LONG-CHARACTERISTICS METHODS WITH PIECEWISE LINEAR SOURCES IN SPACE AND TIME FOR TRANSPORT ON UNSTRUCTURED GRIDS

A Dissertation

by

TARA MARIE PANDYA

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Approved by:

Co-Chairs of Committee,	Marvin Adams
	Jim Morel
Committee Members,	Nancy Amato
	Teresa Bailey
	Jean Ragusa
Head of Department,	Yassin Hassan

December 2012

Major Subject: Nuclear Engineering

ABSTRACT

The method of characteristics (MOC) is a deterministic transport method that has been applied to large-scale problems including those in reactor physics and radiative transfer. Long characteristics, (LC) methods, have been used extensively to discretize and solve transport problems in the spatial domain. There is a need for an equally adequate time-dependent discretization for these transport problems.

The new contributions from this research include the development of a space-time long characteristic (STLC) method with various source approximations including several that employ a piece-wise linear (PWL) approximation spatially. In the prism-PWL (PPWL) method the coefficient of each PWL spatial function is linear in time in each space-time cell. Along with STLC, a PWL-LC method is developed for steady-state problems in (x, y) and (x, y, z). The methods developed in this work use least-squares projections to determine the coefficients of their source approximations.

This work presents a detailed asymptotic analysis of the PWL-LC and STLC methods in the thick diffusion limit, which is of special interest in radiative transfer problems. This is the first such analysis reported for LC methods and these new methods are the first that are expected to perform well in this limit.

Results from test problems executed with a modified version of the Parallel Deterministic Transport code, PDT, show the PWL-LC and STLC methods are more accurate than current methods for streaming problems. From asymptotic analysis and test problems, it is found that the steady-state PWL-LC method is accurate in the thick

diffusion limit with solutions similar to those of analogous discontinuous finite element method, DFEM, solutions. Similarly, the PPWL-STLC method is found to be accurate in time-dependent thick diffusive problems.

STLC is also a promising method for massively parallel applications because it permits the use of track-based sweeping, which appears to have significant advantages over cell-based sweeping. This is a key topic recommended for further research.

DEDICATION

I would like to dedicate this dissertation to my fiancé, Mr. David Gras. You have always supported and encouraged me throughout my education. I thank you for always listening to my frustrations and triumphs about my research, even though you did not always understand the technicalities of my work. Your motivation, optimism, and love have enabled me to complete my graduate education and this research and I will forever be grateful.

ACKNOWLEDGEMENTS

I would first like to thank my committee chair, Dr. Adams, for all of the knowledge and valuable research skills he taught me throughout my education. I am also very appreciative of my committee members, Dr. Jim Morel, Dr. Jean Ragusa, Dr. Nancy Amato, and Dr. Teresa Bailey, for their guidance and support throughout the course of this research. Special thanks go to Mr. W. Daryl Hawkins for his knowledge and extensive help in implementing and completing this research in the PDT code under development at Texas A&M. Also, thanks to Dr. Tim Smith and Mr. Michael Adams for their technical expertise with the PDT code. Thanks also go to my friends and colleagues and the nuclear engineering department, faculty, and staff for making my time at Texas A&M University a rewarding experience.

Many thanks to the Gras family for their continued encouragement to complete this research and pursue my career goals. Finally, thanks to my family for all of their support throughout my undergraduate and graduate careers. Thanks to my brother, Michael Pandya, and my sister, Dr. Rita Pandya, for their insights and witty conversations to keep me grounded. Most of all, thanks to my parents, Dr. Yogesh and Corinne Pandya, for all of their support during my entire education, both monetary and emotional. Without this support I would not have been able to complete this doctoral degree.

NOMENCLATURE

DFEM	Discontinuous Finite Element Method
DO	Discrete Ordinates
FD	Finite Difference
FE	Finite Element
FEM	Finite Element Method
GMRES	General Minimum Residual
LC	Long Characteristics
LD	Linear Discontinuous
LFEM	Linear Finite Element Method
KBA	Koch-Baker-Alcouffe
MOC	Method of Characteristics
PDT	Parallel or Particle Deterministic Transport Code
PPWL	Prism PieceWise Linear
PWC	PieceWise Constant
PWL	PieceWise Linear
PWLD	PieceWise Linear Discontinuous Finite Element Method
STLC	Space-Time Long Characteristics
UCB	Upstream Corner Balance

TABLE OF CONTENTS

Page

ABSTRACT ii

DEDICATION iv

ACKNOWLEDGEMENTS v

NOMENCLATURE vi

TABLE OF CONTENTS vii

LIST OF FIGURES x

LIST OF TABLES xv

1. INTRODUCTION 1

    1.1 Current State of the Problem 2
    1.2 Focus of this Research 6
    1.3 Outline of Remainder 7

2. DEVELOPMENT OF THE PWL-LC METHOD 8

    2.1 Coordinate System Definition in X-Y Geometry 8
    2.2 PWL-LC Least-Squares Method 11
    2.3 Time-Dependent PWL-LC 16
    2.4 Coordinate System Definition in X-Y-Z Geometry 23
    2.5 PWL-LC Method in X-Y-Z 25

3. DEVELOPMENT OF THE STLC METHOD 28

    3.1 Review of Coordinates and STLC Method in X-vT 28
    3.2 Coordinate System Definition in X-Y-vT Geometry 33
    3.3 STLC Least-Squares Method 36
        3.3.1 PWC-STLC 37
        3.3.2 PWL-C-STLC 38
        3.3.3 PWL-L-STLC 39
        3.3.4 PPWL-STLC 41
        3.3.5 Full PWL-STLC 42
    3.4 Prism PWL-LC Method in X-Y-Z 43

Page

4. ANALYSIS OF THE PWL-LC AND STLC LEAST-SQUARES METHODS 45

4.1 Smoothness of Reported Cell-Wise Reaction Rates 45
4.2 Review of Analysis of Radiative Transfer Equations in Diffusion Limit 48
4.3 Time-Dependent PWL-LC Asymptotic Analysis in the Diffusion Limit 54
4.4 PWL-LC Steady-State Asymptotic Analysis in the Diffusion Limit 86
4.5 STLC Asymptotic Analysis in the Diffusion Limit 88
4.5.1 PWC-STLC 89
4.5.2 PWL-C-STLC 90
4.5.3 PWL-L-STLC 91
4.5.4 PPWL-STLC 93
4.6 PPWL-LC Method in X-Y-Z 96

5. SOLVING THE TRANSPORT EQUATION WITH PWL-LC AND STLC 97

5.1 Algorithms 97
5.1.1 Newton Problems 97
5.1.2 Thermal Radiation Problems 105
5.2 Implementation 106
5.2.1 Sweepchunks 109
5.2.2 Time-Dependent Solve 109
5.2.3 STLC Iterative Operator 110
5.2.4 Reporting Values 110
5.2.5 Computational Considerations 111

6. RESULTS 113

6.1 Steady-State PWL-LC Test Problems in X-Y 114
6.1.1 Streaming in a Vacuum 114
6.1.2 Diffusion Limit Symmetric Test Problem 116
6.1.3 Absorbing Block – LC vs. DFEM 121
6.1.4 TRIGA Test Pins 126
6.1.5 Streaming in a Pure Absorber 131
6.1.6 Linear Manufactured Solution 134
6.2 Steady-State PWL-LC Test Problems in X-Y-Z 138
6.2.1 Reactor Pin Cells 138
6.3 Time-Dependent PWL-LC Test Problems in X-Y 150
6.3.1 Streaming in a Vacuum 150
6.3.2 Streaming in a Pure Absorber 154
6.4 STLC Test Problems in X-Y-vT 161

Page

6.4.1 Streaming in a Vacuum 161
6.4.2 Streaming in a Pure Absorber 163
6.4.3 Linear Manufactured Solution 170
6.4.4 The Crooked Pipe (TopHat) Problem 177
6.4.5 Simple Thick Diffusion Limit Problem 186
6.4.6 STLC Order of Convergence in Time 199
6.4.7 Track-Based Conservation 203

7. RECOMMENDATIONS 205

8. SUMMARY AND CONCLUSIONS 210

8.1 Summary 210
8.2 Conclusions 215

REFERENCES 216

APPENDIX A 223

APPENDIX B 322

LIST OF FIGURES

Page

Figure 1. Spatial Cell in (x, y) with Tracks in One Angular Direction 8

Figure 2. Rotated Coordinate System Relation in X-Y 10

Figure 3. Contour Sketch of PWL Basis Function on Pentagonal Cell 14

Figure 4. Rotated Coordinate System Relation in X-Y-Z 25

Figure 5. Space-time Cell with Tracks in One Angular Direction 28

Figure 6. Rotated (u-ω) Coordinate System in x-vt Plane 30

Figure 7. Rotated Coordinate System Relation in X-Y-vT 35

Figure 8. Track Layout in (x, y) for One Angle 46

Figure 9. Unknowns in a Boundary Vertex Equation 64

Figure 10. PDT Execution up to Solving 98

Figure 11. Point-wise Local Convergence Check on Scalar Flux 100

Figure 12. Pseudocode for LC and STLC Radiative Transfer Problems 106

Figure 13. Cell-Average Scalar Flux for Incident Beam in a Vacuum (a) PWL-DFEM (b) PWL-LC 115

Figure 14. Cross Section of Cell-Average Scalar Flux Solution at Y = 2.1 cm (a) PWC-LC (b) PWL-LC 117

Figure 15. PWL-LC Pointwise Scalar Flux on Non-Orthogonal Grid (a) c = 0.9 (b) c = 0.999999 119

Figure 16. Cross Section of Cell-Average Scalar Flux Solution at Y ≅ 2.1 cm (non-orthogonal grid) (a) PWC-LC (b) PWL-LC 120

Figure 17. Geometry and Cross Section for Absorbing Block Problem 122

Page

Figure 18. Cell-Average Scalar Flux Solution for Absorbing Block (a) PWL-LC (b) PWL-DFEM 123

Figure 19. Pointwise Scalar Flux for Absorbing Block (a) PWL-LC (b) PWL-DFEM 125

Figure 20. Mesh and Materials of TRIGA pins (a) Mesh (b) Material Layout.....127

Figure 21. Pointwise Scalar Flux Solution for Group 1 in Pure Absorbing TRIGA pins (a) PWL-LC (b) PWL-DFEM 128

Figure 22. Pointwise Scalar Flux Solution for PWL-DFEM in Scattering TRIGA pins (a) Group 1 (b) Group 2.....130

Figure 23. Pointwise Scalar Flux Solution for PWL-LC in Scattering TRIGA pins (a) Group 1 (b) Group 2 131

Figure 24. Steady-State Cell-Average Scalar Fluxes for Beam in Pure Absorber (a) PWL-LC (b) PWL-DFEM 133

Figure 25. Steady-State Cell-Average Linear PWL-LC Scalar Flux Solution (a) Full (b) Line-Out.....135

Figure 26. Steady-State Pointwise Linear PWL-LC Scalar Flux Solution (a) Full (b) Line-Out.....137

Figure 27. Reactor Pin Cells Layout and Materials (a) z = 10 cm (b) y = -0.6 cm.....140

Figure 28. Pointwise Group 1 Scalar Flux Solution w/o Control Rod, at y = -0.6 cm (a) PWLD (b) PWL-LC.....142

Figure 29. S_8 Pointwise Group 1 Scalar Flux Solution w/o Control Rod, at y = -0.6 cm (a) PWLD (b) PWL-LC.....143

Figure 30. Pointwise Group 7 Scalar Flux Solution w/o Control Rod, at y = -0.6 cm (a) PWLD (b) PWL-LC.....144

Figure 31. Refined Pointwise Group 7 Scalar Flux Solution w/o Control Rod, at y = -0.6 cm (a) PWLD (b) PWL-LC 145

Figure 32. Pointwise Group 1 Scalar Flux Solution w/Control Rod, at y = -0.6 cm (a) PWLD (b) PWL-LC.....147

Page

Figure 33. Pointwise Group 7 Scalar Flux Solution w/Control Rod, at y = -0.6 cm (a) PWLD (b) PWL-LC.....148

Figure 34. Refined Pointwise Group 7 Scalar Flux Solution w/Control Rod, at y = -0.6 cm (a) PWLD (b) PWL-LC 149

Figure 35. Cell-Average Scalar Flux for Incident Beam in a Vacuum (time-dependent) using PWL-DFEM (a) $vt \in (7.0, 8.0)\,cm$ (b) $vt \in (24.0, 25.0)\,cm$ 151

Figure 36. Cell-Average Scalar Flux for Incident Beam in a Vacuum (time-dependent) using PWL-LC (a) $vt \in (7.0, 8.0)\,cm$ (b) $vt \in (24.0, 25.0)\,cm$ ...153

Figure 37. Fully-Implicit PWC-LC Cell-Average Scalar Flux for Beam in Pure Absorber with $\Delta t = 2.0\ \mu s$ (a) $t \in (0.0, 2.0)\,\mu s$ (b) $t \in (18.0, 20.0)\,\mu s$ 155

Figure 38. Fully-Implicit PWC-LC Cell-Average Scalar Flux for Beam in Pure Absorber with $\Delta t = 0.1\ \mu s$ (a) $t \in (1.9, 2.0)\,\mu s$ (b) $t \in (19.9, 20.0)\,\mu s$ 157

Figure 39. Fully-Implicit PWL-LC Cell-Average Scalar Flux for Beam in Pure Absorber with $\Delta t = 2.0\ \mu s$ (a) $t \in (0.0, 2.0)\,\mu s$ (b) $t \in (18.0, 20.0)\,\mu s$ 158

Figure 40. Fully-Implicit PWL-LC Cell-Average Scalar Flux for Beam in Pure Absorber with $\Delta t = 0.1\ \mu s$ (a) $t \in (1.9, 2.0)\,\mu s$ (b) $t \in (19.9, 20.0)\,\mu s$ 160

Figure 41. Cell-Average Scalar Flux for Incident Beam in a Vacuum (time-dependent) using PPWL-STLC (a) $vt \in (7.0, 8.0)\,cm$ (b) $vt \in (24.0, 25.0)\,cm$ 162

Figure 42. PWC-STLC Cell-Average Scalar Flux for Beam in Pure Absorber with $\Delta t = 2.0\ \mu s$ (a) $t \in (0.0, 2.0)\,\mu s$ (b) $t \in (18.0, 20.0)\,\mu s$ 164

Figure 43. PWC-STLC Cell-Average Scalar Flux for Beam in Pure Absorber with $\Delta t = 0.1\ \mu s$ (a) $t \in (1.9, 2.0)\,\mu s$ (b) $t \in (19.9, 20.0)\,\mu s$ 165

Page

Figure 44. PPWL-STLC Cell-Average Scalar Flux for Beam in Pure Absorber with $\Delta t = 2.0\ \mu s$ (a) $t \in (0.0, 2.0)\ \mu s$ (b) $t \in (18.0, 20.0)\ \mu s$ 167

Figure 45. PPWL-STLC Cell-Average Scalar Flux for Beam in Pure Absorber with $\Delta t = 0.1\ \mu s$ (a) $t \in (1.9, 2.0)\ \mu s$ (b) $t \in (19.9, 20.0)\ \mu s$ 169

Figure 46. Time-Dependent Cell-Average Linear PPWL-STLC Scalar Flux Solution (a) vt = 0.1 cm (b) vt = 1.9 cm 171

Figure 47. Time-Dependent Pointwise Linear PPWL-STLC Scalar Flux Solution (a) vt = 0.1 cm (b) vt = 1.9 cm.....172

Figure 48. Time-Dependent PPWL-STLC Cross-Section Scalar Flux Solution (a) Cell-Average at y = 1.1 cm (b) Pointwise at y = 1.0 cm 174

Figure 49. Time-Dependent Cell-Average Linear PPWL-STLC Scalar Flux Solution on Non-Orthogonal Grid (a) vt = 0.1 cm (b) vt = 1.9 cm 175

Figure 50. Time-Dependent Pointwise Linear PPWL-STLC Scalar Flux Solution (a) vt = 0.1 cm (b) vt = 1.9 cm.....176

Figure 51. Material and Geometry Layout for Top-Hat Problem 178

Figure 52. PPWL-STLC Cell-Average Radiation Temperature at T = 0.05 ns (a) Δt = 0.05 ns (b) Δt = 0.01 ns.....180

Figure 53. Fully-Implicit PWLD Cell-Average Radiation Temperature at T = 0.05 ns (a) Δt = 0.05 ns (b) Δt = 0.01 ns 182

Figure 54. TBDF2 PWLD Cell-Average Radiation Temperature at T = 0.05 ns (a) Δt = 0.05 ns (b) Δt = 0.01 ns.....183

Figure 55. Fully-Implicit PWLD Cell-Average Radiation Temperature at T = 0.05 ns with Δt = 0.002 ns.....184

Figure 56. PWLD Material Temperature for the Top-Hat Problem at t = 0.8 μs.....185

Figure 57. PPWL-STLC Material Temperature for the Top-Hat Problem at t = 0.0257 μs 186

