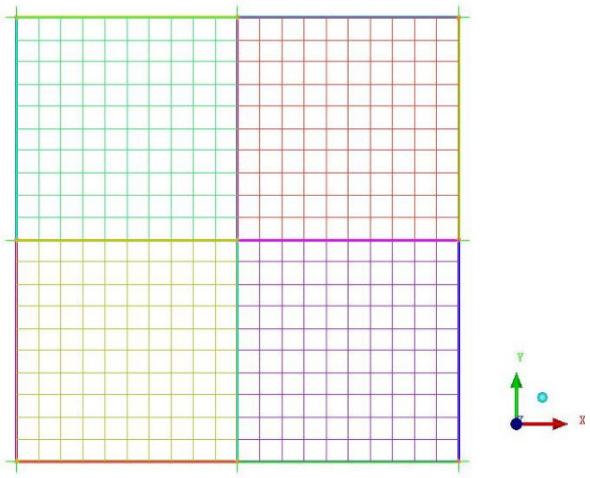

Figure **6.6**: Sample quadrilateral mesh for second test problem

Each side of the square was divided into n equally sized segments of length $h = 10/n$ for n = 20, 40, 80, 160, and 320, for a total of 5 spatial discretization levels. This created a uniform mesh in which the total number of elements is n^2. The source distribution, cross sections, and boundary conditions were provided in the input to SCEPTRE. For each mesh, a spherical harmonics calculation was run for angular approximation order N = 8, 12, 16, 20, 24, 28, and 32. This gave a total of 7 angular approximations. Due to limitations on computational resources, it was not possible to run P_{28} and P_{32} calculations on the meshes with 160 and 320 elements per side and P_{20} and P_{24} calculations on the mesh with 320 elements per side. For each completed run, the

even-parity flux solver was used to generate the scalar flux distribution throughout the square, and the average flux solution within each region was reported by the code. Due to the symmetry present along line $x = y$ in the problem geometry specified in Figure **6.5**, the results in regions II and III were equivalent. Figures **6.7-9** show the average flux solution with decreasing mesh size and increasing angular refinement for regions I, II and III, and IV, respectively.

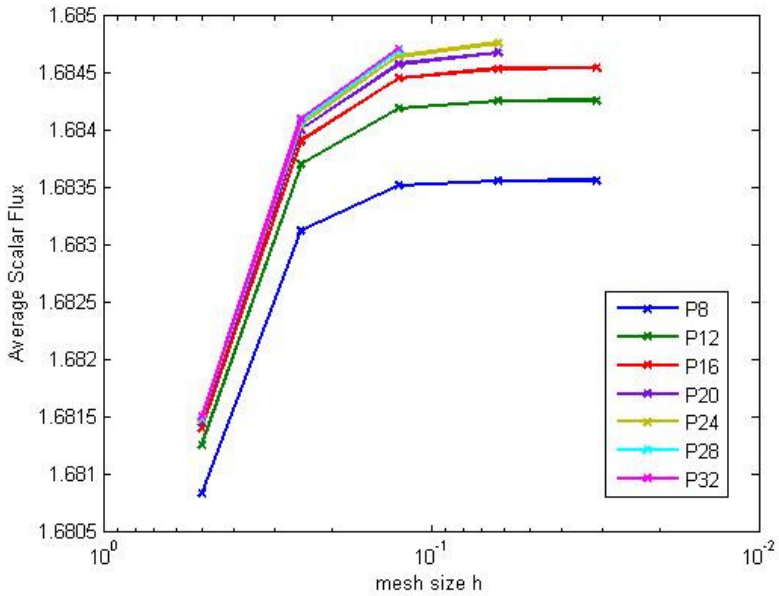

Figure **6.7**: Average scalar flux in region I of the second test problem

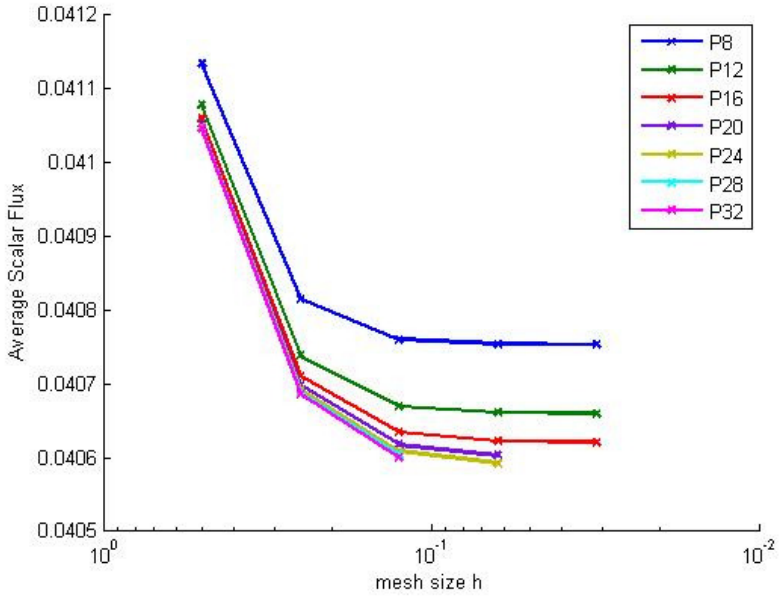

Figure **6.8**: Average scalar flux in regions II and III of the second test problem

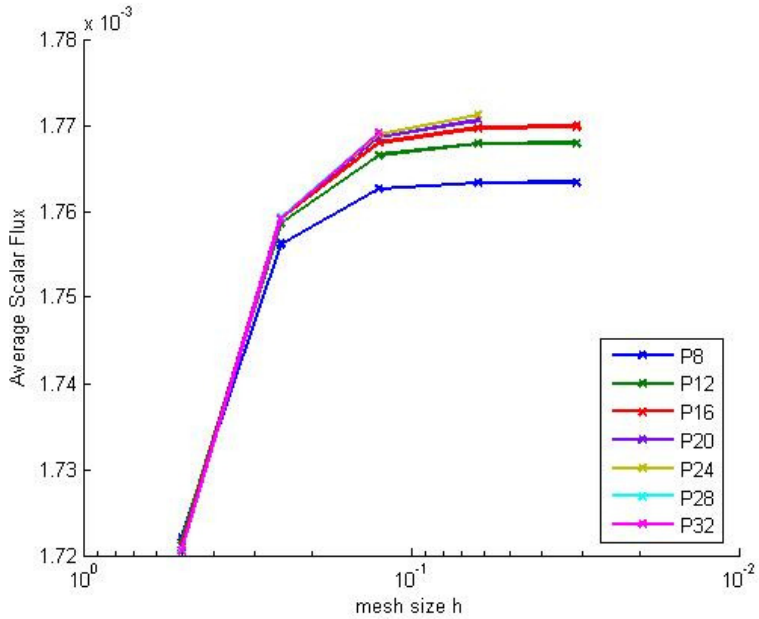

Figure **6.9**: Average scalar flux in region IV of the second test problem

It was also desired to obtain estimates of the convergence rates as the spatial mesh was refined asymptotically, as was performed for the first test problem. Unlike the first problem, however, there were no high-quality reference solutions with which the numerical solutions could be compared. The asymptotically converged solution reported in [26] presumed an S_6 angular quadrature; hence it does not provide a reference solution for the full dependence of the transport equation in both space and angle. Fortunately, techniques exist that give an estimation of both the rate of convergence and the error in the calculation.

The technique chosen for this test problem was the well-known Richardson extrapolation methodology [27]. Suppose we have three data points f_i, each evaluated at a different mesh size h_i, i = 1, 2, 3. We order the mesh sizes such that i = 1 denotes the finest mesh and i = 3 denotes the coarsest mesh. It is also necessary that f_i is monotonic, or more precisely in the asymptotic regime, in the selected mesh size sequence. For simplicity, we define a refinement ratio r such that the following relations hold true:

$$r = \frac{h_2}{h_1} = \frac{h_3}{h_2}$$

In this ordering scheme, it is obvious that $r > 1$. Finally, denote the solution as the mesh size goes to zero as f_{ref}. We postulate that the solution f as a function of mesh size is given by the following relation as the mesh size approaches zero:

$$f(h) = f_{ref} + ah^p \qquad \textbf{6.6}$$

where a and p are parameters to be identified. We evaluate this equation at each of the three data points available:

$$\begin{aligned} f(h_1) = f_1 &= f_{ref} + a h_1^p \\ f(h_2) = f_2 &= f_{ref} + a h_2^p \\ f(h_3) = f_3 &= f_{ref} + a h_3^p \end{aligned} \qquad \textbf{6.7}$$

This is a system of three equations with three unknowns, and thus it is possible to determine these values. Subtract the third equation from the second, and the second equation from the first:

$$\begin{aligned} f_3 - f_2 &= a(h_3^p - h_2^p) \\ f_2 - f_1 &= a(h_2^p - h_1^p) \end{aligned}$$

Next, take the ratio of these two equations:

$$\frac{f_3 - f_2}{f_2 - f_1} = \frac{h_3^p - h_2^p}{h_2^p - h_1^p} = \frac{\left(\frac{h_3^p}{h_2^p} - 1\right) h_2^p}{\left(\frac{h_2^p}{h_1^p} - 1\right) h_1^p}$$

Finally, make use of the refinement ratio r to obtain

$$\frac{f_3 - f_2}{f_2 - f_1} = r^p$$

and thus

$$p = \frac{\ln\left(\frac{f_3 - f_2}{f_2 - f_1}\right)}{\ln r} \qquad \textbf{6.8}$$

The exponent p is known as the rate of convergence, and gives an indication of how fast the numerical solution converges to the reference solution with increasing mesh refinement.

Next, we can determine the value of the coefficient a:

$$a = \frac{f_2 - f_1}{(r^p - 1)h_1^p} \qquad \textbf{6.9}$$

Using p and a, it is now possible to estimate f_{ref}:

$$f_{ref} = f_1 - ah_1^p = f_1 - \frac{f_2 - f_1}{(r^p - 1)} = \frac{r^p}{r^p - 1} f_1 - \frac{1}{r^p - 1} f_2 \qquad \textbf{6.10}$$

The reference solution for the average of the scalar flux within each region shown in Figure **6.5** was obtained using the three data points available on the highest angular discretization (P_{32}). For this problem, the refinement ratio r is equal to 2 for all employed mesh refinements. The reference values are presented in Table **6.2**:

Table **6.2**: Reference solutions based on Richardson Extrapolation

Region	f_{ref}
I	1.684908
II/III	0.040573
IV	0.001773

Using these reference values, the error in the four regions' average scalar flux computed on each mesh for each angular discretization was computed. This data is presented in Figures **6.10-12** for regions I, II and III, and IV, respectively.

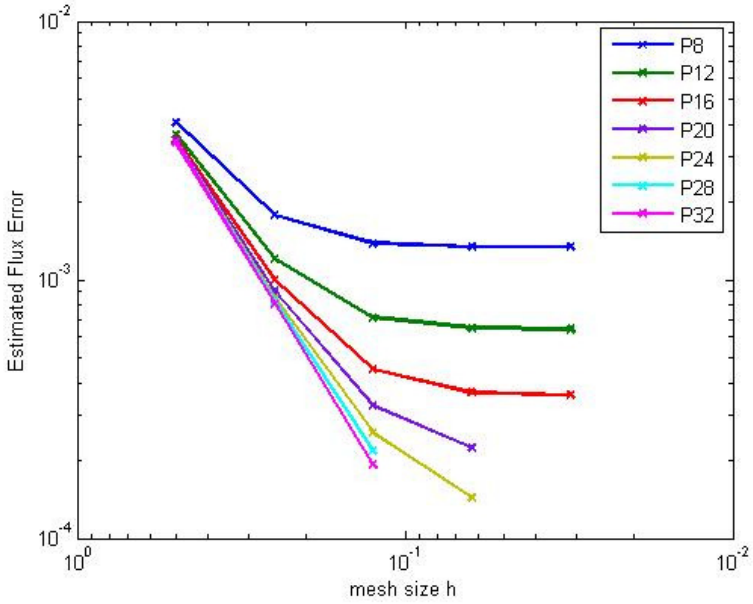

Figure **6.10**: Estimated error in average flux in region I of the second test problem

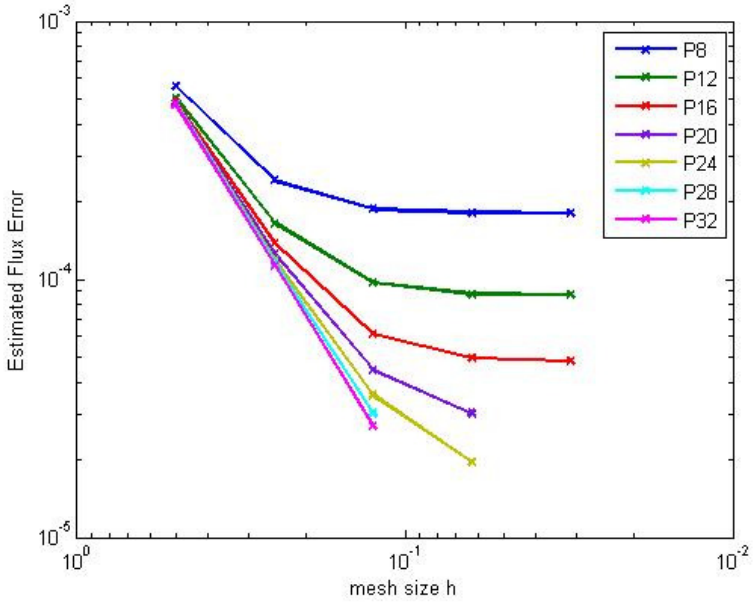

Figure **6.11**: Estimated error in average flux in regions II and III of the second test problem

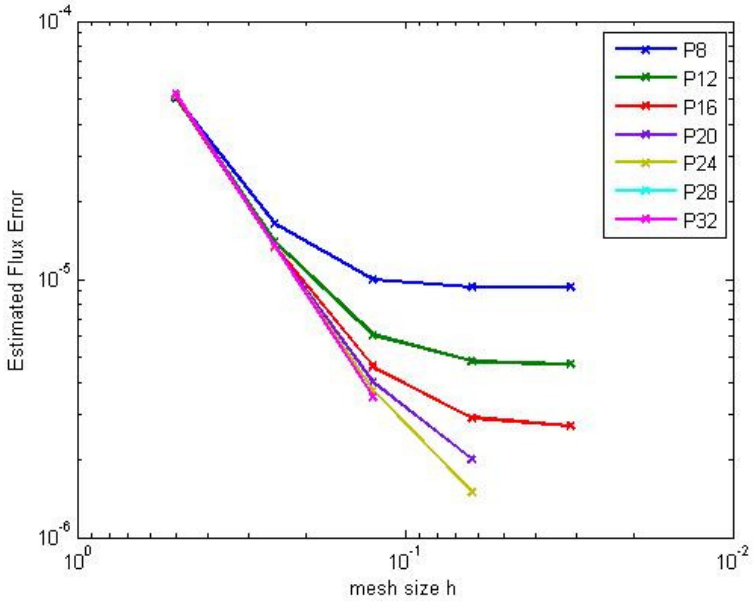

Figure **6.12:** Estimated error in average flux in region IV of the second test problem

6.2.2.3 Discussion of Results

There is a number of trends readily apparent from Figures **6.7-9**. First, for all regions, as the order of angular approximation is increased for a given mesh, the difference between the average flux value and the average flux value for the next-coarser angular refinement becomes smaller. This is an indication of convergence in the angular variables, which one would expect for a successful implementation of the spherical harmonics routines. Second, the magnitude of the average flux in region I is more than an order of magnitude larger than the average flux in regions II and III and almost three orders of magnitude larger than the average flux in region IV. The reason the average

flux in region I is so much larger than the other regions is that region I is the source region. The average flux in region IV is so much smaller than the fluxes in the other non-source regions due to the fact that a particle arriving in region IV must first pass through either region II or region III to get there, since region IV does not share a face with the source region. The cross sections chosen for the non-source regions are characterized by strong absorption (scattering ratio $c = \sigma_s / \sigma_t = 0.1/2 = 0.05$), so it is expected that most of the particles produced in the source region that leak into the adjoining regions will be absorbed within these regions and only a few particles will traverse them and arrive in region IV.

The plots of the error in the average scalar flux within each region (Figures **6.10-12**) display similar behavior to the L_2 error norm in the first test problem; that is, each of the lower-order angular discretizations converges to a different solution as the spatial refinement is increased. The error becomes linear on a log-log plot as the angular approximation order is increased. The convergence rate on the finest angular mesh within each region was computed using Eq. **6.7**, and is presented in Table **6.3**.

Table **6.3**: Convergence rates using P_{32} angular discretization

Region	***p***
I	2.072
II/III	2.066
IV	1.954

It is readily apparent from these computations that in all regions the convergence rate is approximately 2. However, as stated earlier, the convergence rate for the finite elements

used for this test problem should be closer to a value of 3. There is a number of factors that could lead to this discrepancy. First of all, the theoretical convergence rates calculated for a given finite elements mesh are typically arrived at by assuming a generic, smooth solution. It is well documented that solutions to the transport equation can display discontinuities in the first derivative of the flux [28]. Another issue that may be in play is that the errors and convergence rates are calculated for the average scalar flux within each region. In other words, the flux is being averaged, and then the convergence rate of the error is computed. Contrast this with the procedure used in the first problem, where the pointwise error is computed and then implemented into the error calculation. The "smearing" associated with averaging procedures may be negatively affecting the convergence rates. Finally, the finite elements routines used within SCEPTRE were developed for general, unstructured mesh problems. Although the meshes used within this test problem are regular, since the code is built upon an unstructured framework the convergence rates may be akin to the predicted values on these kinds of meshes. There is a lack of analysis on convergence rates on unstructured meshes (since this is a difficult problem), so we do not dwell upon this discrepancy and suggest it as a future area of further investigation.

It is worth noting that similar disagreement with expected convergence rates has been observed in a number of discrete ordinates calculations performed using SCEPTRE [24]. Specifically, it was observed that the convergence rates of the pointwise solutions along the boundaries of the problem domains were much lower than expected. This leads us to conjecture that it is the spatial rather than the angular approximation (specifically

the spatial approximation along the boundaries) which is the main reason for the divergence from expected results.

Chapter 7

Summary and Conclusions

This thesis dealt with spherical harmonics solutions to second-order forms of the Boltzmann transport equation. First, the general transport problem was formulated. Then, the derivation of the Self-Adjoint Angular Flux and the Even/Odd Parity second-order formulations of the transport equation was presented. Next, the derivation of the spherical harmonics approximation to the second-order forms was elaborated, and the coefficients necessary to implement this approximation were derived. Finally, confirmation of the correct evaluation of these coefficients and preliminary results from the implementation of the spherical harmonics equations within Sandia National Laboratories' particle transport code SCEPTRE were presented.

7.1 Conclusions

The verification that the necessary coefficients were derived correctly was performed by first comparing the coefficients computed by the numerical routines to be implemented in SCEPTRE with those computed by directly evaluating the integrals in the software package *Mathematica*. Once it was established that these values were in agreement, a comparison was conducted between the values computed by the subroutines and those computed by an angular quadrature integration technique already present in SCEPTRE for low-order spherical harmonics calculations. These calculations indicated

agreement up to a certain order of spherical harmonics, and rapid divergence after this. When viewed in light of the strong agreement from the first set of comparisons, this indicates that the quadrature integration is only valid up to a relatively low order of spherical harmonics. Thus, the necessity of developing the explicit formulas for the integral values was highlighted; without them, it would be inaccurate for SCEPTRE to perform high-order spherical harmonics calculations.

Once it was established that the coefficients were derived correctly, they were implemented within the SCEPTRE code, and a sequence of two test problems were run. Both test problems were run in two-dimensional Cartesian geometry. The first test problem was performed using the Method of Manufactured Solutions on two separate finite element meshes. Once the solutions showed convergence in angle, the convergence rates with increasing spatial refinement were computed. It was found that these convergence rates were in general agreement with the theoretically expected values.

The second test problem involved the computation of the average scalar flux within four equally-sized subregions of a heterogeneous-material compostion square problem domain. The calculations were carried out on a total of 5 spatial meshes and 7 angular approximation orders. The flux solutions showed convergence with both increasing spatial and angular refinement. A reference solution within each region was estimated using Richardson extrapolation on the highest-order angular approximation, and the resulting convergence rates of the average flux with increasing spatial refinement was calculated. These calculations indicated a convergence rate that was one order less than that expected. This could be due to possible discontinuities in the first derivative of the flux, errors resulting from the region-averaging procedure, or the lack of accurate

convergence rate estimations on the unstructured finite elements routines used by SCEPTRE. It is noted that similar disagreement between predicted and observed convergence rates has been shown with discrete ordinates calculations using SCEPTRE, which leads to the conjecture that the disagreement lies in the spatial rather than the angular discretization.

7.2 Recommendations for Future Work

Now that it has been shown that the spherical harmonics routines have been implemented correctly in SCEPTRE for one-group two-dimensional Cartesian configurations, it is natural to exercise this capability on three-dimensional problems, as the required calculations are already in place. It would also be interesting to experiment with multigroup applications. In addition, comparison studies between the spherical harmonics and discrete ordinates methods for angular discretization would be useful to ascertain the relative efficiencies of the two methods within the SCEPTRE framework.

A further investigation into the discrepancies between the expected and observed convergence rates is also warranted, especially if similar behavior is encountered in both angular discretization methods.

Bibliography

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

[2] W. J. Bohnhoff, C. R. Drumm, W. C. Fan, L. J. Lorence, S. Pautz, and J. L. Powell, "Coupled Electron-Photon Transport with the CEPTRE Code," American Nuclear Society Topical Meeting in Mathematics and Computations, Avignon, France (2005)

[3] C. R. E. de Oliveira, "An Arbitrary Geometry Finite Element Method for Multigroup Neutron Transport with Anisotropic Scattering," *Progress in Nuclear Energy*, **18,** 227 (1986).

[4] G. C. Pomraning and M. Clark, Jr., "The Variational Method Applied to the Monoenergetic Boltzmann Transport Equation, Part II," *Nuclear Science and Engineering*, **16**, 155 (1963).

[5] R. T. Ackroyd, "Least Squares Derivation of Extremum and Weighted-Residual Methods for Equations of Reactor Physics-I," *Annals of Nuclear Energy*, **10**, 65 (1983).

[6] J. E. Morel and J. M. McGhee, "A Self-Adjoint Angular Flux Equation," *Nuclear Science and Engineering,* **132**, 312 (1999).

[7] J. M. McGhee, R. M. Roberts and J. E. Morel, "Dante Transport Solver: An Unstructured Mesh, 3-D Spherical Harmonics Algorithm/Compatible with Parallel

Computer Architecture," Los Alamos Technical Report LA-UR-97-1031, Los Alamos National Laboratory, Los Alamos NM (1996).

[8] J. E. Morel, B. T. Adams, T. Noh, J. M. McGhee, T. M. Evans, and T. J. Urbatsch, "Spatial Discretizations for Self-Adjoint Forms of the Radiative Transfer Equations," *Journal of Computational Physics,* **214**, 12 (2006).

[9] W. C. Fan, C. R. Drumm, and J. L. Powell, "Discrete Ordinates Approximations to First- and Second-Order Forms of the Transport Equation," Sandia Technical Report SAND2002-1880, Sandia National Laboratories/New Mexico, Albuquerque, NM (2002)

[10] S. Chandrasekhar, *Radiative Transfer*, Dover Publications, Inc., Mineola, New York (1960).

[11] E. M. Gelbard, "Spherical Harmonics Methods: P_L and Double- P_L Approximations," *Computing Methods in Reactor Physics*, Gordon & Breach Scientific Publications, New York, pp. 269-357 (1968).

[12] B. Davison, *Neutron Transport Theory*, Clarendon Press, Oxford (1957).

[13] W. C. Fan, "Discrete Ordinates Equations Equivalent to Spherical Harmonics Equations in X-Y-Z Geometry," unpublished memo (2005).

[14] J. J. Duderstadt and W. R. Martin, *Transport Theory*, Wiley-Interscience, New York (1979).

[15] J. E. Morel, "Fokker-Planck Calculations Using Standard Discrete Ordinates Transport Codes," *Nuclear Science and Engineering,* **79**, 340 (1981).

[16] L. J. Lorence, J. E. Morel, and G. D. Valdez, "Physics Guide to CEPXS: A Multigroup Coupled Electron-Photon Cross Section Generating Code (Version

1.0)," Sandia Technical Report SAND89-1685, Sandia National Laboratories, Albuquerque, New Mexico (1989).

[17] C. R. Drumm, "Multidimensional Electron-Photon Transport with Standard Discrete Ordinates Codes," *Nuclear Science and Engineering*, **127**, 1 (1997).

[18] C. R. Drumm, W. C. Fan, L. Lorence, and J. Liscum-Powell, "An Analysis of the Extended Transport Correction with Application to Electron Beam Transport," *Nuclear Science and Engineering*, **155**, 355 (2006).

[19] J. A. Josef and J. E. Morel, "Simplified spherical harmonic method for coupled electron-photon transport calculations," *Physical Review E*, **57,** 6161 (1998).

[20] C. R. Drumm, "Parallel Finite Element Electron-Photon Transport Analysis on 2-D Unstructured Mesh," Sandia Technical Report SAND98-0098, Sandia National Laboratories, Albuquerque, New Mexico (1999).

[21] J. L. Liscum-Powell, A. K. Prinja, J. E. Morel, and L. J. Lorence, "Self-Adjoint Angular Flux Equation for Coupled Electron-Photon Transport," *Transactions of the American Nuclear Society,* **81**, 129 (1999).

[22] J. L. Liscum-Powell, A. K. Prinja, J. E. Morel, and L. J. Lorence, "Finite Element Solution of the Self-Adjoint Angular Flux Equation for Coupled Electron-Photon Transport," *Nuclear Science and Engineering,* **142**, 270 (2002).

[23] J. N. Reddy, *An Introduction to the Finite Element Method* (Second Edition), McGraw-Hill, New York, New York (1993).

[24] C. R. Drumm, "Manufactured Solution Verification of the CEPTRE Code," unpublished memo (2006).

[25] C. R. Drumm, personal communication (2008)

[26] Y. Y. Azmy, "The Weighted Diamond-Difference Form of Nodal Transport Methods," *Nuclear Science and Engineering*, **98**, 29 (1988).

[27] M. T. Heath, *Scientific Computing: An Introductory Survey* (Second Edition), McGraw-Hill, New York, New York (2002).

[28] J. I. Duo and Y. Y. Azmy, "Error Comparison of Diamond Difference, Nodal, and Characteristics Methods for Solving Multidimensional Transport Problems with Discrete Ordinates Approximation," *Nuclear Science and Engineering*, **156**, 139 (2007).