Figure 58. Thick Diffusion Limit Test Problem Materials and Grid 187

Page

Figure 59. PWLD Steady-State Pointwise Scalar Flux for Thick Diffusion Limit Test Problem 188

Figure 60. PWC-STLC Cell-Average Scalar Flux at y = -0.63 cm for Various Times in Thick Diffusive Region 190

Figure 61. PWC-STLC Cell-Average Scalar Flux at y = -0.63 cm Compared to PPWL-STLC Cell-Average Scalar Flux in Thick Diffusive Region 190

Figure 62. PPWL-STLC Pointwise Scalar Flux Solution at Various Times for Thick Diffusive Problem 191

Figure 63. FI-PWLD and TBDF2-PWLD Pointwise Scalar Flux Solutions at Various Times for Thick Diffusive Problem 192

Figure 64. PWL-LC Steady-State Scalar Flux for Thick Diffusion Limit Test Problem 193

Figure 65. Steady-State Scalar Flux Contour Plot (a) PWL-LC (b) PWLD 194

Figure 66. PWL-LC "New" Method Steady-State Discontinuous Scalar Flux Contour Plot 197

Figure 67. Cell-Average Scalar Flux at t = 1.1023 μs for Midpoint of Cell 5 (a) PWC-STLC (b) PPWL-STLC 201

Figure 68. Problem-Integrated Scalar Flux at t = 1.1023 μs (a) PWC-STLC (b) PPWL-STLC 202

Figure 69. Track Layout for One Angular Direction with KBA Partitioning 206

LIST OF TABLES

Page

Table 1. Number of Unknowns to Form the Collision Source in a Cell 112

Table 2. 7-Group Energy Structure for Reactor Pin Cells 138

Table 3. B-10 Macroscopic Cross-Sections 139

Table 4. B-10 Scattering Matrix (macroscopic cross-sections (cm^{-1})) 139

Table 5. Approximate Total Angular Flux Unknowns in Reactor Pin Cell Problem ... 150

Table 6. Material Properties of Top-Hat Problem 179

Table 7. Thick Diffusion Limit Test Problem Properties 187

Table 8. Approximate Total Angular Flux Unknowns for One Time Step of the Time-Dependent Thick Diffusive Problem 199

1. INTRODUCTION

The process of particle transport can be modeled by a transport equation. Many problems involve particle transport coupled to other physical phenomena, as is the case in the fields of radiation hydrodynamics or nuclear reactor analysis. Some of the applications which can be modeled with the transport equation are nuclear reactors, medical imaging, photon interactions in atmospheric systems, industrial systems, and radiative transfer problems in astrophysics and inertial confinement fusion. Analytic solutions to most transport equation models of practical interest are not possible to find, therefore discrete approximations must be employed. The methods chosen to discretize the time and spatial variables of the problem domain are one of the most important aspects of solving a transport problem. Many spatial discretization methods have been applied to multidimensional problems including Finite Difference (FD), Finite Element (FE), and method of characteristics (MOC). The temporal variable has been discretized by using a variety of methods such as FD, Runge-Kutta methods, and the linear discontinuous (LD) method in time[1]. Current and past methods have discretized the spatial and time domains using different methods and then put them together to solve the transport equation. There have recently been explorations of MOC methods for time-dependent problems.[2-4]

A subset of the MOC is the method of long characteristics (LC), which has been applied to a variety of transport problems in multiple spatial dimensions in past research[5-14]. The basic LC methodology involves analytically finding the solution along parallel rays that pass through the problem domain for each angular direction, given

some approximate spatial distribution of the total source rate density. This method analytically inverts the streaming-plus-collision operator in the transport equation to find the angular flux along each ray in a steady-state problem. The LC method is a discrete ordinates method, in which a quadrature set is used to approximate the integrals over the direction variable that must be evaluated to form the source from particle-matter collisions.

1.1 Current State of the Problem

Solving the time-dependent transport equation is an important component of a wide range of applications. The one-group neutron transport equation is given in Eq. (1.1). The first term represents the time rate of change of the neutron population per unit phase-space volume. The second term represents the neutrons streaming through a volume of interest per unit phase-space volume. The third term is the total collision rate of neutrons per unit phase-space volume. Finally the last two terms represent the source of neutrons per unit phase-space volume from scattering and external sources. In the development of the methods in this dissertation, the source, referred to as the collisional source, includes the scattering plus fission source for neutrons or the scattering plus emission source for thermal radiation.

$$\begin{aligned}&\frac{1}{v}\frac{\partial\psi\left(\vec{r},\vec{\Omega},t\right)}{\partial t}+\vec{\Omega}\bullet\vec{\nabla}\psi\left(\vec{r},\vec{\Omega},t\right)+\sigma_t\left(\vec{r}\right)\psi\left(\vec{r},\vec{\Omega},t\right)\\&=\frac{1}{4\pi}\sigma_s\left(\vec{r}\right)\phi\left(\vec{r},t\right)+Q_{ext}\left(\vec{r},\vec{\Omega},t\right)\end{aligned}\tag{1.1}$$

For the rest of this dissertation it is assumed the angular variable is discretized with the discrete ordinates (DO) method[15]. One of the first steps when solving a DO

transport problem is to discretize the time and spatial domains. Besides the many known methods to discretize the spatial variable, such as DFEM and MOC, a variety of methods are also used to discretize the temporal variable. These time discretization methods include explicit methods such as Forward Euler, implicit methods such as Crank-Nicolson or Backward Euler, or combination methods including higher-order methods such as the trapezoidal backward-difference formula (TBDF2). All of these methods have properties that are undesirable including the introduction of a truncation error in the time derivative operator of the transport equation.

As mentioned previously, an important capability in many applications is to discretize the spatial variable for multidimensional problems. The LC method finds the desired angular solution of a transport equation along many parallel rays that pass through the problem domain for each of the discrete-ordinate quadrature directions. If the total source present in the problem can be approximated with a simple functional form, the angular solution along a ray can be found analytically. The scalar flux value (0^{th} angular moment), an important quantity of interest in many applications, for a cell is determined by the usual manner of a discrete-ordinates quadrature sum over directions coupled with an approximation of cell-wise spatial integrals using the solutions along the rays in the cell. Higher-order moments are found in a similar manner if needed. In most widely used reactor physics codes, the source approximation used is a cell-wise constant approximation, which requires only the spatial average of each angular moment.

Recent research has applied higher-order spatial source approximations using LC in multidimensional problems[16-20,48]. Several of these endeavors have explored using

linear and piece-wise linear source approximations in multiple spatial dimensions. One of these research papers also applies a least-squares approximation to find the source that is similar to what will be described in section 3. All of these current approaches apply LC to discretize only the spatial variable while applying other methods to discretize the temporal variable. One recent paper describes the difficulties that arise when applying a finite-difference method in time to a MOC[20].

None of the previously proposed MOC methods will obtain reasonable solutions in the thick diffusion limit unless all spatial cells have simple shapes such as triangles or tetrahedra. Since the thick diffusion limit is an important limit in many radiative transfer applications, it is necessary to use a source approximation with the same number of degrees of freedom as the cell shape that can perform well in this limit on arbitrary cell shapes. Based upon previous research, it has been determined that a piece-wise linear source approximation in space is accurate in highly diffusive problems on arbitrary polygons and polyhedra[21-24]. A main objective of the research presented here is to explore the use of such spatial approximations with LC methods.

The methods mentioned above, including MOC, have been applied to solve large and complex transport problems using parallel computing. The main parallel methods currently used to solve these problems are Block-Jacobi iterations, sweeps, or various combinations of these two methods. In general, Block-Jacobi iteration is performed by finding the local solution on fixed subdomains given previous iteration incoming angular flux information. The number of iterations will grow as the subdomains are refined in space. For a fixed problem size, Block-Jacobi will not scale asymptotically. Sweeps

follow a wavefront through the spatial domain for a given angular direction. A sweep-based method converges at a rate independent of subdomain size.

In order for these iteration techniques to be useful on today's parallel machines, they must be scalable to a very large number of processors. An algorithm is scalable if the solution time remains constant as the numbers of processors and unknowns in a problem are scaled by the same factor for very large processor counts, otherwise known as weak-scaling. The Block-Jacobi method and sweeps lack asymptotic scalability. Current Block-Jacobi methods do not scale because their iteration count increases as domain sizes decrease. Sweeps lack perfect scalability due to their inherent serial nature along the direction of propagation, which is roughly perpendicular to the sweep plane. This sweep plane is the plane across which work can be done in parallel at a given stage. The Koch-Baker-Alcouffe (KBA) algorithm for processor partitioning of structured grids has provided very good results for achieving a high degree of scalability using sweeps[25]. This algorithm decomposes the spatial domain into columnar partitions for each processor. This KBA algorithm with sweeps can scale to a large amount of processors before loss of parallel efficiency, especially if significant work is assigned to each processor. Currently, the sweeping algorithms developed by Hawkins, etc.[26] and Bailey and Falgout[35] provide the best published results for achieving a high degree of scalability with cell-based sweeps. Finding a sweep-based scalable parallel algorithm is an active area of ongoing research in the transport community.[26-31]

1.2 Focus of this Research

The main purpose of the proposed research is to extend the previously developed space-time long characteristic, STLC, method to transport problems in two spatial dimensions, (x, y, t), with a least-squares finite element approximation of the collision source. This STLC method analytically inverts the streaming-plus-collision operator of the transport equation including the time derivative. Therefore, the only approximation made in this STLC method is the piece-wise linear representation of the collisional source. Most previous MOC implementations use a cell-wise constant representation of the collisional source (usually only in the spatial dimension). Other higher-order polynomial approximations for the source using MOC methods have been investigated recently, but these methods do not consider the time variable.[16-20] This research explores a higher-order representation of the collisional source in space and time that is predicted to perform well in the thick diffusion limit on general polygons or polyhedra.[21-24] This work is particularly focused on prism cells in (x, y, t), so it specifically investigates a collisional source approximation that is piece-wise linear in the spatial variables and linear in time for each PWL basis function.

This work has several contributions including the development of several STLC methods for time-dependent two-dimensional transport problems on unstructured meshes in (x, y, t). These methods differ in their choices of function spaces in which the collisional source is approximated in each space-time cell. Steady-state versions have also been developed for (x, y) and (x, y, z). One uses PWL basis functions and is referred to as the piece-wise linear long characteristic (PWL-LC) method. Along with

the development of these various methods, a second contribution of this work is an analysis of how these methods satisfy a definition of conservation. A third set of contributions involve asymptotic analyses of how these methods perform in the thick diffusion limit. A final contribution of this work is showing the properties and testing the predictions of the previous analyses of these methods through implementation and testing in PDT, a parallel deterministic transport code being developed at Texas A&M University.

1.3 Outline of Remainder

The following sections discuss the newly developed methods for solving time-dependent and steady-state transport problems using LC. The following section, section 2, lays out the basic methodology of the steady-state PWL-LC method including coordinate system relations, least-squares definition, and issues regarding conservation of particles. Section 3 discusses STLC methodology with similar subsections as section 2 but adding the development of different source approximations. Section 4 includes an analysis of the PWL-LC and STLC methods in limits of interest including an asymptotic analysis in the thick diffusion limit. The discussion of the implementation of the LC equations in the PDT code is addressed in section 5. Numerical results showing key properties of the PWL-LC and STLC methods as well as a comparison to traditional discretization methods are presented in section 6. Section 7 discusses the main conclusions about the STLC and PWL-LC methods including the major contributions of this research. Finally, recommendations for further research, including new techniques for massively parallel transport sweeps, are discussed in section 8.

2. DEVELOPMENT OF THE PWL-LC METHOD

In this section the piece-wise linear long characteristic, PWL-LC, method is developed for (x, y) and (x, y, z) geometry for steady-state and time-dependent problems. This development includes showing the relation between coordinate systems in two and three dimensions as well as deriving the equations for the solution along a characteristic and the equations for generating the cell-wise collision source from the solution along these characteristic rays.

2.1 Coordinate System Definition in X-Y Geometry

The PWL-LC method solves two-dimensional steady-state transport problems by using long characteristics that span the (x, y) spatial domain. The spatial domain spanned by a two-dimensional or three-dimensional transport problem is usually discretized into cells. A characteristic that spans the entire problem domain is referred to as a *ray*, and the part of a ray intersecting one cell is referred to as a *track*. Figure 1 shows the discretization of the i^{th} cell with tracks passing through this cell for a particular angular direction.

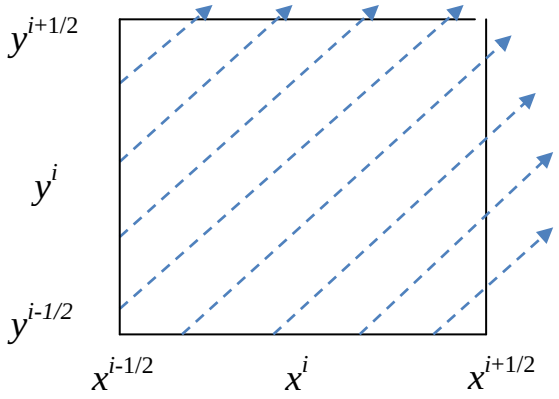


Figure 1. Spatial Cell in (x, y) with Tracks in One Angular Direction

In this (x, y) domain, the distance traveled by a particle is the distance in the x-y plane divided by the cosine of the angle between its trajectory and this plane (or the sine of the angle between the trajectory and the z-axis). The solution along a track will now quickly be reviewed. For a particle at position (x_0, y_0) and moving in direction $\underline{\Omega}$ with $\Omega_x = \mu$ and $\Omega_y = \eta$, the particle position after traversing a distance s is given by Eq. (2.1).

$$\begin{aligned} x &= x_0 + \mu s \\ y &= y_0 + \eta s \end{aligned} \tag{2.1}$$

The analytic solution of the transport equation along a track is given in Eq. (2.2). In this equation, the distance traveled along the k^{th} track from ($x-\mu_m\tau$, $y-\eta_m\tau$) to (x, y) is represented by τ and $q_{tot,m}$ is the total source for the m^{th} quadrature direction including the collisional source. Note that the collisional source is the scattering plus fission source for neutrons or the scattering plus emission source for thermal radiation.

$$\Psi_{m,k}(x,y) = \Psi^{in}_{m,k}(x-\mu_m\tau, y-\eta_m\tau)e^{-\tau\sigma_t} + \int_0^{\tau} ds\, q_{tot,m}(x-\mu_m s, y-\eta_m s)e^{-s\sigma_t} \tag{2.2}$$

In the PWL-LC method development that follows, integrals over the (x, y) cell are performed by using the track spacing and the solution along these tracks. Figure 2 shows the rotated coordinate system, (u, ω), defined for a given azimuthal angle, γ, that defines the particle trajectory in the (x, y) plane.

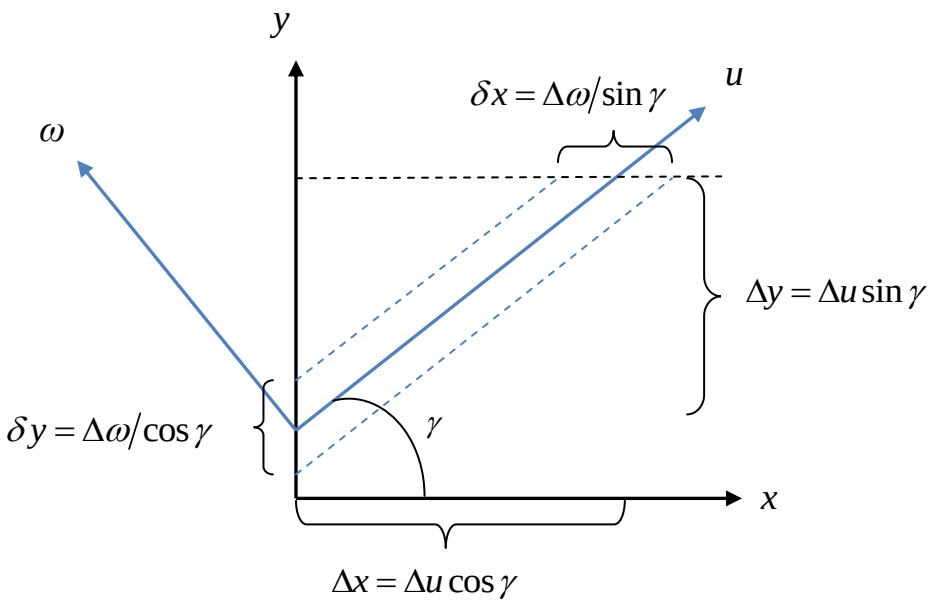


Figure 2. Rotated Coordinate System Relation in X-Y

Along with this azimuthal angle, a polar angle of the particle trajectory is also defined and given by θ, which is pointing out of the page in Figure 2. With these angular definitions, the direction cosines for a particle trajectory are defined in Eq. (2.3).

$$\begin{aligned} \mu &= \sin\theta\cos\gamma \\ \eta &= \sin\theta\sin\gamma \\ \xi &= \cos\theta \end{aligned} \tag{2.3}$$

Using the geometric relations shown in Figure 2 and the relations given in Eq. (2.1) and Eq. (2.3), the relationship between integrals over (x, y) and integrals over (u, ω) can be clearly defined. This relationship is given in Eq. (2.4). The Jacobian that arises in this relationship represents the projection of the area along the particle trajectory, s, into the x-y plane. The limits on u, ω, and s integrals are chosen to coincide with the cell boundaries.

$$\int_0^{\Delta y}\int_0^{\Delta x} dxdyf\left(x,y\right) = \int_{u_{\min}}^{u_{\max}}\int_{\omega_{\min}(u)}^{\omega_{\max}(u)} d\omega duf = \int_{s_{\min}}^{s_{\max}}\int_{\omega_{\min}(s)}^{\omega_{\max}(s)} d\omega ds\left|J\right| f = \int_{s_{\min}}^{s_{\max}}\int_{\omega_{\min}(s)}^{\omega_{\max}(s)} d\omega ds\sqrt{\mu_m^2+\eta_m^2}\, f \tag{2.4}$$

Equation (2.4) is the relation used to approximate integrals over the (x, y) cells in terms of track spacing, $\Delta\omega$, and integrals along these tracks. Also, as with all LC methods, the area represented by the sum of the tracks in a cell will not exactly equal the actual area of the (x, y) cell. Equation (2.5) shows the relationship between these areas.

$$\sum_{k=1}^{\#\text{ tracks in cell i}} \Delta\omega_{k,m}\Delta s_{k,m}\sqrt{\mu_m^2+\eta_m^2} \neq \Delta x_i \Delta y_i \tag{2.5}$$

As track spacing is refined, equality of these areas is approached.