Appendix A

List of Moment Integrals

A.1 $\int_{4\pi} Y^{\alpha}_{l',m'}(\hat{\Omega})\mu_i Y^{\beta}_{l,m}(\hat{\Omega})d\hat{\Omega}$

Table **A.1**: $\int_{4\pi} Y^{c}_{l',m'}(\hat{\Omega})\mu_x Y^{c}_{l,m}(\hat{\Omega})d\hat{\Omega}$

l'	m'	Value
l-1	m-1	$-\frac{1}{2}\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m-1}}}\sqrt{\frac{(l+m)(l+m-1)}{(2l-1)(2l+1)}}$
	m+1	$\frac{1}{2}\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m+1}}}\sqrt{\frac{(l-m)(l-m-1)}{(2l-1)(2l+1)}}$
l+1	m-1	$\frac{1}{2}\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m-1}}}\sqrt{\frac{(l-m+1)(l-m+2)}{(2l+1)(2l+3)}}$
	m+1	$-\frac{1}{2}\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m+1}}}\sqrt{\frac{(l+m+1)(l+m+2)}{(2l+1)(2l+3)}}$

Table **A.2**: $\int_{4\pi} Y^{s}_{l',m'}(\hat{\Omega})\mu_y Y^{c}_{l,m}(\hat{\Omega})d\hat{\Omega}$

l'	***m'***	**Value**	**Exceptions**
l-1	m-1	$\frac{1}{2}\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m-1}}}\sqrt{\frac{(l+m)(l+m-1)}{(2l-1)(2l+1)}}$	If $m' = 0$, then the value of the moment integral is 0
	m+1	$\frac{1}{2}\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m+1}}}\sqrt{\frac{(l-m)(l-m-1)}{(2l-1)(2l+1)}}$	
l+1	m-1	$-\frac{1}{2}\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m-1}}}\sqrt{\frac{(l-m+2)(l-m+1)}{(2l+1)(2l+3)}}$	
	m+1	$-\frac{1}{2}\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m+1}}}\sqrt{\frac{(l+m+1)(l+m+2)}{(2l+1)(2l+3)}}$	

Table **A.3**: $\int_{4\pi} Y^{c}_{l',m'}(\hat{\Omega})\mu_z Y^{c}_{l,m}(\hat{\Omega})d\hat{\Omega}$

l'	***m'***	**Value**
l-1	m	$\sqrt{\frac{(l-m)(l+m)}{(2l-1)(2l+1)}}$
l+1	m	$\sqrt{\frac{(l-m+1)(l+m+1)}{(2l+1)(2l+3)}}$

Table **A.4**: $\int_{4\pi} Y^{s}_{l',m'}(\hat{\Omega})\mu_x Y^{s}_{l,m}(\hat{\Omega})d\hat{\Omega}$

l'	m'	Value	Exceptions
l-1	m-1	$-\frac{1}{2}\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m-1}}}\sqrt{\frac{(l+m)(l+m-1)}{(2l-1)(2l+1)}}$	If $m=0$ or $m'=0$ then the value of the moment integral is 0
	m+1	$\frac{1}{2}\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m+1}}}\sqrt{\frac{(l-m)(l-m-1)}{(2l-1)(2l+1)}}$	
l+1	m-1	$\frac{1}{2}\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m-1}}}\sqrt{\frac{(l-m+1)(l-m+2)}{(2l+1)(2l+3)}}$	
	m+1	$-\frac{1}{2}\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m+1}}}\sqrt{\frac{(l+m+1)(l+m+2)}{(2l+1)(2l+3)}}$	

Table **A.5**: $\int_{4\pi} Y^{c}_{l',m'}(\hat{\Omega})\mu_y Y^{s}_{l,m}(\hat{\Omega})d\hat{\Omega}$

l'	m'	Value	Exceptions
l-1	m-1	$-\frac{1}{2}\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m-1}}}\sqrt{\frac{(l+m)(l+m-1)}{(2l-1)(2l+1)}}$	If $m=0$ then the value of the moment integral is 0
	m+1	$-\frac{1}{2}\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m+1}}}\sqrt{\frac{(l-m)(l-m-1)}{(2l-1)(2l+1)}}$	
l+1	m-1	$\frac{1}{2}\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m-1}}}\sqrt{\frac{(l-m+2)(l-m+1)}{(2l+1)(2l+3)}}$	
	m+1	$\frac{1}{2}\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m+1}}}\sqrt{\frac{(l+m+1)(l+m+2)}{(2l+1)(2l+3)}}$	

Table **A.6**: $\int_{4\pi} Y^{s}_{l',m'}(\hat{\Omega})\mu_z Y^{s}_{l,m}(\hat{\Omega})d\hat{\Omega}$

l'	***m'***	**Value**	**Exceptions**
l-1	m	$\sqrt{\frac{(l-m)(l+m)}{(2l-1)(2l+1)}}$	If $m=0$ or $m'=0$ then the value of the moment integral is 0
l+1	m	$\sqrt{\frac{(l-m+1)(l+m+1)}{(2l+1)(2l+3)}}$	

A.2 $\int_{4\pi} Y^{\alpha}_{l',m'}(\hat{\Omega})\mu_i\mu_j Y^{\beta}_{l,m}(\hat{\Omega})d\hat{\Omega}$

Table **A.7**: $\int_{4\pi} Y^{c}_{l',m'}(\hat{\Omega})\mu_x^2 Y^{c}_{l,m}(\hat{\Omega})d\hat{\Omega}$

l'	m'	Value	Exceptions
l-2	m-2	$\frac{1}{4}\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m-2}}}\sqrt{\frac{(l+m)(l+m-1)(l+m-2)(l+m-3)}{(2l-3)(2l-1)(2l-1)(2l+1)}}$	
	m	$-\frac{1}{2}\sqrt{\frac{(l-m)(l-m-1)(l+m)(l+m-1)}{(2l-3)(2l-1)(2l-1)(2l+1)}}$	If $m=1$, $-\frac{3}{4}\sqrt{\frac{(l-m)(l-m-1)(l+m)(l+m-1)}{(2l-3)(2l-1)(2l-1)(2l+1)}}$
	m+2	$\frac{1}{4}\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m+2}}}\sqrt{\frac{(l-m)(l-m-1)(l-m-2)(l-m-3)}{(2l-3)(2l-1)(2l-1)(2l+1)}}$	
l	m-2	$-\frac{1}{2}\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m-2}}}\frac{\sqrt{(l-m+1)(l-m+2)(l+m)(l+m-1)}}{(2l-1)(2l+3)}$	
	m	$\frac{l^2+l-1+m^2}{(2l-1)(2l+3)}$	If $m=1$, $\frac{3}{2}\frac{l^2+l-1+m^2}{(2l-1)(2l+3)}$
	m+2	$-\frac{1}{2}\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m+2}}}\frac{\sqrt{(l+m+1)(l+m+2)(l-m)(l-m-1)}}{(2l-1)(2l+3)}$	

l+2	m-2	$\frac{1}{4}\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m-2}}}\sqrt{\frac{(l-m+1)(l-m+2)(l-m+3)(l-m+4)}{(2l+1)(2l+3)(2l+3)(2l+5)}}$	
	m	$-\frac{1}{2}\sqrt{\frac{(l-m+1)(l-m+2)(l+m+1)(l+m+2)}{(2l+1)(2l+3)(2l+3)(2l+5)}}$	If $m=1$, $-\frac{3}{4}\sqrt{\frac{(l-m+1)(l-m+2)(l+m+1)(l+m+2)}{(2l+1)(2l+3)(2l+3)(2l+5)}}$
	m+2	$\frac{1}{4}\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m+2}}}\sqrt{\frac{(l+m+1)(l+m+2)(l+m+3)(l+m+4)}{(2l+1)(2l+3)(2l+3)(2l+5)}}$	

Table **A.8**: $\int_{4\pi} Y^s_{l',m'}(\hat{\Omega})\mu_x\mu_y Y^c_{l,m}(\hat{\Omega})d\hat{\Omega}$; if $m'=0$, integral evaluates to 0

l'	m'	**Value**	**Exceptions**
l-2	m-2	$\frac{1}{4}\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m-2}}}\sqrt{\frac{(l+m)(l+m-1)(l+m-2)(l+m-3)}{(2l-3)(2l-1)(2l-1)(2l+1)}}$	
	m	0	If $m=1$, $-\frac{1}{4}\sqrt{\frac{(l-m)(l-m-1)(l+m)(l+m-1)}{(2l-3)(2l-1)(2l-1)(2l+1)}}$
	m+2	$-\frac{1}{4}\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m+2}}}\sqrt{\frac{(l-m)(l-m-1)(l-m-2)(l-m-3)}{(2l-3)(2l-1)(2l-1)(2l+1)}}$	
	m-2	$-\frac{1}{2}\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m-2}}}\frac{\sqrt{(l-m+2)(l-m+1)(l+m)(l+m-1)}}{(2l-1)(2l+3)}$	

	m	0	If $m=1$, $\frac{1}{2}\frac{l^2+l-1+m^2}{(2l-1)(2l+3)}$
	m+2	$\frac{1}{2}\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m+2}}}\frac{\sqrt{(l-m)(l-m-1)(l+m+1)(l+m+2)}}{(2l-1)(2l+3)}$	
l+2	m-2	$\frac{1}{4}\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m-2}}}\sqrt{\frac{(l-m+1)(l-m+2)(l-m+3)(l-m+4)}{(2l+1)(2l+3)(2l+3)(2l+5)}}$	
	m	0	If $m=1$, $-\frac{1}{4}\sqrt{\frac{(l+m+1)(l+m+2)(l-m+1)(l-m+2)}{(2l+1)(2l+3)(2l+3)(2l+5)}}$
	m+2	$-\frac{1}{4}\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m+2}}}\sqrt{\frac{(l+m+1)(l+m+2)(l+m+3)(l+m+4)}{(2l+1)(2l+3)(2l+3)(2l+5)}}$	

Table **A.9**: $\int_{4\pi} Y^c_{l',m'}(\hat{\Omega})\mu_x\mu_z Y^c_{l,m}(\hat{\Omega})d\hat{\Omega}$

l'	m'	Value
l-2	m-1	$-\frac{1}{2}\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m-1}}}\sqrt{\frac{(l-m)(l+m)(l+m-1)(l+m-2)}{(2l-3)(2l-1)(2l-1)(2l+1)}}$
	m+1	$\frac{1}{2}\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m+1}}}\sqrt{\frac{(l-m)(l-m-1)(l-m-2)(l+m)}{(2l-3)(2l-1)(2l-1)(2l+1)}}$

l	m-1	$-\frac{1}{2}\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m-1}}}\frac{(l+m-1)\sqrt{(l+m)(l-m+1)}}{(2l-1)(2l+1)}$ $+\frac{1}{2}\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m-1}}}\frac{(l-m+2)\sqrt{(l+m)(l-m+1)}}{(2l+1)(2l+3)}$
	m+1	$\frac{1}{2}\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m+1}}}\frac{(l-m-1)\sqrt{(l-m)(l+m+1)}}{(2l-1)(2l+1)}$ $-\frac{1}{2}\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m+1}}}\frac{(l+m+2)\sqrt{(l-m)(l+m+1)}}{(2l+1)(2l+3)}$
l+2	m-1	$\frac{1}{2}\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m-1}}}\sqrt{\frac{(l-m+1)(l-m+2)(l-m+3)(l+m+1)}{(2l+1)(2l+3)(2l+3)(2l+5)}}$
	m+1	$-\frac{1}{2}\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m+1}}}\sqrt{\frac{(l-m+1)(l+m+1)(l+m+2)(l+m+3)}{(2l+1)(2l+3)(2l+3)(2l+5)}}$

Table **A.10**: $\int_{4\pi} Y^c_{l',m'}(\hat{\Omega})\mu_y^2 Y^c_{l,m}(\hat{\Omega})d\hat{\Omega}$

l′	***m′***	**Value**	**Exceptions**
	m-2	$-\frac{1}{4}\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m-2}}}\sqrt{\frac{(l+m)(l+m-1)(l+m-2)(l+m-3)}{(2l-3)(2l-1)(2l-1)(2l+1)}}$	

	m	$-\frac{1}{2}\sqrt{\frac{(l-m)(l-m-1)(l+m)(l+m-1)}{(2l-3)(2l-1)(2l-1)(2l+1)}}$	If $m=1$, $-\frac{1}{4}\sqrt{\frac{(l-m)(l-m-1)(l+m)(l+m-1)}{(2l-3)(2l-1)(2l-1)(2l+1)}}$
	m+2	$-\frac{1}{4}\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m+2}}}\sqrt{\frac{(l-m)(l-m-1)(l-m-2)(l-m-3)}{(2l-3)(2l-1)(2l-1)(2l+1)}}$	
l	m-2	$\frac{1}{2}\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m-2}}}\frac{\sqrt{(l+m)(l+m-1)(l-m+2)(l-m+1)}}{(2l-1)(2l+3)}$	
	m	$\frac{l^2+l-1+m^2}{(2l-1)(2l+3)}$	If $m=1$, $\frac{1}{2}\frac{l^2+l-1+m^2}{(2l-1)(2l+3)}$
	m+2	$\frac{1}{2}\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m+2}}}\frac{\sqrt{(l-m)(l-m-1)(l+m+1)(l+m+2)}}{(2l-1)(2l+3)}$	
l+2	m-2	$-\frac{1}{4}\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m-2}}}\sqrt{\frac{(l-m+4)(l-m+3)(l-m+2)(l-m+1)}{(2l+1)(2l+3)(2l+3)(2l+5)}}$	
	m	$-\frac{1}{2}\sqrt{\frac{(l+m+1)(l+m+2)(l-m+2)(l-m+1)}{(2l+1)(2l+3)(2l+3)(2l+5)}}$	If $m=1$, $-\frac{1}{4}\sqrt{\frac{(l+m+1)(l+m+2)(l-m+2)(l-m+1)}{(2l+1)(2l+3)(2l+3)(2l+5)}}$
	m+2	$-\frac{1}{4}\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m+2}}}\sqrt{\frac{(l+m+1)(l+m+2)(l+m+3)(l+m+4)}{(2l+1)(2l+3)(2l+3)(2l+5)}}$	

Table **A.11**: $\int_{4\pi} Y^{s}_{l',m'}(\hat{\Omega})\mu_y\mu_z Y^{c}_{l,m}(\hat{\Omega})d\hat{\Omega}$; if $m' = 0$, integral evaluates to 0

l'	m'	Value
l-2	m-1	$-\frac{1}{2}\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m-1}}}\sqrt{\frac{(l-m)(l+m)(l+m-1)(l+m-2)}{(2l-3)(2l-1)(2l-1)(2l+1)}}$
	m+1	$-\frac{1}{2}\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m+1}}}\sqrt{\frac{(l-m)(l-m-1)(l-m-2)(l+m)}{(2l-3)(2l-1)(2l-1)(2l+1)}}$
l	m-1	$-\frac{1}{2}\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m-1}}}\frac{(l+m-1)\sqrt{(l+m)(l-m+1)}}{(2l-1)(2l+1)}$ $+\frac{1}{2}\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m-1}}}\frac{(l-m+2)\sqrt{(l+m)(l-m+1)}}{(2l+1)(2l+3)}$
	m+1	$-\frac{1}{2}\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m+1}}}\frac{(l-m-1)\sqrt{(l-m)(l+m+1)}}{(2l-1)(2l+1)}$ $+\frac{1}{2}\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m+1}}}\frac{(l+m+2)\sqrt{(l-m)(l+m+1)}}{(2l+1)(2l+3)}$
l+2	m-1	$\frac{1}{2}\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m-1}}}\sqrt{\frac{(l-m+3)(l-m+2)(l-m+1)(l+m+1)}{(2l+1)(2l+3)(2l+3)(2l+5)}}$
	m+1	$\frac{1}{2}\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m+1}}}\sqrt{\frac{(l-m+1)(l+m+1)(l+m+2)(l+m+3)}{(2l+1)(2l+3)(2l+3)(2l+5)}}$