2.2 PWL-LC Least-Squares Method

This section develops the least-squares approximation for finding average quantities from the transport equation as well as showing the construction of the scalar flux function that yields track-wise conservative reaction rates. The PWL-LC method will be developed here in two dimensions, (x, y). The analogous equations in three dimensions, (x, y, z), will also be presented following this development. A quadrature direction will be denoted with the subscript m during the following development.

This PWL-LC method seeks to produce a scalar flux function that yields conservation of particles and conservative reaction rates. The characteristic solution given in Eq. (2.2) satisfies a conservation equation, derived from the transport equation,

given in Eq. (2.6) along the k^{th} track. In this equation, *SS* indicates the function used in the collisional source and an over-bar denotes an average quantity over the track.

$$\Psi_{m,k}^{out} - \Psi_{m,k}^{in} + \sigma_t \Delta s_{k,m} \bar{\Psi}_{m,k} = \frac{1}{4\pi} \Delta s_{k,m} \left[\sigma_s \bar{\phi}_k^{SS} + \bar{Q}_{k,m} \right] \tag{2.6}$$

For simplicity, this method development assumes the collisional redistribution source is isotropic. It is straightforward to generalize to more complicated collisional source functions.

Multiplying Eq. (2.6) by the track spacing and the Jacobian from Eq. (2.4), and then summing over all tracks for all quadrature directions, the PWL-LC conservation statement for an (x, y) cell is found as given in Eq. (2.7). These five terms in Eq. (2.7) represent the numbers of particles that stream out of, stream into, have collisions in, scatter in, and are emitted by a fixed source in the cell, respectively.

$$\sum_{m=1}^{M} w_m \sqrt{\mu_m^2 + \eta_m^2}^{\#} \sum_{k=1}^{\text{tracks}} \Delta\omega_{k,m} \left\{ \begin{array}{l} \Psi_{m,k}^{out} - \Psi_{m,k}^{in} + \sigma_t \Delta s_{k,m} \bar{\Psi}_{m,k} \\ -\dfrac{\sigma_s \Delta s_{k,m}}{4\pi} \bar{\phi}_k^{SS} - \dfrac{\Delta s_{k,m}}{4\pi} \bar{Q}_{k,m} \end{array} \right\} = 0 \tag{2.7}$$

For a method to be conservative, the scattering term must be equal to σ_s / σ_t multiplied by the collision term, which means Eq. (2.8) must be true.

$$\begin{aligned} &\sum_{m=1}^{M} w_m \sqrt{\mu_m^2 + \eta_m^2}^{\#} \sum_{k}^{\text{tracks}} \Delta\omega_{k,m} \Delta s_{k,m} \bar{\Psi}_{m,k} = \\ &\sum_{m=1}^{M} w_m \sqrt{\mu_m^2 + \eta_m^2}^{\#} \sum_{k}^{\text{tracks}} \Delta\omega_{k,m} \frac{\Delta s_{k,m}}{4\pi} \bar{\phi}_k^{SS} \end{aligned} \tag{2.8}$$

Equation (2.8) constrains the scalar flux function that is used to generate the scattering source. For a method that approximates the source as a constant in each cell, also known

as a piece-wise constant (PWC) method, the constraint given by Eq. (2.8) completely specifies the function used for ϕ^{SS}. This function is a constant given by a directional average (sum over angles) of the spatial average of the function over the track-based approximate volume of a cell (sum over tracks with track spacing, track length, and Jacobian) as shown in Eq. (2.8). The LC method that approximates the collisional source as a constant in each cell will be referred to as the PWC-LC method. Results from this method will be presented in the results section for comparison to the PWL-LC method.

The PWL-LC method seeks to represent the scattering source as a piece-wise linear function of *x* and *y* in each cell based upon past research that has shown these basis functions are designed to achieve excellent behavior in the thick diffusion limit for polygonal and polyhedral spatial cells[21,24,32]. This scattering source can be represented as shown in Eq. (2.9).

$$\phi_{cell}^{SS}(x,y) = \sum_{j=1}^{N} \phi_{cell,j}^{SS} b_{cell,j}(x,y) \tag{2.9}$$

In this equation, *N* is the number of basis (or interpolation) functions in the cell, which for the PWL-LC method is the number of cell vertices. Each basis function, *b*, in this equation is a piece-wise linear function of *x* and *y*.[22,23,32] Figure 3 illustrates a PWL basis function of a two-dimensional polygonal cell. The pentagonal cell in this figure has vertices v_1-v_5. Each PWL function is linear on each triangular subcell indicated by dotted lines in Figure 3. The j^{th} function has a value of unity at vertex v_j and zero at other vertices. Its value at the cell center, *z*, depends on the choice of center point.[22]

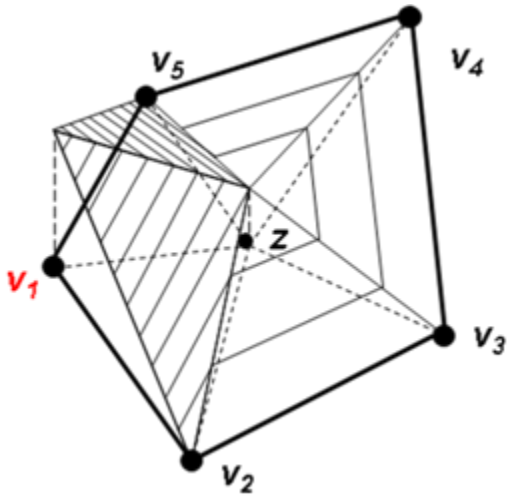


Figure 3. Contour Sketch of PWL Basis Function on Pentagonal Cell

The coefficients given in Eq. (2.9) are determined by satisfying *N* constraints similar to Eq. (2.8). Equation (2.8) arose from the 0th spatial moment of the direction-integrated transport equation. Replacing this one constraint by *N* spatial moments, the *N* constraints needed are found. The conservation equation for each track is multiplied by each of the *N* PWL basis functions and by the track width and associated Jacobian before summing over all tracks and all directions. Performing this operation produces the set of equations given in Eq. (2.10) needed to solve for the required coefficients of the collisional source.

$$\sum_{m=1}^{M} w_m \sqrt{\eta_m^2 + \mu_m^2} \sum_{k=1}^{\#\text{ tracks}} \Delta\omega_{k,m} \int_0^{\Delta s_{k,m}} ds b_i(s) \left[\Psi_{m,k}(s) - \frac{1}{4\pi} \sum_{j=1}^{N} \phi_{cell,j}^{SS} b_{cell,j}(s) \right] = 0,\ i = 1..N \quad (2.10)$$

Equation (2.10) is equivalent to a least-squares determination of the collisional scalar flux function by minimizing the squared difference between this scalar flux function and the LC solution when integrated along each track and summed over all tracks for all directions in a cell. Also note that since the PWL basis functions sum to unity, the sum of all equations in Eq. (2.10) reproduces Eq. (2.8).

A traditional cell-average scalar flux can be defined such that when multiplied by the cell area and a cross section for a given reaction, it gives the same reaction rate as is given by summing over track-wise reaction rates. Following from the equations above, this cell-averaged flux is given by Eq. (2.11).

$$\bar{\Phi}_{cell} A_{cell} = \sum_{m=1}^{\mathrm{M}} w_m \sqrt{\mu_m^2 + \eta_m^2}^{\#} \sum_{k=1}^{\text{tracks}} \Delta\omega_{k,m} \Delta s_{k,m} \bar{\Psi}_{m,k} \tag{2.11}$$

In general, this traditional cell-averaged flux will not equal the average of the ϕ^{SS} function. This difference is due to the fact that the track-based "cell area" does not equal the true cell area except in the limit of fine track spacing. For any problem that has a constant solution, the ϕ^{SS} function will be constant and correct. But the traditional average flux in a cell (as defined in Eq. (2.11)) will depend on the layout of the tracks in the cell and will generally vary from cell to cell.

This least-squares PWL-LC solution is smoothly varying for problems in which the solution should be smoothly varying. It is also generally conservative locally and globally. Since this least-squares solution is found by enforcing the conservation statement of Eq. (2.8), it must satisfy conservation in a local sense. This definition of local conservation is taken on the track-based cell area, which is different for each angle.

Therefore, a summation over angles is a weighted summation over different track-based areas. The reaction rates in this track-based area are found by using the least-squares solution. The inflow and outflow along each track-based cell surface (edge of the track-based cell) can be cleanly defined by using the inflow/outflow from each track area on a surface and projecting onto the basis function space on this surface. Note that these cell surfaces are different for each angle, but, again, a summation over angles is just a weighted summation over different track-based surface areas. With the reaction rates, inflow, and outflow from a track-based cell defined in this manner, this least-squares function satisfies local conservation as given by its definition. Finally, since each track-based cell has local conservation and these areas do not overlap, all of the track-based cells in the problem domain can be added together to satisfy global conservation. A more detailed explanation of the properties of the least-squares flux using a simple constant solution example is given in section 4.1.

2.3 Time-Dependent PWL-LC

One method for solving time-dependent transport problems is applying a finite-difference (FD) method in time to the PWL-LC method. Depending on the details of how it is formulated, a FD discretization of the time variable with an MOC method may produce undesirable properties[4].

A finite-difference method in time in which the time-step averaged angular flux is approximated as a weighted average of the flux at the beginning and end of the time step is given in Eq. (2.12). The θ variable can have a value from zero to one, with the resulting method ranging from fully explicit to fully implicit time differencing.

$$\begin{aligned}&\psi_m^{n+1/2}(x,y)=\theta\psi_m^{n+1}(x,y)+(1-\theta)\psi_m^{n}(x,y)\\&\Rightarrow\frac{\psi_m^{n+1}-\psi_m^{n}}{v\Delta t}=\frac{\psi_m^{n+1/2}-\psi_m^{n}}{\theta v\Delta t}\end{aligned}\tag{2.12}$$

This time approximation leads to a transport equation for the time-step averaged angular flux in which the total cross section and fixed source are augmented as given in Eqs. (2.13) and (2.14).

$$\sigma_t\rightarrow\sigma_t+\frac{1}{v\theta\Delta t}\tag{2.13}$$

$$q_{fixed,m}(x,y)\rightarrow q_{fixed,m}(x,y)+\frac{\psi_m^{n}(x,y)}{v\theta\Delta t}\tag{2.14}$$

The time-dependent transport equation given in Eq. (1.1) can be rewritten for the k^{th} track with these augmentations and LC spatial discretization as given in Eq. (2.15),

$$\begin{aligned}&\frac{d\Psi_{m,k}^{n+1/2}}{ds}+\sigma_t\left[1+\frac{1}{\varepsilon}\right]\Psi_{m,k}^{n+1/2}(s)=\\&\frac{1}{4\pi}\left[\sigma_s\phi_k^{SS,n+1/2}(s)+Q_k^{n+1/2}(s)\right]+\frac{\sigma_t}{\varepsilon}\psi_{m,k}^{n}(s)\end{aligned},\tag{2.15}$$

where

$$\frac{1}{\varepsilon}\equiv\frac{1}{\sigma_t v\theta\Delta t}.\tag{2.16}$$

The first challenge with this discretization is how to represent the time-*n* angular flux. Previous research has explored using different values of theta to approximate the angular flux on each track[4]. The memory requirements for this approach are high and the authors reported difficulties in accuracy and numerical stability[4]. The time-dependent PWL-LC method presented here seeks to approximate the time-*n* angular

fluxes as cell-wise PWL expansions. Therefore, the coefficients of this PWL expansion for the time-averaged flux in the m^{th} quadrature direction are determined by a least-squares projection in each cell as given in Eq. (2.17).

$$\sum_{k=1}^{\#\text{ tracks}} \Delta\omega_{k,m}\sqrt{\mu_m^2+\eta_m^2}\int_0^{\Delta s_{k,m}} ds b_i(s)\left[\Psi_{m,k}(s)-\sum_{j=1}^{N}\psi_{m,j}^{n+1/2}b_j(s)\right]=0,\ i=1..N \quad (2.17)$$

Using the basis-function expansions above, the finite-difference approximation in time holds for each basis-function coefficient in each cell as given in Eq. (2.18).

$$\psi_{m,j}^{n+1/2}=\theta\psi_{m,j}^{n+1}+(1-\theta)\psi_{m,j}^{n}\Rightarrow\psi_{m,j}^{n+1}=\frac{1}{\theta}\left[\psi_{m,j}^{n+1/2}-(1-\theta)\psi_{m,j}^{n}\right],\quad j=1..N \quad (2.18)$$

The solution of a time-dependent transport problem using this method proceeds as follows. For a given time step, the time-*n* angular flux is expressed as a PWL expansion on the right hand side of Eq. (2.15) and this equation is solved analytically along each track via the usual LC procedure described in the previous section with altered total cross section as given in Eq. (2.13). Then, Eq. (2.17) is employed to generate the PWL expansion coefficients for the time-averaged flux in each direction in each cell. Finally, Eq. (2.18) is solved for the time-(*n+1*) angular fluxes and the calculation proceeds to the next time step.

There is a drawback to this time-dependent method. The projection onto PWL, or any other, cell-wise basis functions amounts to a redistribution of particles within each cell at the end of each time step. In the limit of large time steps, which means a large ε in Eq. (2.15), this has little effect on the solution. But in the limit of small time steps (small ε) this redistribution can dominate the solution, which removes the

advantages (more accurate streaming treatment) that PWL-LC has over the related PWL-DFEM. A less than desirable consequence is that if a steady-state problem is solved, the steady-state solution that is obtained varies with the size of the time step. The solution approaches the steady-state PWL-LC solution for large-Δt and a steady-state PWL-DFEM solution for small-Δt. These limits will be shown below.