Table **A.12**: $\int_{4\pi} Y^c_{l',m'}(\hat{\Omega})\mu_z^2 Y^c_{l,m}(\hat{\Omega})d\hat{\Omega}$

l′	***m′***	**Value**
l-2	m	$\sqrt{\frac{(l-m-1)(l-m)(l+m-1)(l+m)}{(2l-3)(2l-1)(2l-1)(2l+1)}}$
l	m	$\frac{2l^2+2l-1-2m^2}{(2l-1)(2l+3)}$
l+2	m	$\sqrt{\frac{(l-m+2)(l-m+1)(l+m+1)(l+m+2)}{(2l+1)(2l+3)(2l+3)(2l+5)}}$

Table **A.13**: $\int_{4\pi} Y^s_{l',m'}(\hat{\Omega})\mu_x^2 Y^s_{l,m}(\hat{\Omega})d\hat{\Omega}$; if $m=0$ or $m'=0$, integral evaluates to 0

l′	***m′***	**Value**	**Exceptions**
	m-2	$\frac{1}{4}\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m-2}}}\sqrt{\frac{(l+m)(l+m-1)(l+m-2)(l+m-3)}{(2l-3)(2l-1)(2l-1)(2l+1)}}$	

	m	$-\frac{1}{2}\sqrt{\frac{(l-m)(l-m-1)(l+m)(l+m-1)}{(2l-3)(2l-1)(2l-1)(2l+1)}}$	If $m=1$, $-\frac{1}{4}\sqrt{\frac{(l-m)(l-m-1)(l+m)(l+m-1)}{(2l-3)(2l-1)(2l-1)(2l+1)}}$
	m+2	$\frac{1}{4}\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m+2}}}\sqrt{\frac{(l-m)(l-m-1)(l-m-2)(l-m-3)}{(2l-3)(2l-1)(2l-1)(2l+1)}}$	
l	m-2	$-\frac{1}{2}\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m-2}}}\frac{\sqrt{(l-m+1)(l-m+2)(l+m)(l+m-1)}}{(2l-1)(2l+3)}$	
	m	$\frac{l^2+l-1+m^2}{(2l-1)(2l+3)}$	If $m=1$, $\frac{1}{2}\frac{l^2+l-1+m^2}{(2l-1)(2l+3)}$
	m+2	$-\frac{1}{2}\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m+2}}}\frac{\sqrt{(l-m)(l-m-1)(l+m+1)(l+m+2)}}{(2l-1)(2l+3)}$	
l+2	m-2	$\frac{1}{4}\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m-2}}}\sqrt{\frac{(l-m+1)(l-m+2)(l-m+3)(l-m+4)}{(2l+1)(2l+3)(2l+3)(2l+5)}}$	
	m	$-\frac{1}{2}\sqrt{\frac{(l+m+1)(l+m+2)(l-m+1)(l-m+2)}{(2l+1)(2l+3)(2l+3)(2l+5)}}$	If $m=1$, $-\frac{1}{4}\sqrt{\frac{(l+m+1)(l+m+2)(l-m+1)(l-m+2)}{(2l+1)(2l+3)(2l+3)(2l+5)}}$
	m+2	$\frac{1}{4}\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m+2}}}\sqrt{\frac{(l+m+1)(l+m+2)(l+m+3)(l+m+4)}{(2l+1)(2l+3)(2l+3)(2l+5)}}$	

Table **A.14**: $\int_{4\pi} Y^{c}_{l',m'}(\hat{\Omega})\mu_x\mu_y Y^{s}_{l,m}(\hat{\Omega})d\hat{\Omega}$; if $m=0,$ integral evaluates to 0

l′	***m′***	**Value**	**Exceptions**
l-2	*m*-2	$-\frac{1}{4}\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m-2}}}\sqrt{\frac{(l+m)(l+m-1)(l+m-2)(l+m-3)}{(2l-3)(2l-1)(2l-1)(2l+1)}}$	
	m	0	If $m=1,\ -\frac{1}{4}\sqrt{\frac{(l-m)(l-m-1)(l+m)(l+m-1)}{(2l-3)(2l-1)(2l-1)(2l+1)}}$
	m+2	$\frac{1}{4}\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m+2}}}\sqrt{\frac{(l-m)(l-m-1)(l-m-2)(l-m-3)}{(2l-3)(2l-1)(2l-1)(2l+1)}}$	
l	*m*-2	$\frac{1}{2}\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m-2}}}\frac{\sqrt{(l+m)(l+m-1)(l-m+1)(l-m+2)}}{(2l-1)(2l+3)}$	
	m	0	If $m=1,\ \frac{1}{2}\frac{l^2+l-1+m^2}{(2l-1)(2l+3)}$
	m+2	$-\frac{1}{2}\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m+2}}}\frac{\sqrt{(l-m)(l-m-1)(l+m+1)(l+m+2)}}{(2l-1)(2l+3)}$	
	m-2	$-\frac{1}{4}\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m-2}}}\sqrt{\frac{(l-m+1)(l-m+2)(l-m+3)(l-m+4)}{(2l+1)(2l+3)(2l+3)(2l+5)}}$	
	m	0	If $m=1,\ -\frac{1}{4}\sqrt{\frac{(l+m+1)(l+m+2)(l-m+1)(l-m+2)}{(2l+1)(2l+3)(2l+3)(2l+5)}}$

	m+2	$\frac{1}{4}\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m+2}}}\sqrt{\frac{(l+m+1)(l+m+2)(l+m+3)(l+m+4)}{(2l+1)(2l+3)(2l+3)(2l+5)}}$	

Table **A.15**: $\int_{4\pi} Y^s_{l',m'}(\hat{\Omega})\mu_x\mu_z Y^s_{l,m}(\hat{\Omega})d\hat{\Omega}$; if $m=0$ or $m'=0$, integral evaluates to 0

l'	m'	Value
l-2	m-1	$-\frac{1}{2}\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m-1}}}\sqrt{\frac{(l-m)(l+m)(l+m-1)(l+m-2)}{(2l-3)(2l-1)(2l-1)(2l+1)}}$
	m+1	$\frac{1}{2}\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m+1}}}\sqrt{\frac{(l-m)(l-m-1)(l-m-2)(l+m)}{(2l-1)(2l+1)(2l-3)(2l-1)}}$
l	m-1	$-\frac{1}{2}\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m-1}}}\frac{(l+m-1)\sqrt{(l+m)(l-m+1)}}{(2l-1)(2l+1)}$ $+\frac{1}{2}\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m-1}}}\frac{(l-m+2)\sqrt{(l+m)(l-m+1)}}{(2l+1)(2l+3)}$
	m+1	$\frac{1}{2}\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m+1}}}\frac{(l-m-1)\sqrt{(l-m)(l+m+1)}}{(2l-1)(2l+1)}$ $-\frac{1}{2}\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m+1}}}\frac{(l+m+2)\sqrt{(l-m)(l+m+1)}}{(2l+1)(2l+3)}$

l+2	m-1	$\frac{1}{2}\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m-1}}}\sqrt{\frac{(l-m+1)(l-m+2)(l-m+3)(l+m+1)}{(2l+1)(2l+3)(2l+3)(2l+5)}}$
	m+1	$-\frac{1}{2}\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m+1}}}\sqrt{\frac{(l-m+1)(l+m+1)(l+m+2)(l+m+3)}{(2l+1)(2l+3)(2l+3)(2l+5)}}$

Table **A.16**: $\int_{4\pi} Y^s_{l',m'}(\hat{\Omega})\mu_y^2 Y^s_{l,m}(\hat{\Omega})d\hat{\Omega}$; if $m=0$ or $m'=0$, integral evaluates to 0

l′	***m′***	**Value**	**Exceptions**
l-2	m-2	$-\frac{1}{4}\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m-2}}}\sqrt{\frac{(l+m)(l+m-1)(l+m-2)(l+m-3)}{(2l-3)(2l-1)(2l-1)(2l+1)}}$	
	m	$-\frac{1}{2}\sqrt{\frac{(l-m)(l-m-1)(l+m)(l+m-1)}{(2l-3)(2l-1)(2l-1)(2l+1)}}$	If $m=1$, $-\frac{3}{4}\sqrt{\frac{(l-m)(l-m-1)(l+m)(l+m-1)}{(2l-3)(2l-1)(2l-1)(2l+1)}}$
	m+2	$-\frac{1}{4}\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m+2}}}\sqrt{\frac{(l-m)(l-m-1)(l-m-2)(l-m-3)}{(2l-3)(2l-1)(2l-1)(2l+1)}}$	
	m-2	$\frac{1}{2}\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m-2}}}\frac{\sqrt{(l+m)(l+m-1)(l-m+2)(l-m+1)}}{(2l-1)(2l+3)}$	
	m	$\frac{l^2+l-1+m^2}{(2l-1)(2l+3)}$	If $m=1$, $\frac{3}{2}\frac{l^2+l-1+m^2}{(2l-1)(2l+3)}$

	m+2	$\frac{1}{2}\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m+2}}}\frac{\sqrt{(l-m)(l-m-1)(l+m+1)(l+m+2)}}{(2l-1)(2l+3)}$	
l+2	m-2	$-\frac{1}{4}\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m-2}}}\sqrt{\frac{(l-m+2)(l-m+1)(l-m+4)(l-m+3)}{(2l+1)(2l+3)(2l+3)(2l+5)}}$	
	m	$-\frac{1}{2}\sqrt{\frac{(l+m+1)(l+m+2)(l-m+2)(l-m+1)}{(2l+1)(2l+3)(2l+3)(2l+5)}}$	If $m=1$, $-\frac{3}{4}\sqrt{\frac{(l+m+1)(l+m+2)(l-m+2)(l-m+1)}{(2l+1)(2l+3)(2l+3)(2l+5)}}$
	m+2	$-\frac{1}{4}\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m+2}}}\sqrt{\frac{(l+m+1)(l+m+2)(l+m+3)(l+m+4)}{(2l+1)(2l+3)(2l+3)(2l+5)}}$	

Table **A.17**: $\int_{4\pi} Y^c_{l',m'}(\hat{\Omega})\mu_y\mu_z Y^s_{l,m}(\hat{\Omega})d\hat{\Omega}$; if $m=0,$ integral evaluates to 0

l′	***m′***	**Value**
l-2	m-1	$\frac{1}{2}\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m-1}}}\sqrt{\frac{(l-m)(l+m)(l+m-1)(l+m-2)}{(2l-3)(2l-1)(2l-1)(2l+1)}}$
	m+1	$\frac{1}{2}\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m+1}}}\sqrt{\frac{(l-m)(l-m-1)(l-m-2)(l+m)}{(2l-3)(2l-1)(2l-1)(2l+1)}}$

l	m-1	$\frac{1}{2}\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m-1}}}\frac{(l+m-1)\sqrt{(l+m)(l-m+1)}}{(2l-1)(2l+1)}$ $-\frac{1}{2}\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m-1}}}\frac{(l-m+2)\sqrt{(l+m)(l-m+1)}}{(2l+1)(2l+3)}$
	m+1	$\frac{1}{2}\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m+1}}}\frac{(l-m-1)\sqrt{(l-m)(l+m+1)}}{(2l-1)(2l+1)}$ $-\frac{1}{2}\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m+1}}}\frac{(l+m+2)\sqrt{(l-m)(l+m+1)}}{(2l+1)(2l+3)}$
l+2	m-1	$-\frac{1}{2}\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m-1}}}\sqrt{\frac{(l-m+3)(l-m+2)(l-m+1)(l+m+1)}{(2l+1)(2l+3)(2l+3)(2l+5)}}$
	m+1	$-\frac{1}{2}\frac{\sqrt{\varepsilon_m}}{\sqrt{\varepsilon_{m+1}}}\sqrt{\frac{(l-m+1)(l+m+1)(l+m+2)(l+m+3)}{(2l+1)(2l+3)(2l+3)(2l+5)}}$

Table **A.18**: $\int_{4\pi} Y^s_{l',m'}(\hat{\Omega})\mu_z^2 Y^s_{l,m}(\hat{\Omega})d\hat{\Omega}$ **;** if $m = 0$ or $m' = 0$, integral evaluates to 0

l'	m'	Value
l-2	m	$\sqrt{\frac{(l-m-1)(l-m)(l+m)(l+m-1)}{(2l-3)(2l-1)(2l-1)(2l+1)}}$
l	m	$\frac{2l^2+2l-1-2m^2}{(2l-1)(2l+3)}$
l+2	m	$\sqrt{\frac{(l-m+1)(l-m+2)(l+m+2)(l+m+1)}{(2l+1)(2l+3)(2l+3)(2l+5)}}$

Appendix B

Subroutines Implemented into SCEPTRE

B.1 First Order Specification File

```
//*************************************************************************
//SPECIFICATION FILE (fOSM.hh)
//This file gives the specification for the fOSM class (first order streaming
//moment).
//Author:  Andrew Bielen
//Date:    5/30/2007
//Modified to include phase 6/6/2007
//*************************************************************************

#ifndef FOSM_HH
#define FOSM_HH

class fOSM
{
public:
  fOSM();                                  //constructor
  void Set(int, int, int, int, int, int, int);  //Set dimension and indices
  double GetMom();                         //return moment
  ~fOSM();                                 //destructor
private:
  int dim, l1, l2, m1, m2, phase1, phase2;
};

#include "fOSM.cc"
```

```
#endif  // FOSM_HH
```