The large-Δt limit is obvious but the small-Δt limit may not be as obvious. A brief analysis of the time-dependent PWL-LC method in the limit of small time steps will show this steady-state limit. For simplicity, this analysis considers a time-dependent problem that reaches a steady state and this steady solution is examined. For a track, the time-dependent PWL-LC solution in the steady-state limit satisfies Eq. (2.19).

$$\frac{d\Psi_{m,k}}{ds}+\sigma_t\left[1+\frac{1}{\varepsilon}\right]\Psi_{m,k}(s)=\frac{1}{4\pi}\left[\sigma_s\phi_k^{SS}(s)+Q_{m,k}(s)\right]+\frac{\sigma_t}{\varepsilon}\psi_{m,k}^{LC\to b}(s) \quad (2.19)$$

In Eq. (2.19), $\psi_{m,k}^{LC\to b}$ is the projection of the LC solution onto the cell-wise basis functions defined by the expression given in Eq. (2.20):

$$\sum_{k=1}^{\#\text{ tracks}}\Delta\omega_{k,m}\sqrt{\mu_m^2+\eta_m^2}\int_0^{\Delta s_{k,m}}ds b_i(s)\left[\begin{array}{l}\Psi_{m,k}(s)-\\ \sum_{j=1}^{N}\psi_{m,j}^{LC\to b}b_j(s)\end{array}\right]=0,\ i=1..N, \quad (2.20)$$

where

$$\psi_{m,k}^{LC\to b}=\sum_{j=1}^{N}\psi_{m,j}^{LC\to b}b_j(s). \quad (2.21)$$

This projection can be written in operator notation as given in Eq. (2.22), where P^{LC} is the least-squares projection operator defined by Eq. (2.20) and Ψ_m is the set of analytic solutions along all tracks in direction *m* in the given cell. Note that this operator includes the summation over the track area in a cell (which approaches the cell volume as track spacing is refined).

$$\psi_m^{LC\to b}(x,y) \equiv \sum_{j=1}^{N} \psi_{m,j}^{LC\to b} b_j(x,y) = P^{LC}\Psi_m \tag{2.22}$$

Consider an asymptotic expansion of the unknowns in the parameter, ε, which becomes small as the time step becomes small. The expansion of the unknowns is given in Eqs. (2.23)-(2.25).

$$\Psi_{m,k}(s) = \Psi_{m,k}^{(0)}(s) + \varepsilon\,\Psi_{m,k}^{(1)}(s) + \varepsilon^2\Psi_{m,k}^{(2)}(s) + \tag{2.23}$$

$$\phi^{SS}(x,y) = \phi^{SS(0)}(x,y) + \varepsilon\,\phi^{SS(1)}(x,y) + \varepsilon^2\phi^{SS(2)}(x,y) + \ldots \tag{2.24}$$

$$\psi_{m,j}^{LC\to b} = \psi_{m,j}^{LC\to b(0)} + \varepsilon\,\psi_{m,j}^{LC\to b(1)} + \varepsilon^2\,\psi_{m,j}^{LC\to b(2)} + \ldots \tag{2.25}$$

These expansions are inserted into Eq. (2.19) and it is required that this relation holds for each power of ε as ε gets small. From this expansion it is easily seen that the $O(1/\varepsilon)$ terms yield the expression given in Eq. (2.26).

$$\Psi_{m,k}^{(0)}(s) = \psi_{m,k}^{LC\to b(0)}\left(x(s), y(s)\right) \tag{2.26}$$

This result means that the LC solution equals its basis function projection solution to leading order as the time step shrinks. Next consider the $O(1)$ terms from the expanded transport equation. These terms are shown in Eq. (2.27).

$$\frac{d\Psi_{m,k}^{(0)}}{ds}+\sigma_t\Psi_{m,k}^{(0)}(s)+\sigma_t\Psi_{m,k}^{(1)}(s)= \frac{1}{4\pi}\left[\sigma_s\phi_k^{SS(0)}(s)+Q_{m,k}(s)\right]+\sigma_t\psi_{m,k}^{LC\to b(1)}(s) \tag{2.27}$$

This equation is rearranged and the projection operator, P^{LC}, is applied as well as the result of Eq. (2.26). Therefore, Eq. (2.27) now becomes the relation shown in Eq. (2.28).

$$P^{LC}\left\{\frac{d\psi_{m,k}^{LC\to b(0)}}{ds}+\sigma_t\psi_{m,k}^{LC\to b(0)}-\frac{1}{4\pi}\left[\sigma_s\phi_k^{SS(0)}+Q_{m,k}\right]\right\}= \sigma_t P^{LC}\left\{\Psi_{m,k}^{(1)}-\psi_{m,k}^{LC\to b(1)}\right\} \tag{2.28}$$

The right hand side of Eq. (2.28) is equal to zero because $P^{LC}\Psi_{m,k}^{(1)}\equiv\psi_{m,k}^{LC\to b(1)}$ and $P^{LC}\psi_m^{LC\to b(1)}=\psi_m^{LC\to b(1)}$. So Eq. (2.28) says that the leading-order basis function expansion, which equals the leading-order LC solution, satisfies a projected transport equation with the projection operator P^{LC}.

This PWL-LC result for small time steps is now compared to a standard DFEM solution that uses the same basis functions (PWL) and time differencing. The PWLD method projects the residual of the solution to zero. Equation (2.29) defines the PWLD angular flux solution in operator notation along with an "upwinding" equation for the flux on each cell edge[23].

$$P^{DFEM}\left\{\frac{d\psi_m^{DFEM}}{ds}+\sigma_t\psi_m^{DFEM}-\frac{1}{4\pi}\left[\sigma_s\phi^{DFEM}+Q_m\right]\right\}=0 \tag{2.29}$$

Both the DFEM and leading-order LC solutions live in the same function space because they use the same basis functions. The equations that determine the coefficients in the two basis function expansions differ only to the degree that the two projection operators, P^{LC} and P^{DFEM}, differ. The difference between these two operators is from the fact that the LC projection operator integrates over a collection of track lengths and widths that approximate the cell volume, while the DFEM operator integrates over the exact cell volume. Therefore it can be concluded that the steady-state LC solution obtained by the finite-difference method discussed above will approach a discontinuous finite-element solution in the limit of small time steps, and this solution differs from the DFEM solution only to the usually small degree that the two projection operators differ. It should be noted that this analysis can also be applied to the PWC-LC method using the projected finite-differencing scheme with the PWC basis in a cell. It is found that the steady-state PWC-LC solution obtained by this projected finite-difference method will approach a PWC-DFEM solution in the limit of small time steps, and this solution differs from the DFEM solution only to the usually small degree that the two projection operators differ.

In summary, applying the simple finite-difference approach for the time discretization outlined above to the PWL-LC method will yield an LC solution that is "between" a finite-differenced PWL-DFEM solution and the solution what would be obtained from a full STLC treatment of the spatial and time variables. This LC solution will approach a DFEM solution in the limit of small time steps and a full LC solution in the limit of large time steps. Numerical results given in section 6 support this theoretical

result. The fact that this finite-differenced LC solution depends on the time step in steady-state is motivation to study and develop the STLC method.

2.4 Coordinate System Definition in X-Y-Z Geometry

The PWL-LC can also be applied to steady-state transport problems in three dimensions in the (x, y, z) coordinate system. The same definition of rays and tracks as defined previously applies in this geometry. Similar to two dimensions, for a particle at position (x_0, y_0, z_0) and moving in direction $\underline{\Omega}$ with $\Omega_x = \mu$, $\Omega_y = \eta$, and $\Omega_z = \xi$, the particle position after traversing a distance s is given by Eq. (2.30).

$$\begin{aligned} x &= x_0 + \mu s \\ y &= y_0 + \eta s \\ z &= z_0 + \xi s \end{aligned} \tag{2.30}$$

The analytic solution along a track, k, in three dimensions is given in Eq. (2.31). Similar to Eq. (2.2), the distance traveled along the k^{th} track from (x–$\mu_m\tau$, y–$\eta_m\tau$, z-$\xi_m\tau$) to (x, y, z) is represented by τ and $q_{tot,m}$ is the total source for the m^{th} quadrature direction including the collisional source.

$$\begin{aligned} \Psi_{m,k}(x,y) &= \Psi^{in}_{m,k}(x-\mu_m\tau, y-\eta_m\tau, z-\xi_m\tau)e^{-\tau\sigma_t} \\ &+\int_0^{\tau} ds\, q_{tot,m}(x-\mu_m s, y-\eta_m s, z-\xi_m s)e^{-s\sigma_t} \end{aligned} \tag{2.31}$$

For the (x, y, z) PWL-LC method development that follows, integrals over the (x, y, z) cell are performed by using two track-spacing values and the solution along these tracks. Figure 4 shows the rotated orthogonal coordinate system, (u, ω, s), defined for a given azimuthal angle, γ, and polar angle, θ, that define the particle trajectory in the (x, y,

z) volume. The particle trajectory is along the *s*-axis which is pointing in direction $\underline{\Omega}$. The direction cosines of the particle trajectory are defined as given in Eq. (2.3).

The track spacing is defined by a $\Delta\omega$ and Δu in the rotated coordinate system. Using the geometric relations shown in Figure 4 and the relations given in Eq. (2.30) and Eq. (2.3), the relationship between integrals over (*x*, *y*, *z*) and integrals over (*u*, ω, *s*) can be defined, similar to the (*x*, *y*) integrals previously defined. This relationship is given in Eq. (2.32).

$$\int_0^{\Delta z}\int_0^{\Delta y}\int_0^{\Delta x} dxdydzf\left(x,y\right) = \int_{s_{\min}}^{s_{\max}}\int_{u_{\min}(s)}^{u_{\max}(s)}\int_{\omega_{\min}(s)}^{\omega_{\max}(s)} d\omega dudsf \tag{2.32}$$

The limits on the *u*, ω, and *s* integrals are chosen to coincide with the cell boundaries. Note that the Jacobian in this relation is equal to unity since the particle trajectory is in the (*x*, *y*, *z*) domain.

Equation (2.32) is the relation used to approximate integrals over the (*x*, *y*, *z*) cells in terms of track spacing, $\Delta\omega$ and Δu, and integrals along these tracks for the PWL-LC method. Again similar to two dimensions, the volume represented by the sum of the tracks in a cell will not exactly equal the volume of the (*x*, *y*, *z*) cell as shown in Eq. (2.33).

$$\sum_{k=1}^{\#\text{ tracks in cell}} \Delta\omega_{k,m}\Delta u_{k,m}\Delta s_{k,m} \neq \Delta x\Delta y\Delta z \tag{2.33}$$

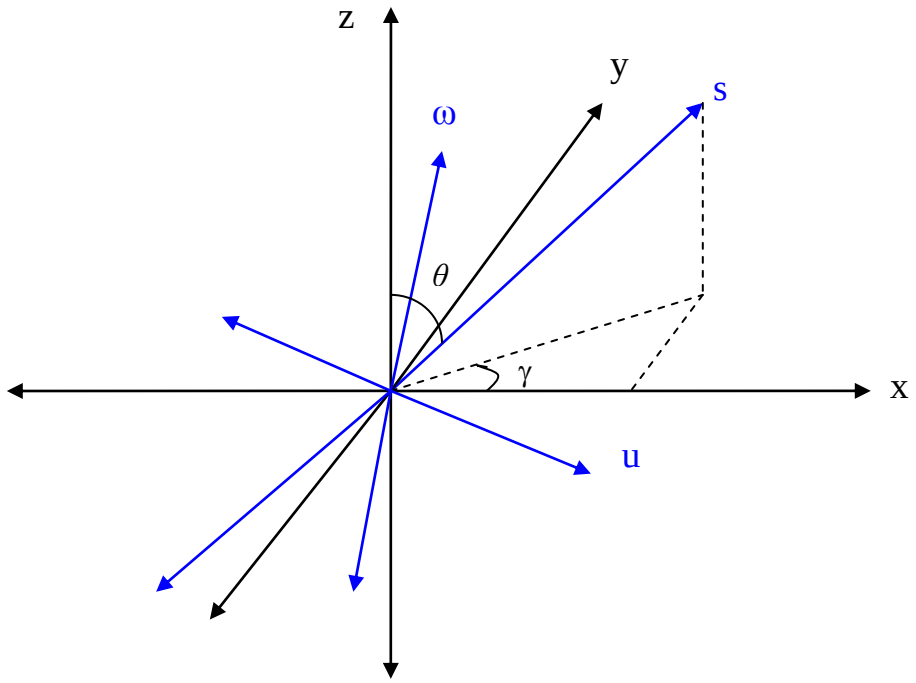


Figure 4. Rotated Coordinate System Relation in X-Y-Z

2.5 PWL-LC Method in X-Y-Z

The equations developed above for two dimensions can also be applied for the PWL-LC method in (x, y, z). The area integrals become volume integrals that include two track-spacings as specified in Section 2.4. Therefore, the Jacobian included in all equations is different based upon the coordinate relations given in Eq. (2.32). This Jacobian and track summation given in the previous equations are replaced as shown in Eq. (2.34).

$$\sum_{m=1}^{M} w_m \sqrt{\mu_m^2 + \eta_m^2} \sum_{k=1}^{\#\text{ tracks}} \Delta\omega_{k,m} \int_0^{\Delta s_{k,m}} ds\, f\left(s(x,y)\right) \rightarrow$$
$$\sum_{m=1}^{M} w_m \sum_{k=1}^{\#\text{ tracks}} \Delta\omega_{k,m} \Delta u_{k,m} \int_0^{\Delta s_{k,m}} ds\, f\left(s(x,y,z)\right) \tag{2.34}$$

The PWL basis functions are now applied on polyhedral cells; therefore the triangular subcells of Figure 4 become tetrahedral subcells. The same least-squares procedure to find the coefficients of the collision source is still applied.

The PWL-LC conservation statement for an (*x*, *y*, *z*) cell is given in Eq. (2.35), similar to Eq. (2.7).

$$\sum_{m=1}^{M} w_m \sum_{k=1}^{\#\text{ tracks}} \Delta\omega_{k,m} \Delta u_{k,m} \left\{ \begin{array}{l} \Psi_{m,k}^{out} - \Psi_{m,k}^{in} + \sigma_t \Delta s_{k,m} \bar{\Psi}_{m,k} \\ -\dfrac{\sigma_s \Delta s_{k,m}}{4\pi} \bar{\phi}_k^{SS} - \dfrac{\Delta s_{k,m}}{4\pi} \bar{Q}_{k,m} \end{array} \right\} = 0 \tag{2.35}$$

From this equation, the constraint on the scalar flux function is determined, similar to Eq. (2.8), as given in Eq. (2.36).

$$\sum_{m=1}^{M} w_m \sum_{k}^{\#\text{ tracks}} \Delta\omega_{k,m} \Delta u_{k,m} \Delta s_{k,m} \bar{\Psi}_{m,k} = \sum_{m=1}^{M} w_m \sum_{k}^{\#\text{ tracks}} \Delta\omega_{k,m} \Delta u_{k,m} \frac{\Delta s_{k,m}}{4\pi} \bar{\phi}_k^{SS} \tag{2.36}$$

The functional form of the collisional source now uses PWL basis functions defined in three-dimensional space and is written as shown in Eq. (2.37), where *N* is equal to the number of vertices of the polyhedral cell.

$$\phi_{cell}^{SS}(x,y,z) = \sum_{j=1}^{N} \phi_{cell,j}^{SS} b_{cell,j}^{PWL}(x,y,z) \tag{2.37}$$

With this functional form, the least-squares approach to finding the coefficients of this expansion, similar to Eq. (2.10), can be written in three dimensions as given in Eq. (2.38).

$$\sum_{m=1}^{M} w_m \sum_{k=1}^{\#\text{ tracks}} \Delta\omega_{k,m}\Delta u_{k,m} \int_0^{\Delta s_{k,m}} ds b_i(s) \left[\begin{array}{l} \Psi_{m,k}(s) - \\ \dfrac{1}{4\pi} \sum_{j=1}^{N} \phi_{cell,j}^{SS} b_{cell,j}(s) \end{array} \right] = 0,\ i = 1..N \quad (2.38)$$

A definition of an average scalar flux on a cell can also be defined in (x, y, z) as given in Eq. (2.39), which is similar to the flux defined in Eq. (2.11), such that when multiplied by the cell volume and a cross section for a given reaction, it gives the same reaction rate as is given by summing over track-wise reaction rates.

$$\bar{\Phi}_{cell}^{cons} V_{cell} = \sum_{m=1}^{\mathrm{M}} w_m \sum_{k=1}^{\#\text{ tracks}} \Delta\omega_{k,m}\Delta u_{k,m}\Delta s_{k,m}\bar{\Psi}_{m,k} \quad (2.39)$$

Again, the difference between this cell-averaged scalar flux and the average of the ϕ^{SS} function is due to the track-based "cell volume" not being equal to the true cell volume except in the limit of very fine track spacing.

This PWL-LC method in (x, y, z) is still generally locally conservative, produces the analytic solution along each track, and produces smoothly varying reaction rates for a problem in which the reaction rates are smoothly varying. The PWL-LC method in three-dimensions, (x, y, z), can also be applied to time-dependent transport problems via the finite-differencing scheme outlined in section 2.3. The results of the analysis from this previous section still apply in three dimensions.

3. DEVELOPMENT OF THE STLC METHOD

This section develops the STLC method for time-dependent problems in (x, y, t) space. First, a brief review of the previously developed STLC method in (x, t) is presented. The equations relating the (x, y, t) coordinate system to a rotated coordinate system are then defined. Also, this section derives the equations defining the solution along a space-time characteristic and the equations for generating the cell-wise collision source from the solution along these characteristic rays for various collision-source approximations.

3.1 Review of Coordinates and STLC Method in X-vT

This section reviews the methodology and properties of the previously developed STLC method in one dimension, (x, t)[2,3]. Figure 5 shows the STLC discretization of the $(i,n)^{th}$ cell in the (x, t) domain, with tracks passing through this cell for a particular angular direction.

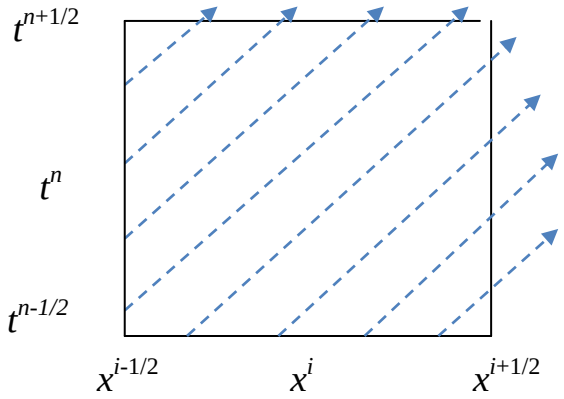


Figure 5. Space-time Cell with Tracks in One Angular Direction

In the space-time cell shown in Figure 5, neither the tracks nor the spacing between them has units of length. It is easier to consider a cell with the vertical coordinate equal to *vt*, where *v* is the speed of the particles, so each dimension has units of length. In this (*x*, *vt*) system, the distance traveled by a particle along a characteristic is simply the change in its *vt* coordinate.

A particle that is initially at a position x_0 at time t_0 moving with a direction cosine μ and speed *v*, has the coordinates given in Eq. (3.1) after moving a distance *s*.

$$\begin{aligned} x &= x_0 + \mu s \\ t &= t_0 + \frac{s}{v} \end{aligned} \tag{3.1}$$

The analytic transport solution along a characteristic track, *k*, in this space-time domain is given in Eq. (3.2).

$$\Psi_{m,k}(x,t) = \Psi^{in}_{m,k}(x-\mu_m\tau, t-\frac{\tau}{v})e^{-\tau\sigma_t} + \int_0^{\tau} dsq_{tot,m}(x-\mu_m s, t-\frac{s}{v})e^{-s\sigma_t} \tag{3.2}$$

The distance traveled along this characteristic from ($x-\mu_m\tau$, $t-\tau/v$) to (*x*, *t*) is defined as τ, and q_{tot} is the total source including the collisional source.

A rotated coordinate system is defined in this (*x*, *vt*) system in order to facilitate the integrals over each space-time cell during the solution process. Figure 6 shows this rotated coordinate system in the *x*-*vt* plane, very similar to Figure 2. In Figure 6, the angle θ is related to μ by the relation given in Eq. (3.3).

$$\sin\theta = \frac{1}{\sqrt{1+\mu^2}} \tag{3.3}$$

From this equation and Figure 6, the relationship between the actual distance traveled by a particle and the distance along the *u*-axis is defined in Eq. (3.4).

$$\Delta u = \Delta s / \sin\theta = \Delta s\sqrt{\mu^2 + 1} \tag{3.4}$$

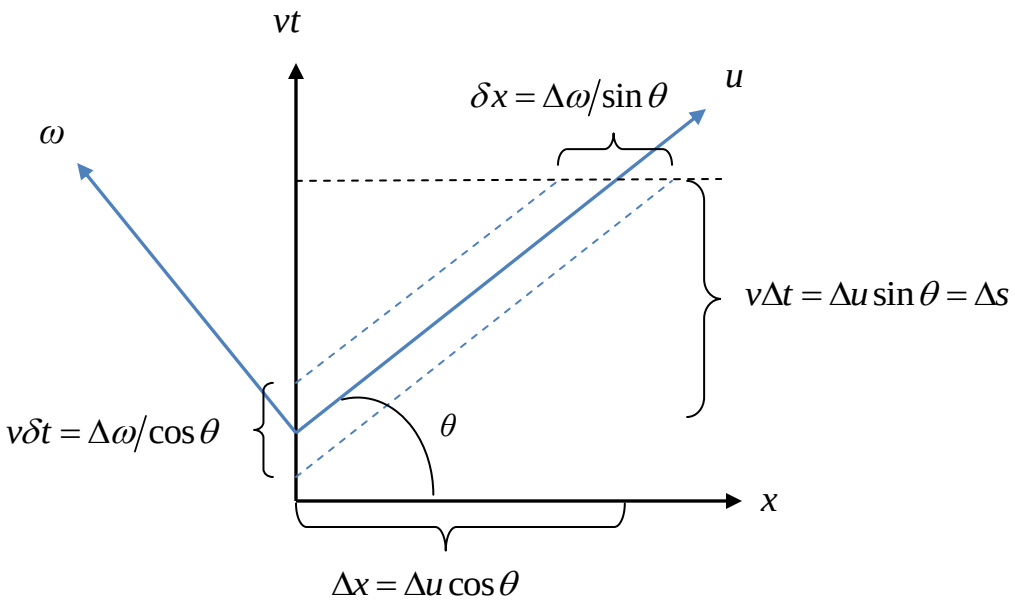


Figure 6. Rotated (*u*-*ω*) Coordinate System in *x*-*vt* Plane

Also from Figure 6 and the previous two relations, the relationship between the integrals over (*x*, v*t*) and integrals over the rotated system (*u*, *ω*) can be determined as given in Eq. (3.5).

$$\int_0^{v\Delta t}\int_0^{\Delta x} dx\,d(vt)\,f(x,t) = \int_{\omega_{\min}}^{\omega_{\max}}\int_{u_{\min}(\omega)}^{u_{\max}(\omega)} du\,d\omega\,f = \int_{s_{\min}}^{s_{\max}}\int_{\omega_{\min}(s)}^{\omega_{\max}(s)} d\omega\,ds\sqrt{1+\mu_m^2}\,f \tag{3.5}$$

This relation shows the term (Jacobian) that arises when converting to an integral over (u, ω), which will be present in all of the track area integrals below.

The characteristic solution given in Eq. (3.2) satisfies a conservation equation along a track as given in Eq. (2.6). A statement of conservation for the STLC method on an (x, t) cell is defined by multiplying Eq. (2.6) by the track width, $\Delta\omega_k$, and by $(1+\mu^2)^{1/2}$, as well as dividing by the speed. Summing over all tracks for all directions in the cell produces the conservation statement given in Eq. (3.6).

$$\sum_{m=1}^{M} w_m \sqrt{1+\mu_m^2} \sum_{k=1}^{\#\text{ of tracks}} \frac{\Delta\omega_{k,m}}{v} \left\{ \begin{array}{l} \Psi_{m,k}^{out} - \Psi_{m,k}^{in} + \sigma_t \Delta s_{k,m} \bar{\Psi}_{m,k} - \\ \frac{\sigma_s \Delta s_{k,m}}{2} \bar{\phi}_k^{SS} - \frac{\Delta s_{k,m}}{2} \bar{Q}_{k,m} \end{array} \right\} = 0 \qquad (3.6)$$

Again note that the *SS* in this equation represents the functional form used for the collisional source in the cell. The five terms in this equation represent the rates at which particles stream out of, stream into, have collisions in, scatter in, and are emitted by the fixed source in a space-time cell.

The STLC method, reported in previous work[3], seeks to represent the collisional source as a linear function of x and t in a cell. To satisfy conservation, the scattering term in Eq. (3.6) must equal the scattering ratio multiplied by the collision term. There is a similar requirement that applies to the first spatial and first time moments. The constraint on the average and flux moments to satisfy conservation is thus given in Eqs. (3.7) - (3.9).

$$\sum_{m=1}^{M} w_m \sqrt{1+\mu_m^2} \sum_{k=1}^{\#\text{ of tracks}} \Delta\omega_{k,m} \Delta s_{k,m} \bar{\Psi}_{m,k} = \sum_{m=1}^{M} w_m \sqrt{1+\mu_m^2} \sum_{k=1}^{\#\text{ of tracks}} \Delta\omega_{k,m} \frac{\Delta s_{k,m}}{2} \bar{\phi}_k^{SS} \tag{3.7}$$

$$\sum_{m=1}^{M} w_m \sqrt{1+\mu_m^2} \sum_{k=1}^{\#\text{ of tracks}} \Delta\omega_{k,m} \Delta s_{k,m} \Psi_{m,k}^{x} = \sum_{m=1}^{M} w_m \sqrt{1+\mu_m^2} \sum_{k=1}^{\#\text{ of tracks}} \Delta\omega_{k,m} \frac{\Delta s_{k,m}}{2} \phi_k^{x,SS} \tag{3.8}$$

$$\sum_{m=1}^{M} w_m \sqrt{1+\mu_m^2} \sum_{k=1}^{\#\text{ of tracks}} \Delta\omega_{k,m} \Delta s_{k,m} \Psi_{m,k}^{t} = \sum_{m=1}^{M} w_m \sqrt{1+\mu_m^2} \sum_{k=1}^{\#\text{ of tracks}} \Delta\omega_{k,m} \frac{\Delta s_{k,m}}{2} \phi_k^{t,SS} \tag{3.9}$$

The first spatial and time moments of the angular flux along a track are represented by $\Psi_{m,k}^{x}$ and $\Psi_{m,k}^{t}$ and the corresponding moments of the scalar flux function for the scattering source are defined as $\phi_k^{x,SS}$ and $\phi_k^{t,SS}$ in Eqs. (3.7)-(3.9).

The linear functional form of the collisional source is shown in Eq. (3.10), where x_i is the spatial midpoint of a cell and t_n is the midpoint of a time step.

$$\phi_{i,n}^{SS}(x,t) = \bar{\phi}_{i,n}^{SS} + \phi_{i,n}^{x,SS} \frac{2(x-x_i)}{\Delta x_i} + \phi_{i,n}^{t,SS} \frac{2(t-t_n)}{\Delta t_n} \tag{3.10}$$

With the definition given in Eq. (3.10), the coefficients, $\bar{\phi}_{i,n}^{SS}$, $\phi_{i,n}^{x,SS}$, and $\phi_{i,n}^{t,SS}$, are uniquely determined by Eqs. (3.7)-(3.9), given mild assumptions about the number of tracks crossing the space-time cell. It is easy to show that Eqs. (3.7)-(3.9) perform a least-squares fit to the assumed linear function for $\phi_{i,n}^{SS}(x,t)$. Solving for these

coefficients in each (x, t) cell amounts to solving a 3x3 system of equations. This least-squares solution satisfies local conservation, uses the analytic solution along each track, and produces smooth cell-to-cell variation in reaction rates for problems in which this is true. The same reasoning for achieving these properties as outlined for the PWL-LC method applies to the STLC method.

The conclusions from the results and analysis of this STLC method in (x, t) showed that this method can accurately find the solution of diffusive and streaming problems. In the thick diffusion limit the STLC method behaves much like an implicit linear FEM discretization with a slight "flaw" due to the time-slope term which arises from using a linear (instead of bilinear) representation of the collision source in each (x, t) cell. The development of the STLC method below with various collisional source approximations will address various forms of time-slope terms and their effects on performance in the thick diffusion limit.

3.2 Coordinate System Definition in X-Y-vT Geometry

Our 2D-plus-time STLC method seeks to solve time-dependent transport problems in the (x, y, t) domain by using long characteristics that span space and time. The space and time domains are discretized into cells. In this method, a characteristic that spans the space-time domain of the problem is referred to as a *ray*, and the part of a ray in one space-time cell is referred to as a *track*. In order for each dimension of the space-time cell to have units of length, the time coordinate, t, will be converted to a vt coordinate where v is the speed of the particles. In the (x, y, vt) system, the distance traveled by a particle is the change in its vt coordinate.

In order to facilitate the integrals over track volume that will be used in the equations that follow, a rotated coordinate system is defined that relates the particle trajectory in the (x, y, vt) domain. Using the direction cosines defined in Eq. (2.3), the expressions given in Eq. (3.11) give the particle position for a particle starting at position and time (x_0, y_0, vt_0) moving in direction $\underline{\Omega}$ after traversing a distance s.

$$\begin{aligned} x &= x_0 + \mu s \\ y &= y_0 + \eta s \\ vt &= vt_0 + s \end{aligned} \tag{3.11}$$

Figure 7 shows an illustration of the rotated coordinate system in the (x, y, vt) domain. The s-axis is not shown because it exists in (x, y, z) space, and the z-axis is not shown in this figure. The rotated coordinate system in (u, ω, vt) uses the same u-axis and ω-axis as defined in Figure 2, in the x-y plane, with the added vt-axis. From this figure, the track volume can be defined using a particle trajectory Δs and two track spacings defined as Δu and $\Delta\omega$. The particle trajectory for a particle moving in direction $\underline{\Omega}$ in (x, y, z) can be projected into this (x, y, vt) space by using the ρ-axis shown in Figure 7.

The analytic solution along a characteristic, k, is defined as given in Eq. (3.12). In this equation, τ is the distance traveled along the k^{th} track from ($x-\mu_m\tau$, $y-\eta_m\tau$, $vt-\tau$) to (x, y, vt) and $q_{tot,m}$ is the total source for the m^{th} quadrature direction, including the collisional source.

$$\begin{aligned} \Psi_{m,k}(x,y,vt) &= \Psi^{in}_{m,k}(x-\mu_m\tau, y-\eta_m\tau, vt-\tau)e^{-\tau\sigma_t} \\ &+ \int_0^{\tau} ds\, q_{tot,m}(x-\mu_m s, y-\eta_m s, vt-s)e^{-s\sigma_t} \end{aligned} \tag{3.12}$$

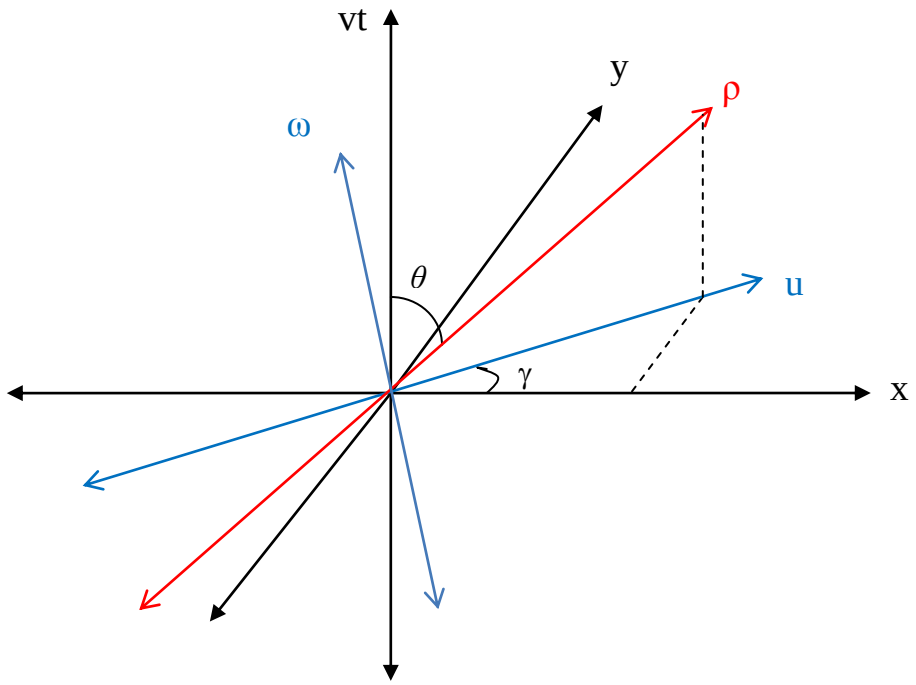


Figure 7. Rotated Coordinate System Relation in X-Y-vT

Using the geometric relations given in Eq. (3.11) and illustrated in Figure 7, the relationship between the integrals over (*x, y, vt*) and integrals over (*u, ω, s*) can be defined. This relationship is given in Eq. (3.13). Note that the same Jacobian that appeared when developing the PWL-LC method in (*x, y*) appears when relating an area in *x* and *y* to the same area in *ω* and *s*. However, since the *vt* coordinate is related to the *u* coordinate by the same term, it cancels.

$$\iiint_{volume} dxdyd(vt)f(x,y,vt) = \iiint_{\substack{same\\volume}} dud\omega dsf(u,\omega,s) \tag{3.13}$$

Also, just as with the PWL-LC method, the volume represented by the sum over tracks in a cell will not equal the actual volume of an (x, y, vt) cell as given in Eq. (3.14). Of course, as track spacing is refined, equality of these volumes is approached.

$$\sum_{k=1}^{\#\,\text{tracks}} \Delta\omega_{k,m}\Delta u_{k,m}\Delta s_{k,m} \neq v\Delta t\Delta x\Delta y \tag{3.14}$$

3.3 STLC Least-Squares Method

This section develops the least-squares approximation for finding the scalar flux source approximation in each (x, y, vt) cell. The construction of this scalar flux function is shown for various functional forms including piece-wise constant in a cell, defined as the PWC-STLC method, piece-wise linear in (x, y) and constant in time, known as the PWL-C-STLC method, piece-wise linear in (x, y) and one time slope per cell, known as the PWL-L-STLC method, piece-wise linear in (x, y) and linear in time on triangular-prism cells, known as the PPWL-STLC method, and the full piece-wise linear approximation in an (x, y, t) cell, defined as the PWL-STLC method.