B.2 First Order Implementation File

```
//*********************************************************************
//IMPLEMENTATION FILE (fOSM.cc)
//This file implements the functions of the fOSM class
//Author:  Andrew Bielen
//Date:    May 30, 2007
//Modified to include phase: June 6, 2007
//Compatible with SCEPTRE tests: July 18, 2007
//*********************************************************************
#include "fOSM.hh"

fOSM::fOSM()
{
  dim = l1 = l2 = m1 = m2 = phase1 = phase2 = 0;
}
void fOSM::Set(int dim_, int l1_, int l2_, int m1_, int m2_, int phase1_, int phase2_)
{
  dim = dim_;
  l1  = l1_;
  l2  = l2_;
  m1  = m1_;
  m2  = m2_;
  phase1 = phase1_;  //phase = 0 for cosine, 1 for sine
  phase2 = phase2_;
}
double fOSM::GetMom()
{
  double r, lr, mr, eps = 1.;
  lr = double(l1);
  mr = double(m1);
```

```
r = 0.0;
if (phase1 == 0)
{
  if (phase2 == 0)
  {
    if (dim == 0)
    {
      if (l2 == l1 - 1)
    {
        if (m2 == m1 - 1)
      {
          if (m1 == 0)
            eps = 1. / sqrt(2.);
          else if (m1 == 1)
            eps = sqrt(2.);
          r = -sqrt((lr+mr-1)*(lr+mr)/((2*lr-1)*(2*lr+1)))/2;
        }
        else if (m2 == m1 + 1)
      {
          if (m1 == 0)
            eps = sqrt(2.);
          r = sqrt((lr-mr-1)*(lr-mr)/((2*lr-1)*(2*lr+1)))/2;
        }
      }
      else if (l2 == l1 + 1)
    {
        if (m2 == m1 - 1)
      {
          if (m1 == 0)
            eps = 1. / sqrt(2.);
          else if (m1 == 1)
            eps = sqrt(2.);
          r = sqrt((lr-mr+1)*(lr-mr+2)/((2*lr+1)*(2*lr+3)))/2;
        }
        else if (m2 == m1 + 1)
      {
          if (m1 == 0)
            eps = sqrt(2.);
```

```
        r = -sqrt((lr+mr+1)*(lr+mr+2)/((2*lr+1)*(2*lr+3)))/2;
      }
    }
  }
  else if (dim == 2)
  {
    if (l2 == l1 - 1)
    {
      if (m2 == m1)
      {
        r = sqrt((lr-mr)*(lr+mr)/((2*lr-1)*(2*lr+1)));
      }
    }
    else if (l2 == l1 + 1)
    {
      if (m2 == m1)
      {
        r = sqrt((lr-mr+1)*(lr+mr+1)/((2*lr+1)*(2*lr+3)));
      }
    }
  }
}
else if (phase2 == 1)
{
  if (dim == 1)
  {
   if (m2 != 0)
   {
    if (l2 == l1 - 1)
  {
      if (m2 == m1 - 1)
    {
        if (m1 == 0)
          eps = 1. / sqrt(2.);
        else if (m1 == 1)
          eps = sqrt(2.);
        r = sqrt((lr+mr-1)*(lr+mr)/((2*lr-1)*(2*lr+1)))/2;
      }
```

```
        else if (m2 == m1 + 1)
      {
          if (m1 == 0)
            eps = sqrt(2.);
          r = sqrt((lr-mr-1)*(lr-mr)/((2*lr-1)*(2*lr+1)))/2;
        }
      }
      else if (l2 == l1 + 1)
    {
        if (m2 == m1 - 1)
      {
          if (m1 == 0)
            eps = 1. / sqrt(2.);
          else if (m1 == 1)
            eps = sqrt(2.);
          r = -sqrt((lr-mr+1)*(lr-mr+2)/((2*lr+1)*(2*lr+3)))/2;
        }
        else if (m2 == m1 + 1)
      {
          if (m1 == 0)
            eps = sqrt(2.);
          r = -sqrt((lr+mr+1)*(lr+mr+2)/((2*lr+1)*(2*lr+3)))/2;
        }
      }
     }
    }
  }
}
else if (phase1 == 1)
{
 if (m1 != 0)
 {
  if (phase2 == 0)
  {
    if (dim == 1)
    {
      if (l2 == l1 - 1)
    {
```

```
      if (m2 == m1 - 1)
    {
        if (m1 == 0)
          eps = 1. / sqrt(2.);
        else if (m1 == 1)
          eps = sqrt(2.);
        r = -sqrt((lr+mr-1)*(lr+mr)/((2*lr-1)*(2*lr+1)))/2;
      }
      else if (m2 == m1 + 1)
    {
        if (m1 == 0)
          eps = sqrt(2.);
        r = -sqrt((lr-mr-1)*(lr-mr)/((2*lr-1)*(2*lr+1)))/2;
      }
    }
    else if (l2 == l1 + 1)
  {
      if (m2 == m1 - 1)
    {
        if (m1 == 0)
          eps = 1. / sqrt(2.);
        else if (m1 == 1)
          eps = sqrt(2.);
        r = sqrt((lr-mr+1)*(lr-mr+2)/((2*lr+1)*(2*lr+3)))/2;
      }
      else if (m2 == m1 + 1)
    {
        if (m1 == 0)
          eps = sqrt(2.);
        r = sqrt((lr+mr+1)*(lr+mr+2)/((2*lr+1)*(2*lr+3)))/2;
      }
    }
  }
}
else if (phase2 == 1)
{
 if (m2 != 0)
 {
```

```
if (dim == 0)
{
  if (l2 == l1 - 1)
{
    if (m2 == m1 - 1)
  {
      if (m1 == 0)
        eps = 1. / sqrt(2.);
      else if (m1 == 1)
        eps = sqrt(2.);
      r = -sqrt((lr+mr-1)*(lr+mr)/((2*lr-1)*(2*lr+1)))/2;
    }
    else if (m2 == m1 + 1)
  {
      if (m1 == 0)
        eps = sqrt(2.);
      r = sqrt((lr-mr-1)*(lr-mr)/((2*lr-1)*(2*lr+1)))/2;
    }
  }
  else if (l2 == l1 + 1)
{
    if (m2 == m1 - 1)
  {
      if (m1 == 0)
        eps = 1. / sqrt(2.);
      else if (m1 == 1)
        eps = sqrt(2.);
      r = sqrt((lr-mr+1)*(lr-mr+2)/((2*lr+1)*(2*lr+3)))/2;
    }
    else if (m2 == m1 + 1)
  {
      if (m1 == 0)
        eps = sqrt(2.);
      r = -sqrt((lr+mr+1)*(lr+mr+2)/((2*lr+1)*(2*lr+3)))/2;
    }
  }
}
else if (dim == 2)
```

```
      {
        if (l2 == l1 - 1)
      {
          if (m2 == m1)
        {
            r = sqrt((lr-mr)*(lr+mr)/((2*lr-1)*(2*lr+1)));
          }
        }
        else if (l2 == l1 + 1)
      {
          if (m2 == m1)
        {
            r = sqrt((lr-mr+1)*(lr+mr+1)/((2*lr+1)*(2*lr+3)));
          }
        }
      }
     }
    }
   }
  }
  return(eps*r);
}
fOSM::~fOSM()
{
}
```

B.3 Second Order Specification File

```
//*************************************************************************
//SPECIFICATION FILE (sOSM.hh)
//This file gives the specification for the sOSM class (second order streaming
//moment).
//Author:  Andrew Bielen
//Date:    5/31/2007
```

```
//Modified to include phase 6/6/2007
//Modified for SCEPTRE inclusion 7/18/2007
//*************************************************************************

#ifndef SOSM_HH
#define SOSM_HH

class sOSM
{
public:
  sOSM();                              //constructor
  sOSM(int, int, int, int, int, int, int, int);          //constructor
  void Set(int, int, int, int, int, int, int, int);      //Set parameters
  double GetMom();                       //return moment
  ~sOSM();                             //destructor
private:
  int dim1,dim2, l1, l2, m1, m2, phase1, phase2;
};

#include "sOSM.cc"

#endif      // SOSM_HH
```

B.4 Second Order Implementation File

```
//*********************************************************************
//IMPLEMENTATION FILE (sOSM.cc)
//This file implements the functions of the sOSM class
//Author:  Andrew Bielen
//Date:    May 31, 2007
//Modified to include phase 6/6/2007
//Modified to include kronecker delta 6/18/2007
//Modified for SCEPTRE inclusion 7/18/2007
//*********************************************************************
```

```
#include "sOSM.hh"

sOSM::sOSM()
{
  dim1 = dim2 = l1 = l2 = m1 = m2 = phase1 = phase2 = 0;
}
sOSM::sOSM(int dim1_, int dim2_, int l1_, int l2_, int m1_, int m2_, int phase1_, int phase2_)
{
  dim1 = dim1_;
  dim2 = dim2_;
  l1 = l1_;
  l2 = l2_;
  m1 = m1_;
  m2 = m2_;
  phase1 = phase1_;
  phase2 = phase2_;
}
void sOSM::Set(int dim1_, int dim2_, int l1_, int l2_, int m1_, int m2_, int phase1_, int phase2_)
{
  dim1 = dim1_;
  dim2 = dim2_;
  l1  = l1_;
  l2  = l2_;
  m1  = m1_;
  m2  = m2_;
  phase1 = phase1_;  //phase = 0 for cosine, 1 for sine
  phase2 = phase2_;
}
double sOSM::GetMom()
{
   double r, lr, mr, eps = 1.;
   lr = double(l1);
   mr = double(m1);
   r = 0.0;
   if (phase1 == 0)
   {
     if (phase2 == 0)
     {
```

```
if (dim1 == 0)
{
  if (dim2 == 0)
  {
    if (l2 == l1 - 2)
    {
      if (m2 == m1 - 2)
      {
        if (m1 == 0)
          eps = 1. / sqrt(2.);
        else if (m1 == 2)
          eps = sqrt(2.);
        r = eps*sqrt((lr+mr-3)*(lr+mr-2)*(lr+mr-1)*(lr+mr)/((2*lr-3)*(2*lr+1)))/(4*(2*lr-1));
      }
      else if (m2 == m1)
    {
        r = -sqrt((lr+mr-1)*(lr+mr)*(lr-mr-1)*(lr-mr)/((2*lr-3)*(2*lr+1)))/(2*(2*lr-1));
        if (m1 == 1)
          r = 1.5*r;
      }
      else if (m2 == m1 + 2)
    {
        if (m1 == 0)
          eps = sqrt(2.);
        r = eps*sqrt((lr-mr-3)*(lr-mr-2)*(lr-mr-1)*(lr-mr)/((2*lr-3)*(2*lr+1)))/(4*(2*lr-1));
    }
    }
    else if (l2 == l1)
    {
      if (m2 == m1 - 2)
    {
        if (m1 == 0)
          eps = 1. / sqrt(2.);
        else if (m1 == 2)
          eps = sqrt(2.);
        r = -eps*sqrt((lr-mr+1)*(lr-mr+2)*(lr+mr-1)*(lr+mr))/(2*(2*lr-1)*(2*lr+3));
      }
      else if (m2 == m1)
```

```
  {
      r = (pow(lr,2)+lr-1+pow(mr,2))/((2*lr-1)*(2*lr+3));
      if (m1 == 1)
        r = 1.5*r;
    }
    else if (m2 == m1 + 2)
  {
      if (m1 == 0)
        eps = sqrt(2.);
      r = -eps*sqrt((lr-mr-1)*(lr-mr)*(lr+mr+1)*(lr+mr+2))/(2*(2*lr-1)*(2*lr+3));
  }
  }
  else if (l2 == l1 + 2)
  {
    if (m2 == m1 - 2)
    {
      if (m1 == 0)
        eps = 1. / sqrt(2.);
      else if (m1 == 2)
        eps = sqrt(2.);
      r = eps*sqrt((lr-mr+1)*(lr-mr+2)*(lr-mr+3)*(lr-mr+4)/((2*lr+1)*(2*lr+5)))/(4*(2*lr+3));
    }
    else if (m2 == m1)
  {
      r = -sqrt((lr+mr+1)*(lr+mr+2)*(lr-mr+1)*(lr-mr+2)/((2*lr+1)*(2*lr+5)))/(2*(2*lr+3));
      if (m1 == 1)
        r = 1.5*r;
    }
    else if (m2 == m1 + 2)
  {
      if (m1 == 0)
        eps = sqrt(2.);
      r = eps*sqrt((lr+mr+1)*(lr+mr+2)*(lr+mr+3)*(lr+mr+4)/((2*lr+1)*(2*lr+5)))/(4*(2*lr+3));
  }
  }
}
else if (dim2 == 2)
{
```

```
if (l2 == l1 - 2)
{
  if (m2 == m1 - 1)
{
    if (m1 == 0)
      eps = 1. / sqrt(2.);
  else if (m1 == 1)
      eps = sqrt(2.);
    r = -eps*sqrt((lr+mr-2)*(lr+mr-1)*(lr+mr)*(lr-mr)/((2*lr-3)*(2*lr+1)))/(2*(2*lr-1));
  }
  else if (m2 == m1 + 1)
{
    if (m1 == 0)
      eps = sqrt(2.);
    r = eps*sqrt((lr+mr)*(lr-mr-2)*(lr-mr-1)*(lr-mr)/((2*lr-3)*(2*lr+1)))/(2*(2*lr-1));
}
}
else if (l2 == l1)
{
  if (m2 == m1 - 1)
                                        {
    if (m1 == 0)
      eps = 1. / sqrt(2.);
  else if (m1 == 1)
      eps = sqrt(2.);
    r = eps*(-sqrt((lr+mr)*(lr-mr+1))*(lr+mr-1)/(2*(2*lr-1)*(2*lr+1))+sqrt((lr+mr)*
           (lr-mr+1))*(lr-mr+2)/(2*(2*lr+1)*(2*lr+3)));
  }
  else if (m2 == m1 + 1)
{
    if (m1 == 0)
      eps = sqrt(2.);
    r = eps*(sqrt((lr+mr+1)*(lr-mr))*(lr-mr-1)/(2*(2*lr-1)*(2*lr+1))-sqrt((lr+mr+1)*
             (lr-mr))*(lr+mr+2)/(2*(2*lr+1)*(2*lr+3)));
}
}
else if (l2 == l1 + 2)
{
```

```
      if (m2 == m1 - 1)
    {
        if (m1 == 0)
          eps = 1. / sqrt(2.);
      else if (m1 == 1)
          eps = sqrt(2.);
        r = eps*sqrt((lr+mr+1)*(lr-mr+1)*(lr-mr+2)*(lr-mr+3)/((2*lr+1)*(2*lr+5)))/(2*(2*lr+3));
      }
      else if (m2 == m1 + 1)
    {
        if (m1 == 0)
          eps = sqrt(2.);
        r = -eps*sqrt((lr+mr+1)*(lr+mr+2)*(lr+mr+3)*(lr-mr+1)/((2*lr+1)*(2*lr+5)))/(2*(2*lr+3));
    }
    }
  }
}
else if (dim1 == 1)
{
  if (dim2 == 1)
  {
    if (l2 == l1 - 2)
    {
      if (m2 == m1 - 2)
    {
        if (m1 == 0)
          eps = 1. / sqrt(2.);
        else if (m1 == 2)
          eps = sqrt(2.);
        r = -eps*sqrt((lr+mr-3)*(lr+mr-2)*(lr+mr-1)*(lr+mr)/((2*lr-3)*(2*lr+1)))/(4*(2*lr-1));
      }
      else if (m2 == m1)
    {
        r = -sqrt((lr-mr-1)*(lr-mr)*(lr+mr-1)*(lr+mr)/((2*lr-3)*(2*lr+1)))/(2*(2*lr-1));
        if (m1 == 1)
          r = 0.5*r;
      }
      else if (m2 == m1 + 2)
```

```
{
    if (m1 == 0)
      eps = sqrt(2.);
    r = -eps*sqrt((lr-mr)*(lr-mr-1)*(lr-mr-2)*(lr-mr-3)/((2*lr-3)*(2*lr+1)))/(4*(2*lr-1));
}
}
else if (l2 == l1)
{
  if (m2 == m1 - 2)
{
    if (m1 == 0)
      eps = 1. / sqrt(2.);
    else if (m1 == 2)
      eps = sqrt(2.);
    r = eps*sqrt((lr-mr+1)*(lr-mr+2)*(lr+mr-1)*(lr+mr))/(2*(2*lr-1)*(2*lr+3));
  }
  else if (m2 == m1)
{
    r = (pow(lr,2)+lr-1+pow(mr,2))/((2*lr-1)*(2*lr+3));
    if (m1 == 1)
      r = 0.5*r;
  }
  else if (m2 == m1 + 2)
{
    if (m1 == 0)
      eps = sqrt(2.);
    r = eps*sqrt((lr-mr-1)*(lr-mr)*(lr+mr+1)*(lr+mr+2))/(2*(2*lr-1)*(2*lr+3));
}
}
else if (l2 == l1 + 2)
{
  if (m2 == m1 - 2)
{
    if (m1 == 0)
      eps = 1. / sqrt(2.);
    else if (m1 == 2)
      eps = sqrt(2.);
    r = -eps*sqrt((lr-mr+1)*(lr-mr+2)*(lr-mr+3)*(lr-mr+4)/((2*lr+1)*(2*lr+5)))/(4*(2*lr+3));
```

```
      }
      else if (m2 == m1)
    {
        r = -sqrt((lr+mr+1)*(lr+mr+2)*(lr-mr+1)*(lr-mr+2)/((2*lr+1)*(2*lr+5)))/(2*(2*lr+3));
        if (m1 == 1)
          r = 0.5*r;
      }
      else if (m2 == m1 + 2)
    {
        if (m1 == 0)
          eps = sqrt(2.);
        r = -eps*sqrt((lr+mr+1)*(lr+mr+2)*(lr+mr+3)*(lr+mr+4)/((2*lr+1)*(2*lr+5)))/(4*(2*lr+3));
    }
    }
  }
}
else if (dim1 == 2)
{
  if (dim2 == 0)
  {
    if (l2 == l1 - 2)
    {
      if (m2 == m1 - 1)
    {
        if (m1 == 0)
          eps = 1. / sqrt(2.);
      else if (m1 == 1)
          eps = sqrt(2.);
        r = -eps*sqrt((lr+mr-2)*(lr+mr-1)*(lr+mr)*(lr-mr)/((2*lr-3)*(2*lr+1)))/(2*(2*lr-1));
      }
      else if (m2 == m1 + 1)
    {
        if (m1 == 0)
          eps = sqrt(2.);
        r = eps*sqrt((lr+mr)*(lr-mr-2)*(lr-mr-1)*(lr-mr)/((2*lr-3)*(2*lr+1)))/(2*(2*lr-1));
    }
    }
    else if (l2 == l1)
```

```
  {
    if (m2 == m1 - 1)
  {
      if (m1 == 0)
        eps = 1. / sqrt(2.);
    else if (m1 == 1)
        eps = sqrt(2.);
      r = eps*(-sqrt((lr+mr)*(lr-mr+1))*(lr+mr-1)/(2*(2*lr-1)*(2*lr+1))+sqrt((lr+mr)*
               (lr-mr+1))*(lr-mr+2)/(2*(2*lr+1)*(2*lr+3)));
    }
    else if (m2 == m1 + 1)
  {
      if (m1 == 0)
        eps = sqrt(2.);
      r = eps*(sqrt((lr+mr+1)*(lr-mr))*(lr-mr-1)/(2*(2*lr-1)*(2*lr+1))-sqrt((lr+mr+1)*
                    (lr-mr))*(lr+mr+2)/(2*(2*lr+1)*(2*lr+3)));
  }
  }
  else if (l2 == l1 + 2)
  {
    if (m2 == m1 - 1)
  {
      if (m1 == 0)
        eps = 1. / sqrt(2.);
    else if (m1 == 1)
        eps = sqrt(2.);
      r = eps*sqrt((lr+mr+1)*(lr-mr+1)*(lr-mr+2)*(lr-mr+3)/((2*lr+1)*(2*lr+5)))/(2*(2*lr+3));
    }
    else if (m2 == m1 + 1)
  {
      if (m1 == 0)
        eps = sqrt(2.);
      r = -eps*sqrt((lr+mr+1)*(lr+mr+2)*(lr+mr+3)*(lr-mr+1)/((2*lr+1)*(2*lr+5)))/(2*(2*lr+3));
  }
  }
}
else if (dim2 == 2)
{
```

```
      if (l2 == l1 - 2)
      {
        if (m2 == m1)
      {
          r = sqrt((lr-mr-1)*(lr-mr)*(lr+mr-1)*(lr+mr)/((2*lr-3)*(2*lr+1)))/(2*lr-1);
        }
      }
      else if (l2 == l1)
      {
        if (m2 == m1)
      {
          r = (2*pow(lr,2)+2*lr-1-2*pow(mr,2))/((2*lr-1)*(2*lr+3));
        }
      }
      else if (l2 == l1 + 2)
      {
        if (m2 == m1)
      {
          r = sqrt((lr-mr+1)*(lr-mr+2)*(lr+mr+1)*(lr+mr+2)/((2*lr+1)*(2*lr+5)))/(2*lr+3);
        }
      }
    }
  }
}
else if (phase2 == 1)
{
  if (m2 != 0)
  {
    if (dim1 == 0)
    {
      if (dim2 == 1)
      {
        if (l2 == l1 - 2)
        {
          if (m2 == m1 - 2)
        {
            if (m1 == 0)
              eps = 1. / sqrt(2.);
```

```
                else if (m1 == 2)
                  eps = sqrt(2.);
                r = -eps*sqrt((lr+mr-3)*(lr+mr-2)*(lr+mr-1)*(lr+mr)/((2*lr-3)*(2*lr+1)))/(4*(2*lr-1));
              }
              else if (m2 == m1)
            {
                if (m1 == 1)
              {
                  r = -sqrt((lr-mr)*(lr-mr-1)*(lr+mr)*(lr+mr-1)/((2*lr-3)*(2*lr+1)))/(4*(2*lr-1));
              }
              }
              else if (m2 == m1 + 2)
            {
                if (m1 == 0)
                eps = sqrt(2.);
                r = eps*sqrt((lr-mr-3)*(lr-mr-2)*(lr-mr-1)*(lr-mr)/((2*lr-3)*(2*lr+1)))/(4*(2*lr-1));
            }
            }
            else if (l2 == l1)
            {
              if (m2 == m1 - 2)
            {
                if (m1 == 0)
                  eps = 1. / sqrt(2.);
                else if (m1 == 2)
                  eps = sqrt(2.);
                r = eps*sqrt((lr-mr+1)*(lr-mr+2)*(lr+mr-1)*(lr+mr))/(2*(2*lr-1)*(2*lr+3));
              }
              else if (m2 == m1)
            {
                if (m1 == 1)
              {
                  r = ((lr-mr)*(lr-mr-1)/((2*lr-
1)*(2*lr+1))+(lr+mr+1)*(lr+mr+2)/((2*lr+1)*(2*lr+3)))/4.;
                }
            }
              else if (m2 == m1 + 2)
            {
```

```
                if (m1 == 0)
                eps = sqrt(2.);
                r = -eps*sqrt((lr-mr-1)*(lr-mr)*(lr+mr+1)*(lr+mr+2))/(2*(2*lr-1)*(2*lr+3));
            }
            }
            else if (l2 == l1 + 2)
            {
              if (m2 == m1 - 2)
            {
                if (m1 == 0)
                  eps = 1. / sqrt(2.);
                else if (m1 == 2)
                  eps = sqrt(2.);
                r = -eps*sqrt((lr-mr+1)*(lr-mr+2)*(lr-mr+3)*(lr-
mr+4)/((2*lr+1)*(2*lr+5)))/(4*(2*lr+3));
              }
              else if (m2 == m1)
            {
                if (m1 == 1)
              {
                  r = -sqrt((lr+mr+1)*(lr+mr+2)*(lr-mr+1)*(lr-mr+2)/((2*lr+1)*(2*lr+5)))/(4*(2*lr+3));
              }
            }
              else if (m2 == m1 + 2)
            {
                if (m1 == 0)
                eps = sqrt(2.);
                r = eps*sqrt((lr+mr+1)*(lr+mr+2)*(lr+mr+3)*(lr+mr+4)/((2*lr+1)*(2*lr+5)))/(4*(2*lr+3));
            }
            }
          }
        }
        else if (dim1 == 1)
        {
          if (dim2 == 0)
          {
            if (l2 == l1 - 2)
            {
```

```
              if (m2 == m1 - 2)
            {
                if (m1 == 0)
                  eps = 1. / sqrt(2.);
                else if (m1 == 2)
                  eps = sqrt(2.);
                r = -eps*sqrt((lr+mr-3)*(lr+mr-2)*(lr+mr-1)*(lr+mr)/((2*lr-3)*(2*lr+1)))/(4*(2*lr-1));
              }
              else if (m2 == m1)
            {
                if (m1 == 1)
              {
                  r = -sqrt((lr-mr)*(lr-mr-1)*(lr+mr)*(lr+mr-1)/((2*lr-3)*(2*lr+1)))/(4*(2*lr-1));
              }
              }
              else if (m2 == m1 + 2)
            {
                if (m1 == 0)
                eps = sqrt(2.);
                r = eps*sqrt((lr-mr-3)*(lr-mr-2)*(lr-mr-1)*(lr-mr)/((2*lr-3)*(2*lr+1)))/(4*(2*lr-1));
            }
            }
            else if (l2 == l1)
            {
              if (m2 == m1 - 2)
            {
                if (m1 == 0)
                  eps = 1. / sqrt(2.);
                else if (m1 == 2)
                  eps = sqrt(2.);
                r = eps*sqrt((lr-mr+1)*(lr-mr+2)*(lr+mr-1)*(lr+mr))/(2*(2*lr-1)*(2*lr+3));
              }
              else if (m2 == m1)
            {
                if (m1 == 1)
              {