The analytic solution along a track given in Eq. (3.12) satisfies the conservation equation along a track given in Eq. (3.15), where *SS* indicates the function used in the collision source, an over-bar denotes an average over the track, and the Δs indicates the length of the particle trajectory in the cell along the given track.

$$\Psi_{m,k}^{out} - \Psi_{m,k}^{in} + \sigma_t \Delta s_{k,m} \bar{\Psi}_{m,k} = \frac{1}{4\pi} \Delta s_{k,m} \left[\sigma_s \bar{\phi}_k^{SS} + \bar{Q}_{k,m} \right] \tag{3.15}$$

Multiplying Eq. (3.15) by the track volume and summing over all tracks for all quadrature directions, the STLC track-based conservation statement for a cell is found as shown in Eq. (3.16).

$$\sum_{m=1}^{M} w_m \sum_{k=1}^{\#\text{ tracks}} \Delta\omega_{k,m}\Delta u_{k,m} \left\{ \begin{array}{l} \Psi_{m,k}^{out} - \Psi_{m,k}^{in} + \sigma_t \Delta s_{k,m} \bar{\Psi}_{m,k} \\ -\dfrac{\sigma_s \Delta s_{k,m}}{4\pi} \bar{\phi}_k^{SS} + \dfrac{\Delta s_{k,m}}{4\pi} \bar{Q}_k \end{array} \right\} = 0 \qquad (3.16)$$

The five terms in Eq. (3.16) represent the number of particles streaming out of, streaming into, having collisions in, scattering in, and being emitted by a fixed source in the cell.

For a method to be conservative, the scattering term must equal σ_s/σ_t times the collision term. This condition means the following track-based relation must be met for the STLC method.

$$\sum_{m=1}^{M} w_m \sum_{k=1}^{\#\text{ tracks}} \Delta\omega_{k,m}\Delta u_{k,m}\Delta s_{k,m} \bar{\Psi}_{m,k} = \sum_{m=1}^{M} w_m \sum_{k=1}^{\#\text{ tracks}} \Delta\omega_{k,m}\Delta u_{k,m} \frac{\Delta s_{k,m}}{4\pi} \bar{\phi}_k^{SS} \qquad (3.17)$$

Equation (3.17) gives the constraint on the scalar flux function that should be used to generate the scattering source. The following sections present different source approximations used with this STLC method and how to find them using this constraint or analogous higher moments as needed.

3.3.1 PWC-STLC

The simplest collisional source approximation in a cell is a constant. This method is referred to as the piece-wise constant method, PWC-STLC, since each cell can have a different constant value for the source approximation. The functional form for the collisional source in a cell using this approximation is written as given in Eq. (3.18).

$$\phi_{cell}^{SS}(x, y, t) = \phi_{cell}^{SS} \qquad (3.18)$$

This constant, or one degree of freedom, in each cell is found by directly using Eq. (3.17). Manipulating this equation to solve for this constant, Eq. (3.19) gives the expression for this constant in each cell.

$$\phi_{cell}^{SS} = \frac{\sum_{m=1}^{M} w_m \sum_{k=1}^{\#\text{ tracks}} \Delta\omega_{k,m}\Delta u_{k,m} \int_0^{\Delta s_{k,m}} ds\psi_{m,k}}{\sum_{m=1}^{M} w_m \sum_{k=1}^{\#\text{ tracks}} \Delta\omega_{k,m}\Delta u_{k,m} \frac{\Delta s_{k,m}}{4\pi}} \tag{3.19}$$

With this constant-source approximation, the PWC-STLC method finds the exact solution along each track, produces smooth variation in cell reaction rates, and satisfies a definition of local conservation. Although this source approximation may be adequate for a variety of problems, it has been proven through past research on characteristic methods that it will fail in the thick diffusion limit[21]. This property of the PWC-STLC method will be shown through an asymptotic analysis in section 4 and a test problem in section 5.

3.3.2 PWL-C-STLC

The next logical approximation to use for the collisional source is a PWL function in *x* and *y* with a constant approximation of time dependence. This method is referred to as the piece-wise linear and constant method, PWL-C-STLC. This approximation uses the same PWL basis used in the development of the PWL-LC method in (*x*, *y*). Therefore, *N* equals the number of cell vertices of the polygonal cell and also the degrees of freedom in the (*x*, *y*, *vt*) cell. Each of these basis functions is linear in *x* and *y* on each triangular prism subcell but constant in *vt*. The functional form of this source approximation is given in Eq. (3.20).

$$\phi_{cell}^{SS}(x,y,t)=\sum_{j=1}^{N}\phi_{cell,j}^{SS}b_{PWL,j}(x,y) \tag{3.20}$$

The N coefficients in Eq. (3.20), $\phi_{cell,j}^{SS}$, are found by replacing the 0th spatial-moment equation given in Eq. (3.17) by N spatial-moment equations obtained by multiplying the equation of a track by each PWL basis function and then summing over all tracks and directions. Equation (3.21) gives the set of N equations required to find these coefficients of the collisional source.

$$\sum_{m=1}^{M}w_m\sum_{k=1}^{\#\text{ tracks}}\Delta\omega_{k,m}\Delta u_{k,m}\int_{0}^{\Delta s_{k,m}}dsb_i(s)\left[\begin{array}{l}\Psi_{m,k}(s)-\\ \dfrac{1}{4\pi}\displaystyle\sum_{j=1}^{N}\phi_{cell,j}^{SS}b_{cell,j}(s)\end{array}\right]=0,\ i=1...N \tag{3.21}$$

This equation is equivalent to a least-squares determination of the scalar flux function; it minimizes the squared difference between the scalar flux function and the PWL-C-STLC solution when integrated along each track and summed over all tracks for all directions in a cell. Therefore, this set of equations is an NxN system that is solved to find the coefficients of the scalar flux function. Note again that these PWL functions are chosen for this source approximation due to their good behavior in the thick diffusion limit on polygonal cells in steady-state problems. However for these (x, y, vt) cells, it is expected that this source approximation does not have enough degrees of freedom in the approximation for the time variable to perform well in the thick diffusion limit on polygonal prism cells.

3.3.3 PWL-L-STLC

The next approximation builds upon the previous method and uses the same PWL basis in x and y with a single added basis function per cell that is a constant

multiplied by t, therefore it is referred to as the piece-wise linear in (x, y) and linear in t, PWL-L, approximation. In other words, this source approximation has a PWL basis approximation in (x, y) with an added time slope and the basis can be written as given in Eq. (3.22). This time-slope notation is based upon the previously developed STLC method in $(x, t)^3$.

$$b_i(x,y,t) = \begin{cases} b_{i,PWL}(x,y) & i = 1,...,N \\ \dfrac{2\left(t - t^{n+1/2}\right)}{\Delta t^n}, & i = N+1 \end{cases} \tag{3.22}$$

The functional form for this source approximation is given in Eq. (3.23). The number of unknowns in an (x, y, t) cell with this approximation is N+1 where N is still equal to the number of vertices in the corresponding (x, y) polygonal cell.

$$\phi_{cell}^{SS}(x,y,t) = \sum_{j=1}^{N+1} \phi_{cell,j}^{SS} b_{cell,j}(x,y,t) \tag{3.23}$$

The coefficients, $\phi_{cell,j}^{SS}$, in this function are found by the same least-squares procedure given in Eq. (3.21), but with the number of basis functions and equations equal to N+1. Based upon the behavior of the STLC method in (x, t) in the thick diffusion limit, using only one time slope in (x, y, t) will not yield the desired accuracy in this limit because an unphysical time-slope term will be present in the leading-order discretized diffusion equation. Therefore, the next section explores the use of a piece-wise linear function in (x, y) with each PWL function multiplied by a linear function of t to approximate the collision source.

3.3.4 PPWL-STLC

This section develops an approximation for the collisional source using a piece-wise linear functional form in x and y and linear form in time specifically designed for prism cells. This method, referred to as the prism piece-wise linear method, or PPWL-STLC method, uses piece-wise linear functions in (x, y) with each multiplied by two independent linear functions in t. The collisional source using this basis can be represented as given in Eq. (3.24), where N is equal to the number of cell vertices in the (x, y) cell. The cells in (x, y, vt) space require $2N$ vertices and basis functions to fully describe the source approximation in the cell due to each spatial degree of freedom being allowed to vary linearly in time.

$$\phi_{cell}^{SS}(x,y,t) = \sum_{j=1}^{2N} \phi_{cell,j}^{SS} b_{cell,j}(x,y,t) = \sum_{j=1}^{2N} \phi_{cell,j}^{SS} \left(f_{PWL}(x,y)\, f_L(t) \right)_j \tag{3.24}$$

As mentioned previously and based upon the results from the STLC method in (x, t), this functional form is chosen due to its postulated ability to perform well in the thick diffusion limit on prism cells. Other bases exist which can achieve good behavior in this limit in two dimensions [24] and three spatial dimensions, including Wachspress's rational bases[33] and Warsa's piecewise polynomial bases[34]; however there are no cell-wise polynomial bases that achieve good behavior in the diffusion limit on general polyhedra[23,35]. Since the (x, y, vt) cells of this PPWL-STLC method are arbitrary prisms, this excellent behavior in the diffusion limit is expected for this method.

To find the coefficients, $\phi_{cell,j}^{SS}$, of the source approximation given in Eq. (3.24), the 0^{th} spatial moment equation given in Eq. (3.17) is replaced by $2N$ spatial-moment

equations obtained by multiplying the equation of a track by each basis function and then summing over all tracks and directions. Equation (3.25) gives the set of equations that determines the required coefficients of the collisional source.

$$\sum_{m=1}^{M} w_m \sum_{k=1}^{\#\text{ tracks}} \Delta\omega_{k,m}\Delta u_{k,m} \int_0^{\Delta s_{k,m}} ds b_i(s) \left[\Psi_{m,k}(s) - \frac{1}{4\pi} \sum_{j=1}^{2N} \phi_{cell,j}^{SS} b_{cell,j}(s) \right] = 0,\ i = 1...2N \qquad (3.25)$$

This equation is equivalent to a least-squares determination of the scalar flux function. It minimizes the squared difference between the scalar flux function and the PPWL-STLC solution when integrated along each track and summed over all tracks for all directions in a cell. Therefore, this set of equations is a 2*N*x2*N* system that is solved to find the coefficients of the scalar flux function.

One computationally advantageous aspect of this PPWL-STLC method is that sets of tracks in the *vt* dimension have the same geometry or track length travelled in a space-time cell. Taking advantage of these sets of tracks can lead to much less computational work. This source approximation can also be applied to prism cells in (*x*, *y*, *z*) space by replacing the *t* variable by *z*, as outlined in a following section.

3.3.5 Full PWL-STLC

Finally, another source approximation that can be useful with this STLC method is a full piece-wise linear functional form for the collision source in each polygonal prism cell in (*x*, *y*, *t*). The basis functions for this PWL function include cross terms not included with the PWL representation of the previous section. This PWL basis is very similar to the basis used in the PWL-LC method in (*x*, *y*, *z*) but the *z* variable is replaced

by *vt*. The functional form for this PWL collisional source is given in Eq. (3.26) where *N* is still equal to the number of vertices of the polygonal cell.

$$\phi_{cell}^{SS}(x,y,t) = \sum_{j=1}^{2N} \phi_{cell,j}^{SS} b_{PWL,j}(x,y,t) \tag{3.26}$$

This PWL basis is linear on each tetrahedral subcell of the (*x*, *y*, *vt*) cell, similar to the tetrahedral subcell defined with the PWL-LC method in (*x*, *y*, *z*).

The same least-squares procedure from the previous section given in Eq. (3.25) is used to find the coefficients, $\phi_{cell,j}^{SS}$, of Eq. (3.26). Based upon previous analysis, it is expected this PWL-STLC method will perform very well in the thick diffusion limit but will require more computational work than the previous prism method due to splitting the (*x*, *y*, *vt*) cell into 6*N* tetrahedral subcells during the solving procedure. An asymptotic analysis of this full PWL-STLC method in the thick diffusion limit is not performed in this work due to its complexity and it is not necessary to show the usefulness of the PPWL-STLC.

3.4 Prism PWL-LC Method in X-Y-Z

The PPWL-STLC method can be extended to prism cells in (*x*, *y*, *z*). This prismatic geometry occurs for most reactor problems and could therefore be very useful for neutron transport calculations. The basis for this PPWL-LC is PWL in (*x*, *y*) with each *N*/2 basis multiplied by two linear functions in *z*, where *N* is the number of cell vertices. The collisional source using this basis can be represented as given in Eq. (3.27), where *N* is equal to the number of cell vertices in the (*x*, *y*, *z*) cell.

$$\phi_{cell}^{SS}(x,y,z) = \sum_{j=1}^{N/2} \phi_{cell,j}^{SS} b_{cell,j}(x,y,z) = \sum_{j=1}^{N/2} \phi_{cell,j}^{SS} \left(f_{PWL}(x,y) f_L(z) \right)_j \qquad (3.27)$$

The same least-squares procedure as outline in previous section can be applied to find the coefficients of this collisional source in each (x, y, z) cell. A big advantage of this PPWL-LC method over the general PWL-LC method is that it requires much less computational work. Similar to the PPWL-STLC method, calculations only need to be done on N/2 triangular prism subcells instead of 6N tetrahedral subcells. This method can also take advantage of the sets of tracks in z with the same geometric layout during a transport sweep.

4. ANALYSIS OF THE PWL-LC AND STLC LEAST-SQUARES METHODS

This section presents a series of analyses of the PWL-LC and STLC methods. It begins with a discussion and example that show the difference between cell-wise conservation and track-wise conservation and illustrates the unsmooth reaction rates that LC and STLC methods can produce. Then it turns to a series of asymptotic analyses of the methods in the thick diffusion limit in the context of radiative transfer. This is introduced by a review of the results of an asymptotic analysis of the analytic solution to the radiative transfer equations in the thick diffusion limit is presented, following the work of Larsen, et al[36]. The following subsections present analyses of the steady-state PWL-LC method and then the time-dependent PWL-LC method with implicit time differencing in the thick diffusion limit, applying techniques outlined in Adams, et al.,[21] Adams and Nowak,[37] and Morel, et al[39]. The final subsection presents an analysis of the STLC method in the thick diffusion limit for various source approximations that were presented in section 3. The details not included in this section can be found in Appendix A.

4.1 Smoothness of Reported Cell-Wise Reaction Rates

This section seeks to show the difference between reaction rates obtained by integrating the least-squares flux over a cell and those obtained by summing over the tracks in a cell. It also gives more details on the concept of generalized local conservation for the LC and STLC methods.

Consider a problem with a constant solution and consider a ray spacing for a particular quadrature direction such that (x, y) or (x, y, t) cells have an alternating pattern

of 2/3/2/3… tracks as shown in Figure 8. The angular flux along each track is found exactly for the postulated constant solution if the correct constant source is used, and it is the same constant along each track in each cell. Therefore, summing the reaction rates attributed to the tracks in each cell for this particular direction will lead to reaction rates for each cell in the following pattern: 2γ, 3γ, 2γ, 3γ, etc. With this definition of cell-wise reaction rates (the sum over tracks), the LC or STLC method would therefore report unsmooth cell-wise reaction rates for this problem.

However, if the least-squares flux is integrated over each cell to calculate the cell-wise reaction rates, the reaction rates will be constant and correct because the least-squares flux in this example would be the correct constant in each cell.

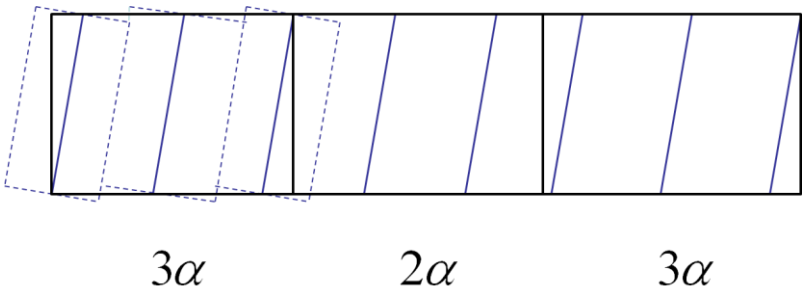


Figure 8. Track Layout in (x, y) for One Angle

A local conservation statement can be defined on a volume that is associated with a cell. This volume is a weighted average of the track-based volumes for the different quadrature directions. In Figure 8, the track-based cell volume for one quadrature direction is given by the dotted rectangles for one cell. This local conservation statement

can be derived starting with the exact equation along one track integrated along a track. This equation is given in Eq. (4.1).

$$\left\{ \Psi_{m,k}^{out} - \Psi_{m,k}^{in} + \sigma_t \int_0^{\Delta s_{k,m}} ds \Psi_{m,k}(s) - \frac{1}{4\pi} \int_0^{\Delta s_{k,m}} ds \left[\sigma_s \sum_{j=1}^{N} \phi_{cell}^{SS} b_j(s) + Q(s) \right] \right\} = 0 \qquad (4.1)$$

This equation can be multiplied by the track area and summed over all tracks (which includes multiplying by a Jacobian). It can also be multiplied by a quadrature weight and summed over all quadrature directions to give the following expression.

$$\sum_{m=1}^{M} w_m J_m \sum_{k=1}^{\#\text{tracks}} A_{k,m} \left\{ \begin{array}{l} \Psi_{m,k}^{out} - \Psi_{m,k}^{in} + \sigma_t \int_0^{\Delta s_{k,m}} ds \Psi_{m,k}(s) - \\ \frac{1}{4\pi} \int_0^{\Delta s_{k,m}} ds \left[\sigma_s \sum_{j=1}^{N} \phi_{cell,j}^{SS} b_j(s) + Q(s) \right] \end{array} \right\} = 0 \qquad (4.2)$$

Recall the least-squares definition for the collision source function defined in section 3. This least-squares definition is given in Eq. (4.3). Note that the sum over all basis functions, b_i, is equal to unity.

$$\sum_{m=1}^{M} w_m J_m \sum_{k=1}^{\#\text{tracks}} A_{k,m} \sum_{i=1}^{N} \int_0^{\Delta s_{k,m}} ds b_i(s) \left[\Psi_{m,k}(s) - \frac{1}{4\pi} \sigma_s \sum_{j=1}^{N} \phi_{cell,j}^{SS} b_j(s) \right] = 0 \qquad (4.3)$$

This least-squares definition contains the same integral over the angular flux, $\Psi_{m,k}$, that is in the track-summed expression of Eq. (4.2). Therefore, this angular flux term can be replaced in Eq. (4.2) by the scattering source function. This new expression is given in Eq. (4.4).

$$\sum_{m=1}^{M} w_m J_m \sum_{k=1}^{\#\text{tracks}} A_{k,m} \left\{ \begin{array}{l} \Psi_{m,k}^{out} - \Psi_{m,k}^{in} + \sigma_t \frac{1}{4\pi} \int_0^{\Delta s_{k,m}} ds \sum_{j=1}^{N} \phi_{cell,j}^{SS} b_j(s) - \\ \frac{1}{4\pi} \int_0^{\Delta s_{k,m}} ds \left[\sigma_s \sum_{j=1}^{N} \phi_{cell,j}^{SS} b_j(s) + Q(s) \right] \end{array} \right\} = 0 \qquad (4.4)$$

Subtracting like terms in this equation yields a true conservation statement as given in Eq. (4.5).

$$\sum_{m=1}^{M} w_m J_m \sum_{k=1}^{\#\text{tracks}} A_{k,m} \left\{ \begin{array}{l} \Psi_{m,k}^{out} - \Psi_{m,k}^{in} + \sigma_a \frac{1}{4\pi} \int_0^{\Delta s_{k,m}} ds \sum_{j=1}^{N} \phi_{cell,j}^{SS} b_j(s) - \\ \frac{1}{4\pi} \int_0^{\Delta s_{k,m}} ds Q(s) \end{array} \right\} = 0 \qquad (4.5)$$

Note that this conservation statement is on a volume that is an average of the track-based volumes for different directions.