                  r = ((lr-mr)*(lr-mr-1)/((2*lr-
1)*(2*lr+1))+(lr+mr+1)*(lr+mr+2)/((2*lr+1)*(2*lr+3)))/4.;
```

```
                }
            }
              else if (m2 == m1 + 2)
            {
                if (m1 == 0)
                eps = sqrt(2.);
                r = -eps*sqrt((lr-mr-1)*(lr-mr)*(lr+mr+1)*(lr+mr+2))/(2*(2*lr-1)*(2*lr+3));
            }
            }
            else if (l2 == l1 + 2)
            {
              if (m2 == m1 - 2)
            {
                if (m1 == 0)
                  eps = 1. / sqrt(2.);
                else if (m1 == 2)
                  eps = sqrt(2.);
                r = -eps*sqrt((lr-mr+1)*(lr-mr+2)*(lr-mr+3)*(lr-
mr+4)/((2*lr+1)*(2*lr+5)))/(4*(2*lr+3));
              }
              else if (m2 == m1)
            {
                if (m1 == 1)
              {
                  r = -sqrt((lr+mr+1)*(lr+mr+2)*(lr-mr+1)*(lr-mr+2)/((2*lr+1)*(2*lr+5)))/(4*(2*lr+3));
              }
            }
              else if (m2 == m1 + 2)
            {
                if (m1 == 0)
                eps = sqrt(2.);
                r = eps*sqrt((lr+mr+1)*(lr+mr+2)*(lr+mr+3)*(lr+mr+4)/((2*lr+1)*(2*lr+5)))/(4*(2*lr+3));
            }
            }
          }
          else if (dim2 == 2)
          {
            if (l2 == l1 - 2)
```

```
            {
              if (m2 == m1 - 1)
            {
                if (m1 == 0)
                  eps = 1. / sqrt(2.);
                else if (m1 == 1)
                eps = sqrt(2.);
                r = eps*sqrt((lr+mr-2)*(lr+mr-1)*(lr+mr)*(lr-mr)/((2*lr-3)*(2*lr+1)))/(2*(2*lr-1));
              }
              else if (m2 == m1 + 1)
            {
                if (m1 == 0)
                  eps = sqrt(2.);
                r = eps*sqrt((lr+mr)*(lr-mr-1)*(lr-mr)*(lr-mr-2)/((2*lr-3)*(2*lr+1)))/(2*(2*lr-1));
            }
            }
            else if (l2 == l1)
            {
              if (m2 == m1 - 1)
            {
                if (m1 == 0)
                  eps = 1. / sqrt(2.);
                else if (m1 == 1)
                eps = sqrt(2.);
                r = eps*(sqrt((lr-mr+1)*(lr+mr))*(lr+mr-1)/(2*(2*lr-1)*(2*lr+1))-sqrt((lr-
mr+1)*(lr+mr))*(lr-mr+2)/(2*(2*lr+1)*(2*lr+3)));
              }
              else if (m2 == m1 + 1)
            {
                if (m1 == 0)
                  eps = sqrt(2.);
                r = eps*(sqrt((lr-mr)*(lr+mr+1))*(lr-mr-1)/(2*(2*lr-1)*(2*lr+1))-sqrt((lr-
mr)*(lr+mr+1))*(lr+mr+2)/(2*(2*lr+1)*(2*lr+3)));
            }
            }
            else if (l2 == l1 + 2)
            {
              if (m2 == m1 - 1)
```

```
            {
                if (m1 == 0)
                   eps = 1. / sqrt(2.);
                else if (m1 == 1)
                 eps = sqrt(2.);
                r = -eps*sqrt((lr+mr+1)*(lr-mr+1)*(lr-mr+2)*(lr-
mr+3)/((2*lr+1)*(2*lr+5)))/(2*(2*lr+3));
              }
              else if (m2 == m1 + 1)
            {
                if (m1 == 0)
                  eps = sqrt(2.);
                r = -eps*sqrt((lr+mr+1)*(lr+mr+2)*(lr+mr+3)*(lr-
mr+1)/((2*lr+1)*(2*lr+5)))/(2*(2*lr+3));
            }
            }
          }
        }
        else if (dim1 == 2)
        {
          if (dim2 == 1)
          {
            if (l2 == l1 - 2)
            {
              if (m2 == m1 - 1)
            {
                if (m1 == 0)
                  eps = 1. / sqrt(2.);
                else if (m1 == 1)
                eps = sqrt(2.);
                r = eps*sqrt((lr+mr-2)*(lr+mr-1)*(lr+mr)*(lr-mr)/((2*lr-3)*(2*lr+1)))/(2*(2*lr-1));
              }
              else if (m2 == m1 + 1)
            {
                if (m1 == 0)
                  eps = sqrt(2.);
                r = eps*sqrt((lr+mr)*(lr-mr-1)*(lr-mr)*(lr-mr-2)/((2*lr-3)*(2*lr+1)))/(2*(2*lr-1));
            }
```

```
              }
              else if (l2 == l1)
              {
                if (m2 == m1 - 1)
              {
                  if (m1 == 0)
                    eps = 1. / sqrt(2.);
                  else if (m1 == 1)
                  eps = sqrt(2.);
                  r = eps*(sqrt((lr-mr+1)*(lr+mr))*(lr+mr-1)/(2*(2*lr-1)*(2*lr+1))-sqrt((lr-
mr+1)*(lr+mr))*(lr-mr+2)/(2*(2*lr+1)*(2*lr+3)));
                }
                else if (m2 == m1 + 1)
              {
                  if (m1 == 0)
                    eps = sqrt(2.);
                  r = eps*(sqrt((lr-mr)*(lr+mr+1))*(lr-mr-1)/(2*(2*lr-1)*(2*lr+1))-sqrt((lr-
mr)*(lr+mr+1))*(lr+mr+2)/(2*(2*lr+1)*(2*lr+3)));
              }
              }
              else if (l2 == l1 + 2)
              {
                if (m2 == m1 - 1)
              {
                  if (m1 == 0)
                    eps = 1. / sqrt(2.);
                  else if (m1 == 1)
                  eps = sqrt(2.);
                  r = -eps*sqrt((lr+mr+1)*(lr-mr+1)*(lr-mr+2)*(lr-
mr+3)/((2*lr+1)*(2*lr+5)))/(2*(2*lr+3));
                }
                else if (m2 == m1 + 1)
              {
                  if (m1 == 0)
                    eps = sqrt(2.);
                  r = -eps*sqrt((lr+mr+1)*(lr+mr+2)*(lr+mr+3)*(lr-
mr+1)/((2*lr+1)*(2*lr+5)))/(2*(2*lr+3));
              }
```

```
          }
        }
      }
    }
  }
}
else if (phase1 == 1)
{
  if (m1 != 0)
  {
    if (phase2 == 0)
    {
      if (dim1 == 0)
      {
        if (dim2 == 1)
        {
          if (l2 == l1 - 2)
          {
            if (m2 == m1 - 2)
            {
                if (m1 == 0)
                  eps = 1. / sqrt(2.);
                else if (m1 == 2)
                  eps = sqrt(2.);
                r = eps*sqrt((lr+mr-3)*(lr+mr-2)*(lr+mr-1)*(lr+mr)/((2*lr-3)*(2*lr+1)))/(4*(2*lr-1));
            }
            else if (m2 == m1)
            {
                if (m1 == 1)
              {
                  r = -sqrt((lr-mr)*(lr-mr-1)*(lr+mr)*(lr+mr-1)/((2*lr-3)*(2*lr+1)))/(4*(2*lr-1));
              }
              }
            else if (m2 == m1 + 2)
            {
                if (m1 == 0)
                eps = sqrt(2.);
                r = -eps*sqrt((lr-mr-3)*(lr-mr-2)*(lr-mr-1)*(lr-mr)/((2*lr-3)*(2*lr+1)))/(4*(2*lr-1));
```

```
            }
            }
            else if (l2 == l1)
            {
              if (m2 == m1 - 2)
            {
                if (m1 == 0)
                  eps = 1. / sqrt(2.);
                else if (m1 == 2)
                  eps = sqrt(2.);
                r = -eps*sqrt((lr-mr+1)*(lr-mr+2)*(lr+mr-1)*(lr+mr))/(2*(2*lr-1)*(2*lr+3));
              }
              else if (m2 == m1)
            {
                if (m1 == 1)
              {
                  r = ((lr-mr)*(lr-mr-1)/((2*lr-
1)*(2*lr+1))+(lr+mr+1)*(lr+mr+2)/((2*lr+1)*(2*lr+3)))/4.;
                }
            }
              else if (m2 == m1 + 2)
            {
                if (m1 == 0)
                eps = sqrt(2.);
                r = eps*sqrt((lr-mr-1)*(lr-mr)*(lr+mr+1)*(lr+mr+2))/(2*(2*lr-1)*(2*lr+3));
            }
            }
            else if (l2 == l1 + 2)
            {
              if (m2 == m1 - 2)
            {
                if (m1 == 0)
                  eps = 1. / sqrt(2.);
                else if (m1 == 2)
                  eps = sqrt(2.);
                r = eps*sqrt((lr-mr+1)*(lr-mr+2)*(lr-mr+3)*(lr-mr+4)/((2*lr+1)*(2*lr+5)))/(4*(2*lr+3));
              }
              else if (m2 == m1)
```

```
            {
                if (m1 == 1)
              {
                  r = -sqrt((lr+mr+1)*(lr+mr+2)*(lr-mr+1)*(lr-mr+2)/((2*lr+1)*(2*lr+5)))/(4*(2*lr+3));
              }
            }
              else if (m2 == m1 + 2)
            {
                if (m1 == 0)
                eps = sqrt(2.);
                r = -
eps*sqrt((lr+mr+1)*(lr+mr+2)*(lr+mr+3)*(lr+mr+4)/((2*lr+1)*(2*lr+5)))/(4*(2*lr+3));
            }
            }
          }
        }
        else if (dim1 == 1)
        {
          if (dim2 == 0)
          {
            if (l2 == l1 - 2)
            {
              if (m2 == m1 - 2)
            {
                if (m1 == 0)
                  eps = 1. / sqrt(2.);
                else if (m1 == 2)
                  eps = sqrt(2.);
                r = eps*sqrt((lr+mr-3)*(lr+mr-2)*(lr+mr-1)*(lr+mr)/((2*lr-3)*(2*lr+1)))/(4*(2*lr-1));
              }
              else if (m2 == m1)
            {
                if (m1 == 1)
              {
                  r = -sqrt((lr-mr)*(lr-mr-1)*(lr+mr)*(lr+mr-1)/((2*lr-3)*(2*lr+1)))/(4*(2*lr-1));
              }
              }
              else if (m2 == m1 + 2)
```

```
            {
                if (m1 == 0)
                eps = sqrt(2.);
                r = -eps*sqrt((lr-mr-3)*(lr-mr-2)*(lr-mr-1)*(lr-mr)/((2*lr-3)*(2*lr+1)))/(4*(2*lr-1));
            }
            }
            else if (l2 == l1)
            {
              if (m2 == m1 - 2)
            {
                if (m1 == 0)
                  eps = 1. / sqrt(2.);
                else if (m1 == 2)
                  eps = sqrt(2.);
                r = -eps*sqrt((lr-mr+1)*(lr-mr+2)*(lr+mr-1)*(lr+mr))/(2*(2*lr-1)*(2*lr+3));
              }
              else if (m2 == m1)
            {
                if (m1 == 1)
              {
                  r = ((lr-mr)*(lr-mr-1)/((2*lr-
1)*(2*lr+1))+(lr+mr+1)*(lr+mr+2)/((2*lr+1)*(2*lr+3)))/4.;
                }
            }
              else if (m2 == m1 + 2)
            {
                if (m1 == 0)
                eps = sqrt(2.);
                r = eps*sqrt((lr-mr-1)*(lr-mr)*(lr+mr+1)*(lr+mr+2))/(2*(2*lr-1)*(2*lr+3));
            }
            }
            else if (l2 == l1 + 2)
            {
              if (m2 == m1 - 2)
            {
                if (m1 == 0)
                  eps = 1. / sqrt(2.);
                else if (m1 == 2)
```

```
                    eps = sqrt(2.);
                  r = eps*sqrt((lr-mr+1)*(lr-mr+2)*(lr-mr+3)*(lr-mr+4)/((2*lr+1)*(2*lr+5)))/(4*(2*lr+3));
                }
                else if (m2 == m1)
              {
                  if (m1 == 1)
                {
                    r = -sqrt((lr+mr+1)*(lr+mr+2)*(lr-mr+1)*(lr-mr+2)/((2*lr+1)*(2*lr+5)))/(4*(2*lr+3));
                }
              }
                else if (m2 == m1 + 2)
              {
                  if (m1 == 0)
                  eps = sqrt(2.);
                  r = -
eps*sqrt((lr+mr+1)*(lr+mr+2)*(lr+mr+3)*(lr+mr+4)/((2*lr+1)*(2*lr+5)))/(4*(2*lr+3));
              }
              }
            }
            else if (dim2 == 2)
            {
              if (l2 == l1 - 2)
              {
                if (m2 == m1 - 1)
              {
                  if (m1 == 0)
                    eps = 1. / sqrt(2.);
                  else if (m1 == 1)
                  eps = sqrt(2.);
                  r = -eps*sqrt((lr+mr-2)*(lr+mr-1)*(lr+mr)*(lr-mr)/((2*lr-3)*(2*lr+1)))/(2*(2*lr-1));
                }
                else if (m2 == m1 + 1)
              {
                  if (m1 == 0)
                    eps = sqrt(2.);
                  r = -eps*sqrt((lr+mr)*(lr-mr-1)*(lr-mr)*(lr-mr-2)/((2*lr-3)*(2*lr+1)))/(2*(2*lr-1));
              }
              }
```

```
            else if (l2 == l1)
            {
              if (m2 == m1 - 1)
            {
                if (m1 == 0)
                  eps = 1. / sqrt(2.);
                else if (m1 == 1)
                eps = sqrt(2.);
                r = eps*(-sqrt((lr-mr+1)*(lr+mr))*(lr+mr-1)/(2*(2*lr-1)*(2*lr+1))+sqrt((lr-
mr+1)*(lr+mr))*(lr-mr+2)/(2*(2*lr+1)*(2*lr+3)));
              }
              else if (m2 == m1 + 1)
            {
                if (m1 == 0)
                  eps = sqrt(2.);
                r = eps*(-sqrt((lr-mr)*(lr+mr+1))*(lr-mr-1)/(2*(2*lr-1)*(2*lr+1))+sqrt((lr-
mr)*(lr+mr+1))*(lr+mr+2)/(2*(2*lr+1)*(2*lr+3)));
            }
            }
            else if (l2 == l1 + 2)
            {
              if (m2 == m1 - 1)
            {
                if (m1 == 0)
                  eps = 1. / sqrt(2.);
                else if (m1 == 1)
                eps = sqrt(2.);
                r = eps*sqrt((lr+mr+1)*(lr-mr+1)*(lr-mr+2)*(lr-mr+3)/((2*lr+1)*(2*lr+5)))/(2*(2*lr+3));
              }
              else if (m2 == m1 + 1)