4.2 Review of Analysis of Radiative Transfer Equations in Diffusion Limit

A quick review of the analytic solution to the time-dependent and frequency-dependent radiative transfer equations in the thick diffusion limit is presented in this section[36,-39]. The radiative transfer equations are given in Eq. (4.6) and Eq. (4.7),

$$\begin{array}{l} \frac{1}{c} \frac{\partial \psi}{\partial t}(\vec{r}, \vec{\Omega}, \nu, t) + \vec{\Omega} \bullet \vec{\nabla} \psi(\vec{r}, \vec{\Omega}, \nu, t) + \sigma_a(\vec{r}, \nu, T(\vec{r})) \psi(\vec{r}, \vec{\Omega}, \nu, t) = \\ \sigma_a(\vec{r}, \nu, T(\vec{r})) \frac{1}{4\pi} B(\nu, T) \end{array} , \qquad (4.6)$$

$$C_v(\vec{r}, T) \frac{\partial T}{\partial t} = \int_0^{\infty} d\nu \int_{4\pi} d\Omega \sigma_a(\vec{r}, \nu, T(\vec{r})) \left[\psi(\vec{r}, \vec{\Omega}, \nu, t) - \frac{1}{4\pi} B(\nu, T) \right] + Q(\vec{r}, t) , \qquad (4.7)$$

where

$c \equiv$ photon speed, speed of light

$\vec{r} \equiv$ photon location

$\vec{\Omega} \equiv$ photon direction of travel

$\nu \equiv$ photon frequency

$t \equiv$ time [s]

$\psi \equiv$ specific intensity of particles [c*energy per cm^3-str-freq].

$\sigma_a \equiv$ total photon opacity [cm^1]

$B \equiv$ Planck function

$C_v \equiv$ specific heat capacity of material [energy per cm^3-K]

$T \equiv$ material Temperature

$Q \equiv$ heat source [energy per cm^3-s]

The scalar intensity is defined as given in Eq. (4.8).

$$\phi(\vec{r},\nu,t) = \int_{4\pi} d\Omega \psi\left(\vec{r},\vec{\Omega},\nu,t\right) \tag{4.8}$$

Also the Planck function is defined as given in Eq. (4.9).

$$B(\nu,T) = \frac{4\pi h\nu^3}{c^2}\frac{1}{e^{h\nu/kT}-1} \tag{4.9}$$

The radiation energy density can be defined as given in Eq. (4.10).

$$E_R = aT_R^4(\vec{r},t) = \frac{1}{c}\int_0^\infty d\nu \int_{4\pi} d\Omega \psi\left(\vec{r},\vec{\Omega},\nu,t\right) \tag{4.10}$$

Equation (4.10) also defines the "radiation temperature", T_R. In standard notation, *h* is Planck's constant, *k* is Boltzmann's constant, and *a* is the radiation constant:

$$a = \frac{8\pi^5 k^4}{15h^3c^3}. \tag{4.11}$$

The radiative transfer equations can be scaled by a small parameter, ε, such that small epsilon corresponds to an optically thick domain whose radiation interactions are

strongly dominated by absorption and re-emission. This limit as epsilon goes to zero is known as the "diffusion limit," and when this scaling is applied to discretized equations in which cell size does not very with epsilon, it is known as the "thick diffusion limit"[36-38]. The unknowns, ψ and T, are postulated to have a power series expansion in terms of ε, as shown for the angular intensity in Eq. (4.12).

$$\psi = \psi^{(0)} + \varepsilon\psi^{(1)} + \varepsilon^2\psi^{(2)} + \ldots \tag{4.12}$$

Since the opacity and Planck function depend on temperature, they are expanded as well. The scaled radiative transfer equations are given in Eq. (4.13) and Eq. (4.14) along with the associated boundary and initial conditions.

$$\begin{aligned} &\frac{\varepsilon}{c}\frac{\partial\psi}{\partial t}\left(\vec{r},\vec{\Omega},\nu,t\right)+\vec{\Omega}\bullet\vec{\nabla}\psi\left(\vec{r},\vec{\Omega},\nu,t\right)+\frac{\sigma_a\left(\vec{r},\nu,T\left(\vec{r}\right)\right)}{\varepsilon}\psi\left(\vec{r},\vec{\Omega},\nu,t\right)= \\ &\frac{\sigma_a\left(\vec{r},\nu,T\left(\vec{r}\right)\right)}{\varepsilon}B\left(\nu,T\right) \end{aligned} \tag{4.13}$$

$$\varepsilon C_v\left(\vec{r},T\right)\frac{\partial T}{\partial t}=\int_0^\infty d\nu\int_{4\pi}d\Omega\frac{\sigma_a\left(\vec{r},\nu,T\left(\vec{r}\right)\right)}{\varepsilon}\left[\psi\left(\vec{r},\vec{\Omega},\nu,t\right)-B\left(\nu,T\right)\right]+\varepsilon Q\left(\vec{r},t\right) \tag{4.14}$$

$$\begin{aligned} &\psi\left(\vec{r},\vec{\Omega},\nu,t\right)=F\left(\vec{r},\vec{\Omega},\nu,t\right), \qquad \vec{r}\in\partial D, \\ &\psi\left(\vec{r},\vec{\Omega},\nu,0\right)=\psi_i\left(\vec{r},\vec{\Omega},\nu\right), \\ &T\left(\vec{r},0\right)=T_i\ \left(\vec{r}\right) \end{aligned} \tag{4.15}$$

Physically, a small value of ε means the absorption/emission dominates streaming and sources in the problem. In other words, the absorption and emission rates are large relative to the time rates of change and source rates in the problem. The standard "equilibrium" diffusion equation is invariant under this scaling which is what must be

true in order for a scaling to define a diffusion limit. For more details on this scaling refer to the paper by Larsen, et. al[36].

An asymptotic analysis is performed on these scaled equations using the power series expansions of the unknowns. The results from this analysis first show that, away from boundaries, the leading-order analytic solution is Planckian as given in Eq. (4.16),

$$\psi^{(0)}\left(\vec{r},\vec{\Omega},\nu,t\right)=\frac{1}{4\pi}B\left(\nu,T^{(0)}\left(\vec{r},t\right)\right), \tag{4.16}$$

where the leading-order temperature satisfies an energy-balance equation given in Eq. (4.17).

$$a\frac{\partial}{\partial t}\left[T^{(0)}\right]^4+C_v\left(\vec{r},T^{(0)}\right)\frac{\partial T^{(0)}}{\partial t}+\vec{\nabla}\bullet\vec{J}^{(1)}=Q\left(\vec{r},t\right) \tag{4.17}$$

The O(ε) "net current density," $\vec{J}^{(1)}$, which is also called the "flux" in other fields, is defined as given in Eq. (4.18) where the mean Rosseland opacity, σ_R, defined in Eq. (4.19).

$$\vec{J}^{(1)}\left(\vec{r}\right)=-\frac{ac}{3\sigma_R\left(\vec{r},T^{(0)}\right)}\vec{\nabla}\left(T^{(0)}\right)^4 \tag{4.18}$$

$$\frac{1}{\sigma_R\left(\vec{r},T\right)}\equiv\frac{\int_0^\infty\frac{1}{\sigma\left(\vec{r},\nu,T\right)}\frac{\partial B\left(\nu,T\right)}{\partial T}d\nu}{\int_0^\infty\frac{\partial B\left(\nu,T\right)}{\partial T}d\nu}=\frac{\int_0^\infty\frac{1}{\sigma\left(\vec{r},\nu,T\right)}\frac{\partial B\left(\nu,T\right)}{\partial T}d\nu}{4acT^3} \tag{4.19}$$

Combining Eqs. (4.17) and (4.18) results in a diffusion equation satisfied by the leading order temperature, as given in Eq. (4.20).

$$a\frac{\partial}{\partial t}\left[T^{(0)}\right]^4 + C_v\left(\vec{r},T^{(0)}\right)\frac{\partial T^{(0)}}{\partial t} - \frac{\partial}{\partial \vec{r}}\left[\frac{ac}{3\sigma_R\left(\vec{r},T^{(0)}\right)}\vec{\nabla}\left(T^{(0)}\right)^4\right] = Q\left(\vec{r},t\right) \qquad (4.20)$$

This leading-order temperature satisfies an initial condition that can be found through an analysis of the "initial-layer". The result is given in Eq. (4.21), where *E* is the matter energy density and *i* indicates an actual initial condition for the underlying transport problem.

$$a\left[T^{(0)}\left(\vec{r},0\right)\right]^4 + E\left(T^{(0)}\left(\vec{r},0\right)\right) = \frac{1}{c}\int_0^\infty d\nu \int_{4\pi} d\Omega \psi_i\left(\vec{r},\vec{\Omega},\nu\right) + E\left(T_i\left(\vec{r}\right)\right) \qquad (4.21)$$

This equation defined the appropriate initial value of the leading-order temperature, $T^{(0)}\left(\vec{r},0\right)$, for the diffusion equation defined in Eq. (4.20).

There is also a boundary condition that this leading-order temperature needs to satisfy. An approximation of this condition can be found by using a variational procedure to find an approximate solution to a half-space problem. The incoming directions on the boundary have been defined by the following notation where *n* represents the outward unit normal on the boundary:

$$\mu = \vec{n}\bullet\vec{\Omega}\,. \qquad (4.22)$$

The result from the variational procedure yields the following boundary condition:

$$\begin{aligned} &T^{(0)}\left(\vec{r},t\right)\Big|_{\vec{r}} \approx T^{\text{var}}, \qquad \vec{r}\in\partial D \\ &ac\left(T^{\text{var}}\right)^4 = \\ &ac\left(T_M^{ex}\right)^4 + \sigma_R\left(\vec{r},T_M^{ex}\right)\int_0^\infty d\nu \int_\gamma d\gamma \int_{\mu<0} d\mu 3\mu^2 \frac{\psi_{inc}\left(\vec{\Omega},\nu,t\right) - B\left(\nu,T_M^{ex}\right)}{\sigma\left(\partial r,\nu,T_M^{ex}\right)} \end{aligned} \qquad . \qquad (4.23)$$

The exact "Marshak" boundary temperature in the boundary equation is defined as given in Eq. (4.24).

$$ac\left(T_M^{ex}\right)^4 = 4\pi\int_0^\infty d\nu\int_\gamma d\gamma\int_{\mu<0} d\mu 2\mu\psi_{inc}\left(\vec{\Omega},\nu,t\right) \tag{4.24}$$

Defining a Planckian-weighted inverse opacity,

$$\int_0^\infty d\nu\frac{B\left(\nu,T\right)}{\sigma\left(\vec{r},\nu,T\right)} = \frac{1}{\sigma_P\left(\vec{r},T\right)}acT^4 \tag{4.25}$$

,

the boundary condition in Eq. (4.23) can be rewritten by using this opacity and Eq. (4.24) to give the following form of the boundary condition.

$$ac\left(T^{\text{var}}\right)^4 = \int_0^\infty d\nu\int_\gamma d\gamma\int_{\mu<0} d\mu\left[\begin{array}{l} 2\mu\left(2-\dfrac{\sigma_R\left(\partial r,T_M^{ex}\right)}{\sigma_P\left(\partial r,T_M^{ex}\right)}\right)+ \\ 3\mu^2\dfrac{\sigma_R\left(\partial r,T_M^{ex}\right)}{\sigma\left(\partial r,\nu,T_M^{ex}\right)} \end{array}\right]\psi_{inc}\left(\vec{\Omega},\nu,t\right) \tag{4.26}$$

This result is a variational approximation to the exact boundary condition.

In summary, the leading-order solution on the interior satisfies Eq. (4.16) with the leading-order temperature satisfying the diffusion equation of Eq. (4.20) with initial condition given by Eq. (4.21) and a boundary condition approximately given by Eq. (4.26). Since the radiative-transfer solution limits to a diffusion solution in the optically thick limit, if a numerical method is to be accurate in this limit it must produce a solution that limits to the numerical solution of an accurate discretization of the same diffusion problem that is satisfied (to leading-order) by the analytic transport solution.

4.3 Time-Dependent PWL-LC Asymptotic Analysis in the Diffusion Limit

This section analyzes the time-dependent radiative-transfer equations in the diffusion limit using the PWL-LC method for spatial discretization in (x, y) and the fully implicit time discretization that was introduced in section 2.3. Equations (4.27) and (4.28) show the discrete-ordinates multigroup radiative transfer equations.

$$\frac{1}{c}\frac{\partial \psi_{m,g}}{\partial t}\left(\vec{r},t\right)+\vec{\Omega}_m \bullet \vec{\nabla}\psi_{m,g}\left(\vec{r},t\right)+\sigma_{a,g}\left(\vec{r},T\left(\vec{r}\right)\right)\psi_{m,g}\left(\vec{r},t\right)= \sigma_{a,g}\left(\vec{r},T\left(\vec{r}\right)\right)\frac{1}{4\pi}B_g\left(T\left(\vec{r}\right)\right) \tag{4.27}$$

$$C_v\left(\vec{r},T\right)\frac{\partial T\left(\vec{r}\right)}{\partial t}= \sum_{g=1}^{G}\sum_{m=1}^{M} w_m \sigma_{a,g}\left(\vec{r},T\left(\vec{r}\right)\right)\left[\psi_{m,g}\left(\vec{r},t\right)-\frac{1}{4\pi}B_g\left(T\left(\vec{r}\right)\right)\right]+Q\left(\vec{r},t\right) \tag{4.28}$$

Next, fully implicit time discretization is applied to these equations as shown in Eqs. (4.29) and (4.30) where n indicates the beginning of the time step (t^n, t^{n+1}).

$$\frac{1}{c}\frac{\psi_{m,g}^{n+1}\left(\vec{r}\right)-\psi_{m,g}^{n}\left(\vec{r}\right)}{\Delta t^n}+\vec{\Omega}_m \bullet \vec{\nabla}\psi_{m,g}^{n+1}\left(\vec{r}\right)+\sigma_{a,g}\left(\vec{r},T\left(\vec{r}\right)\right)\psi_{m,g}^{n+1}\left(\vec{r}\right)= \sigma_{a,g}\left(\vec{r},T\left(\vec{r}\right)\right)\frac{1}{4\pi}B_g\left(T^{n+1}\left(\vec{r}\right)\right) \tag{4.29}$$

$$C_v\frac{T^{n+1}\left(\vec{r}\right)-T^{n}\left(\vec{r}\right)}{\Delta t^n}= \sum_{g=1}^{G}\sum_{m=1}^{M} w_m \sigma_{a,g}\left(\vec{r},T\left(\vec{r}\right)\right)\left[\psi_{m,g}^{n+1}\left(\vec{r}\right)-\frac{1}{4\pi}B_g\left(T^{n+1}\left(\vec{r}\right)\right)\right]+Q\left(\vec{r}\right) \tag{4.30}$$