            {
                if (m1 == 0)
                  eps = sqrt(2.);
                r = eps*sqrt((lr+mr+1)*(lr+mr+2)*(lr+mr+3)*(lr-mr+1)/((2*lr+1)*(2*lr+5)))/(2*(2*lr+3));
            }
            }
          }
        }
```

```
        else if (dim1 == 2)
        {
          if (dim2 == 1)
          {
            if (l2 == l1 - 2)
            {
              if (m2 == m1 - 1)
            {
                if (m1 == 0)
                  eps = 1. / sqrt(2.);
                else if (m1 == 1)
                eps = sqrt(2.);
                r = -eps*sqrt((lr+mr-2)*(lr+mr-1)*(lr+mr)*(lr-mr)/((2*lr-3)*(2*lr+1)))/(2*(2*lr-1));
              }
              else if (m2 == m1 + 1)
            {
                if (m1 == 0)
                  eps = sqrt(2.);
                r = -eps*sqrt((lr+mr)*(lr-mr-1)*(lr-mr)*(lr-mr-2)/((2*lr-3)*(2*lr+1)))/(2*(2*lr-1));
            }
            }
            else if (l2 == l1)
            {
              if (m2 == m1 - 1)
            {
                if (m1 == 0)
                  eps = 1. / sqrt(2.);
                else if (m1 == 1)
                eps = sqrt(2.);
                r = eps*(-sqrt((lr-mr+1)*(lr+mr))*(lr+mr-1)/(2*(2*lr-1)*(2*lr+1))+sqrt((lr-
mr+1)*(lr+mr))*(lr-mr+2)/(2*(2*lr+1)*(2*lr+3)));
              }
              else if (m2 == m1 + 1)
            {
                if (m1 == 0)
                  eps = sqrt(2.);
                r = eps*(-sqrt((lr-mr)*(lr+mr+1))*(lr-mr-1)/(2*(2*lr-1)*(2*lr+1))+sqrt((lr-
mr)*(lr+mr+1))*(lr+mr+2)/(2*(2*lr+1)*(2*lr+3)));
```

```
      }
      }
      else if (l2 == l1 + 2)
      {
        if (m2 == m1 - 1)
      {
          if (m1 == 0)
            eps = 1. / sqrt(2.);
          else if (m1 == 1)
          eps = sqrt(2.);
          r = eps*sqrt((lr+mr+1)*(lr-mr+1)*(lr-mr+2)*(lr-mr+3)/((2*lr+1)*(2*lr+5)))/(2*(2*lr+3));
        }
        else if (m2 == m1 + 1)
      {
          if (m1 == 0)
            eps = sqrt(2.);
          r = eps*sqrt((lr+mr+1)*(lr+mr+2)*(lr+mr+3)*(lr-mr+1)/((2*lr+1)*(2*lr+5)))/(2*(2*lr+3));
      }
      }
    }
  }
}
else if (phase2 == 1)
{
  if (m2 != 0)
  {
    if (dim1 == 0)
    {
      if (dim2 == 0)
      {
        if (l2 == l1 - 2)
        {
          if (m2 == m1 - 2)
        {
            if (m1 == 0)
              eps = 1. / sqrt(2.);
            else if (m1 == 2)
              eps = sqrt(2.);
```

```
      r = eps*sqrt((lr+mr-3)*(lr+mr-2)*(lr+mr-1)*(lr+mr)/((2*lr-3)*(2*lr+1)))/(4*(2*lr-1));
    }
    else if (m2 == m1)
  {
      r = -sqrt((lr+mr-1)*(lr+mr)*(lr-mr-1)*(lr-mr)/((2*lr-3)*(2*lr+1)))/(2*(2*lr-1));
      if (m1 == 1)
         r = 0.5*r;
    }
    else if (m2 == m1 + 2)
  {
       if (m1 == 0)
         eps = sqrt(2.);
       r = eps*sqrt((lr-mr-3)*(lr-mr-2)*(lr-mr-1)*(lr-mr)/((2*lr-3)*(2*lr+1)))/(4*(2*lr-
1));
  }
  }
  else if (l2 == l1)
  {
    if (m2 == m1 - 2)
  {
      if (m1 == 0)
        eps = 1. / sqrt(2.);
      else if (m1 == 2)
        eps = sqrt(2.);
      r = -eps*sqrt((lr-mr+1)*(lr-mr+2)*(lr+mr-1)*(lr+mr))/(2*(2*lr-1)*(2*lr+3));
    }
    else if (m2 == m1)
  {
      r = (pow(lr,2)+lr-1+pow(mr,2))/((2*lr-1)*(2*lr+3));
      if (m1 == 1)
        r = 0.5*r;
    }
    else if (m2 == m1 + 2)
  {
      if (m1 == 0)
        eps = sqrt(2.);
      r = -eps*sqrt((lr-mr-1)*(lr-mr)*(lr+mr+1)*(lr+mr+2))/(2*(2*lr-1)*(2*lr+3));
  }
```

```
                }
                else if (l2 == l1 + 2)
                {
                  if (m2 == m1 - 2)
                {
                    if (m1 == 0)
                      eps = 1. / sqrt(2.);
                    else if (m1 == 2)
                      eps = sqrt(2.);
                    r = eps*sqrt((lr-mr+1)*(lr-mr+2)*(lr-mr+3)*(lr-
mr+4)/((2*lr+1)*(2*lr+5)))/(4*(2*lr+3));
                  }
                  else if (m2 == m1)
                {
                    r = -sqrt((lr+mr+1)*(lr+mr+2)*(lr-mr+1)*(lr-mr+2)/((2*lr+1)*(2*lr+5)))/(2*(2*lr+3));
                    if (m1 == 1)
                      r = 0.5*r;
                  }
                  else if (m2 == m1 + 2)
                {
                    if (m1 == 0)
                      eps = sqrt(2.);
                    r =
eps*sqrt((lr+mr+1)*(lr+mr+2)*(lr+mr+3)*(lr+mr+4)/((2*lr+1)*(2*lr+5)))/(4*(2*lr+3));
                }
                }
              }
              else if (dim2 == 2)
              {
                if (l2 == l1 - 2)
                {
                  if (m2 == m1 - 1)
                {
                    if (m1 == 0)
                      eps = 1. / sqrt(2.);
                  else if (m1 == 1)
                      eps = sqrt(2.);
                    r = -eps*sqrt((lr+mr-2)*(lr+mr-1)*(lr+mr)*(lr-mr)/((2*lr-3)*(2*lr+1)))/(2*(2*lr-1));
```

```
                }
                else if (m2 == m1 + 1)
              {
                  if (m1 == 0)
                    eps = sqrt(2.);
                  r = eps*sqrt((lr+mr)*(lr-mr-2)*(lr-mr-1)*(lr-mr)/((2*lr-3)*(2*lr+1)))/(2*(2*lr-1));
              }
              }
              else if (l2 == l1)
              {
                if (m2 == m1 - 1)
              {
                  if (m1 == 0)
                    eps = 1. / sqrt(2.);
                else if (m1 == 1)
                    eps = sqrt(2.);
                  r = eps*(-sqrt((lr+mr)*(lr-mr+1))*(lr+mr-1)/(2*(2*lr-1)*(2*lr+1))+sqrt((lr+mr)*(lr-
mr+1))*(lr-mr+2)/(2*(2*lr+1)*(2*lr+3)));
                }
                else if (m2 == m1 + 1)
              {
                  if (m1 == 0)
                    eps = sqrt(2.);
                  r = eps*(sqrt((lr+mr+1)*(lr-mr))*(lr-mr-1)/(2*(2*lr-1)*(2*lr+1))-sqrt((lr+mr+1)*(lr-
mr))*(lr+mr+2)/(2*(2*lr+1)*(2*lr+3)));
              }
              }
              else if (l2 == l1 + 2)
              {
                if (m2 == m1 - 1)
              {
                  if (m1 == 0)
                    eps = 1. / sqrt(2.);
                else if (m1 == 1)
                    eps = sqrt(2.);
                  r = eps*sqrt((lr+mr+1)*(lr-mr+1)*(lr-mr+2)*(lr-
mr+3)/((2*lr+1)*(2*lr+5)))/(2*(2*lr+3));
                }
```

```
                else if (m2 == m1 + 1)
              {
                  if (m1 == 0)
                    eps = sqrt(2.);
                  r = -eps*sqrt((lr+mr+1)*(lr+mr+2)*(lr+mr+3)*(lr-
mr+1)/((2*lr+1)*(2*lr+5)))/(2*(2*lr+3));
              }
              }
            }
          }
          else if (dim1 == 1)
          {
            if (dim2 == 1)
            {
              if (l2 == l1 - 2)
              {
                if (m2 == m1 - 2)
              {
                  if (m1 == 0)
                    eps = 1. / sqrt(2.);
                  else if (m1 == 2)
                    eps = sqrt(2.);
                  r = -eps*sqrt((lr+mr-3)*(lr+mr-2)*(lr+mr-1)*(lr+mr)/((2*lr-3)*(2*lr+1)))/(4*(2*lr-
1));
                }
                else if (m2 == m1)
              {
                  r = -sqrt((lr-mr-1)*(lr-mr)*(lr+mr-1)*(lr+mr)/((2*lr-3)*(2*lr+1)))/(2*(2*lr-1));
                  if (m1 == 1)
                    r = 1.5*r;
                }
                else if (m2 == m1 + 2)
              {
                  if (m1 == 0)
                    eps = sqrt(2.);
                  r = -eps*sqrt((lr-mr)*(lr-mr-1)*(lr-mr-2)*(lr-mr-3)/((2*lr-3)*(2*lr+1)))/(4*(2*lr-
1));
              }
```

```
                }
                else if (l2 == l1)
                {
                  if (m2 == m1 - 2)
                {
                    if (m1 == 0)
                      eps = 1. / sqrt(2.);
                    else if (m1 == 2)
                      eps = sqrt(2.);
                    r = eps*sqrt((lr-mr+1)*(lr-mr+2)*(lr+mr-1)*(lr+mr))/(2*(2*lr-1)*(2*lr+3));
                  }
                  else if (m2 == m1)
                {
                    r = (pow(lr,2)+lr-1+pow(mr,2))/((2*lr-1)*(2*lr+3));
                    if (m1 == 1)
                      r = 1.5*r;
                  }
                  else if (m2 == m1 + 2)
                {
                    if (m1 == 0)
                     eps = sqrt(2.);
                    r = eps*sqrt((lr-mr-1)*(lr-mr)*(lr+mr+1)*(lr+mr+2))/(2*(2*lr-1)*(2*lr+3));
                }
                }
                else if (l2 == l1 + 2)
                {
                  if (m2 == m1 - 2)
                {
                    if (m1 == 0)
                      eps = 1. / sqrt(2.);
                    else if (m1 == 2)
                      eps = sqrt(2.);
                    r = -eps*sqrt((lr-mr+1)*(lr-mr+2)*(lr-mr+3)*(lr-
mr+4)/((2*lr+1)*(2*lr+5)))/(4*(2*lr+3));
                  }
                  else if (m2 == m1)
                {
                    r = -sqrt((lr+mr+1)*(lr+mr+2)*(lr-mr+1)*(lr-mr+2)/((2*lr+1)*(2*lr+5)))/(2*(2*lr+3));
```

```
                    if (m1 == 1)
                      r = 1.5*r;
                  }
                  else if (m2 == m1 + 2)
                {
                    if (m1 == 0)
                      eps = sqrt(2.);
                    r = -
eps*sqrt((lr+mr+1)*(lr+mr+2)*(lr+mr+3)*(lr+mr+4)/((2*lr+1)*(2*lr+5)))/(4*(2*lr+3));
                }
                }
              }
            }
            else if (dim1 == 2)
            {
              if (dim2 == 0)
              {
                if (l2 == l1 - 2)
                {
                  if (m2 == m1 - 1)
                {
                    if (m1 == 0)
                      eps = 1. / sqrt(2.);
                  else if (m1 == 1)
                      eps = sqrt(2.);
                    r = -eps*sqrt((lr+mr-2)*(lr+mr-1)*(lr+mr)*(lr-mr)/((2*lr-3)*(2*lr+1)))/(2*(2*lr-1));
                  }
                  else if (m2 == m1 + 1)
                {
                    if (m1 == 0)
                      eps = sqrt(2.);
                    r = eps*sqrt((lr+mr)*(lr-mr-2)*(lr-mr-1)*(lr-mr)/((2*lr-3)*(2*lr+1)))/(2*(2*lr-1));
                }
                }
                else if (l2 == l1)
                {
                  if (m2 == m1 - 1)
                {
```

```
                  if (m1 == 0)
                    eps = 1. / sqrt(2.);
                else if (m1 == 1)
                    eps = sqrt(2.);
                  r = eps*(-sqrt((lr+mr)*(lr-mr+1))*(lr+mr-1)/(2*(2*lr-1)*(2*lr+1))+sqrt((lr+mr)*(lr-
mr+1))*(lr-mr+2)/(2*(2*lr+1)*(2*lr+3)));
                }
                else if (m2 == m1 + 1)
              {
                  if (m1 == 0)
                    eps = sqrt(2.);
                  r = eps*(sqrt((lr+mr+1)*(lr-mr))*(lr-mr-1)/(2*(2*lr-1)*(2*lr+1))-sqrt((lr+mr+1)*(lr-
mr))*(lr+mr+2)/(2*(2*lr+1)*(2*lr+3)));
              }
              }
              else if (l2 == l1 + 2)
              {
                if (m2 == m1 - 1)
              {
                  if (m1 == 0)
                    eps = 1. / sqrt(2.);
                else if (m1 == 1)
                    eps = sqrt(2.);
                  r = eps*sqrt((lr+mr+1)*(lr-mr+1)*(lr-mr+2)*(lr-
mr+3)/((2*lr+1)*(2*lr+5)))/(2*(2*lr+3));
                }
                else if (m2 == m1 + 1)
              {
                  if (m1 == 0)
                    eps = sqrt(2.);
                  r = -eps*sqrt((lr+mr+1)*(lr+mr+2)*(lr+mr+3)*(lr-
mr+1)/((2*lr+1)*(2*lr+5)))/(2*(2*lr+3));
              }
              }
            }
            else if (dim2 == 2)
            {
              if (l2 == l1 - 2)
```

```
                {
                  if (m2 == m1)
                {
                    r = sqrt((lr-mr-1)*(lr-mr)*(lr+mr-1)*(lr+mr)/((2*lr-3)*(2*lr+1)))/(2*lr-1);
                  }
                }
                else if (l2 == l1)
                {
                  if (m2 == m1)
                {
                    r = (2*pow(lr,2)+2*lr-1-2*pow(mr,2))/((2*lr-1)*(2*lr+3));
                  }
                }
                else if (l2 == l1 + 2)
                {
                  if (m2 == m1)
                {
                    r = sqrt((lr-mr+1)*(lr-mr+2)*(lr+mr+1)*(lr+mr+2)/((2*lr+1)*(2*lr+5)))/(2*lr+3);
                  }
                }
              }
            }
          }
        }
      }
    }
    return(r);
}
sOSM::~sOSM()
{
}
```