The rest of this analysis will not specify the definitions for the specific heat and opacities. They may be treated as explicit, fully implicit, or somewhere between. The

same scaling from section 4.2 is applied to the discretized radiative transfer equations as shown in Eqs. (4.31) and (4.32).

$$\begin{aligned}&\frac{\varepsilon}{c}\frac{\psi_{m,g}^{n+1}\left(\vec{r}\right)-\psi_{m,g}^{n}\left(\vec{r}\right)}{\Delta t^{n}}+\vec{\Omega}_{m}\bullet\vec{\nabla}\psi_{m,g}^{n+1}\left(\vec{r}\right)+\\&\frac{\sigma_{a,g}\left(\vec{r},T\left(\vec{r}\right)\right)}{\varepsilon}\psi_{m,g}^{n+1}\left(\vec{r}\right)=\frac{\sigma_{a,g}\left(\vec{r},T\left(\vec{r}\right)\right)}{\varepsilon}\frac{1}{4\pi}B_{g}\left(T^{n+1}\left(\vec{r}\right)\right)\end{aligned}\tag{4.31}$$

$$\begin{aligned}&\varepsilon C_{v}\frac{T^{n+1}\left(\vec{r}\right)-T^{n}\left(\vec{r}\right)}{\Delta t^{n}}=\\&\sum_{g=1}^{G}\sum_{m=1}^{M}w_{m}\frac{\sigma_{a,g}\left(\vec{r},T\left(\vec{r}\right)\right)}{\varepsilon}\left[\psi_{m,g}^{n+1}\left(\vec{r}\right)-\frac{1}{4\pi}B_{g}\left(T^{n+1}\left(\vec{r}\right)\right)\right]+\varepsilon Q\left(\vec{r}\right)\end{aligned}\tag{4.32}$$

Next the PWL-LC spatial discretization is applied to the scaled radiative transfer equations. The scaled transport equation for one track, denoted by *k*, is given below.

$$\begin{aligned}&\frac{\varepsilon}{c}\frac{\tilde{\psi}_{m,g,k}^{n+1}-\tilde{\psi}_{m,g,k}^{n}}{\Delta t^{n}}+\frac{\partial\psi_{m,g,k}^{n+1}}{\partial s}+\frac{\sigma_{a,g}\left(T\left(s\right)\right)}{\varepsilon}\psi_{m,g,k}^{n+1}=\\&\frac{\sigma_{a,g}\left(T\left(s\right)\right)}{\varepsilon}\frac{1}{4\pi}B_{g,k}^{n+1}\left(s\right)\end{aligned}\tag{4.33}$$

Remember from section 2.3 that this PWL-LC method chooses to represent the flux at the beginning of each time step using the PWL basis. This flux and the flux at the end of the time step are defined by Eq. (4.34).

$$\tilde{\psi}_{m,g,k}^{n}=\sum_{j=1}^{N}\psi_{m,j}^{n}b_{PWL}\left(s\right)\tag{4.34}$$

The scalar intensity for this PWL-LC method can be expressed using the PWL basis as given in Eqs. (4.35).

$$\phi_{cell,g}^{n+1}(s) = \sum_{j=1}^{N} \phi_{cell,g,j}^{n+1} b_j(s) \tag{4.35}$$

The Planckian is also represented using the PWL bases as given in Eq. (4.36).

$$B_{cell,g}^{n+1}(s) = \sum_{j=1}^{N} B_{cell,g,j}^{n+1} b_j(s) \tag{4.36}$$

The right hand side of the material energy equation is a linear combination of basis functions (assuming the source, Q, can be expressed in terms of basis functions). If this is true, the temperature must also be a combination of basis functions. Since the basis functions are linearly independent, the material energy equation holds for each coefficient of the basis function expansion. Therefore, the scaled material energy equation can be written for each j^{th} coefficient as given in Eq. (4.37).

$$\varepsilon C_v \frac{T_j^{n+1} - T_j^n}{\Delta t^n} = \sum_{g=1}^{G} \frac{\sigma_{a,g}}{\varepsilon} \left[\phi_{cell,g,j}^{SS,n+1} - B_{cell,g,j}^{n+1} \right] + \varepsilon Q_{cell,j} \tag{4.37}$$

These unknown coefficients are found by the least-squares fit in each cell as introduced previously. Therefore, the coefficients of the scalar intensity at the end of the time step and the average of the time step are found by equations in Eq. (4.38). In order to do this analysis in general so it can be applied in (x, y) and (x, y, z), the Jacobian that arose from different coordinate systems in section 2 will be denoted by J_m and the area associated with a track, which was defined by track spacings ($\Delta\omega_{k,m}$ and $\Delta u_{k,m}$) in the previous sections, will be denoted by $A_{k,m}$. Both of these terms are present in the least-squares fit for a cell.

$$\sum_{m=1}^{M} w_m J_m \sum_{k=1}^{\#\text{ tracks}} A_{k,m} \int_0^{\Delta s_k} dsb_i(s) \left[\begin{array}{l} \psi_{m,g,k}^{n+1}(s) - \\ \dfrac{1}{4\pi} \sum_{j=1}^{N} \phi_{cell,g,j}^{n+1} b_j(s) \end{array} \right] = 0, \qquad i = 1...N \tag{4.38}$$

The PWL-LC equation for a cell is found by summing over all tracks in a cell and all angles, equivalent to the least-squares procedure. This summation leads to the set of equations (for $i = 1,\ldots, N$) given in Eqs. (4.39).

$$\sum_{m=1}^{M} w_m J_m \sum_{k=1}^{\#tracks} A_{k,m} \int_0^{\Delta s_{k,m}} dsb_i(s) \left[\begin{array}{l} \dfrac{\varepsilon}{c} \dfrac{\tilde{\psi}_{m,g,k}^{n+1} - \tilde{\psi}_{m,g,k}^{n}}{\Delta t^n} + \dfrac{\partial \psi_{m,g,k}^{n+1}}{\partial s} \\ + \dfrac{\sigma_{a,g}(T(s))}{\varepsilon} \psi_{m,g,k}^{n+1} = \\ \dfrac{\sigma_{a,g}(T(s))}{\varepsilon} \dfrac{1}{4\pi} B_{g,k}^{n+1}(s) \end{array} \right] \tag{4.39}$$

In the following analysis, a symmetric quadrature set is assumed with the following properties:

$$\sum_{m=1}^{M} w_m = 4\pi, \tag{4.40}$$

$$\sum_{m=1}^{M} w_m \vec{\Omega}_m = 0. \tag{4.41}$$

Also, for ease of notation the definition shown in Eq. (4.42) will be used in the following analysis. This notation, $[\sigma\psi]^{(1)}$, is a more compact way to write $\sigma^{(0)}\psi^{(1)} + \sigma^{(1)}\psi^{(0)}$, for example.

$$[fg]^{(k)} \equiv \text{coefficient of } \varepsilon^k \text{ in product expansion of } f \text{ and } g \tag{4.42}$$

To begin performing this analysis, it is postulated that the unknowns can be expanded as a power series as given in Eqs. (4.43) and (4.44).

$$\psi_{m,g,k} = \psi^{(0)}_{m,g,k} + \varepsilon\psi^{(1)}_{m,g,k} + \varepsilon^2\psi^{(2)}_{m,g,k} + \ldots \tag{4.43}$$

$$\begin{aligned} T_k &= T_k^{(0)} + \varepsilon T_k^{(1)} + \varepsilon^2 T_k^{(2)} + \ldots \\ T_k^4 &= T_k^{4,(0)} + \varepsilon T_k^{4,(1)} + \varepsilon^2 T_k^{4,(2)} + \ldots = \left(T_k^{(0)}\right)^4 + 4\varepsilon\left(T_k^{(0)}\right)^3 T_k^{(1)} + \ldots \end{aligned} \tag{4.44}$$

Similar expansions are also applied to temperature-dependent quantities including $\sigma_{a,g,k}$ and $B_{g,k}$. These expansions are then substituted into the scaled and discretized radiative transfer equations.

First, the solution on the interior of the problem is analyzed. From the $O(1/\varepsilon)$ terms found by substituting the expansions into Eq. (4.33), it is found that the leading-order angular intensity along a track is isotropic and Planckian as shown in Eq. (4.45).

$$\psi^{(0),n+1}_{m,g,k}(s) = \frac{1}{4\pi} B^{(0),n+1}_{g,k}(s) \tag{4.45}$$

From the $O(1/\varepsilon)$ terms of Eq. (4.37), it is also found that the leading-order least-squares PWL scalar intensity is Planckian.

$$\phi^{(0),n+1}_{cell,g,j} = B^{(0),n+1}_{cell,g,j}, \qquad j = 1,..,N. \tag{4.46}$$

The $O(\varepsilon)$ terms of the discretized equations for a cell, Eq. (4.39), yield the following expression when summed over all groups for the i^{th} equation.

$$\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=1}^{\#tracks} A_{k,m} \int_0^{\Delta s_{k,m}} ds b_i(s) \left[\frac{1}{c}\frac{\tilde{\psi}^{(0),n+1}_{m,g,k} - \tilde{\psi}^{(0),n}_{m,g,k}}{\Delta t^n} + \frac{\partial \psi^{(1),n+1}_{m,g,k}}{\partial s} + \left[\sigma_{a,g}\left(\psi^{n+1}_{m,g,k} - \frac{1}{4\pi} B^{n+1}_{g,k} \right) \right]^{(2)} \right] = 0 \tag{4.47}$$

Substituting the result from Eq. (4.45) into Eq. (4.47) yields the following expression.

$$\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=1}^{\#tracks} A_{k,m} \int_0^{\Delta s_{k,m}} dsb_i(s) \left[\begin{array}{l} \frac{1}{4\pi} \frac{B_{g,k}^{(0),n+1} - B_{g,k}^{(0),n}}{c\Delta t^n} + \frac{\partial \psi_{m,g,k}^{(1),n+1}}{\partial s} \\ + \left[\sigma_{a,g} \left(\psi_{m,g,k}^{n+1} - \frac{1}{4\pi} B_{g,k}^{n+1} \right) \right]^{(2)} \end{array} \right] = 0 \qquad (4.48)$$

The $O(\varepsilon)$ terms from Eq. (4.37) are given as the following.

$$\left(C_v \frac{T_j^{(0),n+1} - T_j^{(0),n}}{\Delta t^n} - Q_j \right) = \sum_{g=1}^{G} \left[\sigma_{a,g} \left(\phi_{cell,g,j}^{n+1} - B_{cell,g,j}^{n+1} \right) \right]^{(2)} \qquad (4.49)$$

For the rest of this analysis, it will be assumed that the quadrature set and track spacings are the same for all energy groups; therefore the group summation can be moved inside the angle summation when needed. Returning to Eq. (4.48) and applying this group assumption, the second-order terms are equivalent to the following expression from the least-squares definition (*LS* = least-squares in Eq. (4.50)).

$$\begin{array}{l} \sum_{m=1}^{M} w_m J_m \sum_{k=1}^{\#tracks} A_{k,m} \int_0^{\Delta s_{k,m}} dsb_i(s) \sum_{g=1}^{G} \left[\sigma_{a,g,k} \left(\psi_{m,g,k}^{n+1} - \frac{1}{4\pi} B_{g,k}^{n+1} \right) \right]^{(2)} = \\ \sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=1}^{\#tracks} A_{k,m} \int_0^{\Delta s_{k,m}} dsb_i(s) \left[\sigma_{a,g,k} \frac{1}{4\pi} \left(\phi_{g,k}^{n+1} - B_{g,k}^{n+1} \right) \right]_{LS}^{(2)} \end{array} \qquad (4.50)$$

The second-order terms in Eq. (4.48) can be eliminated using Eq. (4.49). Doing so yields the expression given in Eq. (4.51), where the temperature, Planckian, and source are all PWL functions.

$$\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=1}^{\#tracks} A_{k,m} \int_0^{\Delta s_{k,m}} dsb_i(s) \left[\begin{array}{l} \frac{1}{4\pi c} \frac{B_{g,k}^{(0),n+1} - B_{g,k}^{(0),n}}{\Delta t^n} + \frac{\partial \psi_{m,g,k}^{(1),n+1}}{\partial s} + \\ \frac{1}{4\pi} \left(C_v \frac{T_{PWL}^{(0),n+1} - T_{PWL}^{(0),n}}{\Delta t^n} - Q_{PWL} \right) \end{array} \right] = 0 \qquad (4.51)$$

The first-order derivative term in Equation (4.51) needs some special analysis. First the integral over s in this term is analyzed. This integration can be done by parts, which turns the integral into the expression in Eq. (4.52).

$$\int_0^{\Delta s_{k,m}} ds b_i(s) \frac{\partial \psi_{m,g,k}^{(1),n+1}}{\partial s} = b_i(s) \psi_{m,g,k}^{(1),n+1}(s)\Big|_0^{\Delta s_{k,m}} - \int_0^{\Delta s_{k,m}} ds \frac{\partial b_i}{\partial s} \psi_{m,g,k}^{(1),n+1}(s) \tag{4.52}$$

Recall the identity,

$$\frac{\partial b_i}{\partial s} = \vec{\Omega}_m \bullet \vec{\nabla} b_i \,. \tag{4.53}$$

With this relation, the term in question now becomes the following expression.

$$\sum_{g=1}^{G} \sum_{m=1}^{M} w_m J_m \sum_{k=1}^{\#tracks} A_{k,m} \left[b_i(s) \psi_{m,g,k}^{(1),n+1}(s)\Big|_0^{\Delta s_{k,m}} - \int_0^{\Delta s_{k,m}} ds \vec{\nabla} b_i \bullet \vec{\Omega}_m \psi_{m,g,k}^{(1),n+1}(s) \right] \tag{4.54}$$

The first term in Eq. (4.54) can be analyzed for the tracks along an edge of the cell for which b_i is nonzero. Here "edge" refers to an edge of the (x, y) cell or a face of the (x, y, z) cell. For tracks on an edge, the following expression is known based upon geometry, where $\vec{e}_n$ is the outward unit normal of an edge (where an edge is denoted by e) and $\partial A_{k,m,e}$ is the area associated with the k^{th} track on the edge.

$$J_m A_{k,m} = \partial A_{k,m,e} \left(\vec{e}_n \bullet \vec{\Omega}_m \right) \tag{4.55}$$

With this substitution, the first term in expression (4.54) can be identified as the weighted net current across the edges of the cell that touch the vertex where b_i is centered, because $s = 0$ corresponds to the incoming position of a track on the cell and $s = \Delta s_{k,m}$ corresponds to the exiting position of the track on the cell. Therefore, the summation over all tracks entering and exiting along an edge turns into a net current.

Equation (4.56) shows what this first term becomes, a weighted net current for all edges that touch vertex v_i, at which b_i is centered and is nonzero. This expression defines an edge-integrated b_i-weighted net current that allows the net current across each edge (e) to be written in a simpler way.

$$\begin{aligned}&\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=1}^{\#tracks} A_{k,m}\, b_i(s)\psi_{m,g,k}^{(1),n+1}(s)\Big|_0^{\Delta s_{k,m}} = \\ &-\sum_{g=1}^{G}\sum_{m=1}^{M} w_m \sum_{k=1}^{\#tracks} b_i(s)\partial A_{k,m,e}\left(\vec{e}_n \bullet \vec{\Omega}_m\right)\psi_{m,g,k}^{(1),n+1}(s)\Big|_0^{\Delta s_{k,m}} \equiv \sum_{g=1}^{G}\sum_{edge\in v_i}\int_{\partial A_e} dA_e b_{i,e} J_e^{(1),n+1}\end{aligned} \tag{4.56}$$

Again, the edge-integrated J_e is just a defined term that reminds us of the physical meaning of the more complicated expression it replaces. Expression (4.54) can now be rewritten by substituting the expression in Eq. (4.56).

$$\sum_{g=1}^{G}\sum_{edge\in v_i}\int_{\partial A_e} dA_e b_{i,e} J_e^{(1),n+1} - \sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=1}^{\#tracks} A_{k,m}\int_0^{\Delta s_{k,m}} ds\vec{\nabla} b_i \bullet \vec{\Omega}_m \psi_{m,g,k}^{(1),n+1}(s) \tag{4.57}$$

Note that these edge net currents have contributions from angular intensities from neighboring cells. Putting this expression back into Eq. (4.51), the i^{th} equation for the cell now becomes the following equation.

$$\begin{aligned}&\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=1}^{\#tracks} A_{k,m}\int_0^{\Delta s_{k,m}} ds b_i(s)\frac{1}{4\pi c}\frac{B_{g,k}^{(0),n+1}-B_{g,k}^{(0),n}}{\Delta t^n} + \\ &\sum_{edge\in v_i}\int_{\partial A_e} dA_e b_{i,e} J_e^{(1),n+1} - \sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=1}^{\#tracks} A_{k,m}\int_0^{\Delta s_{k,m}} ds\vec{\nabla} b_i \bullet \vec{\Omega}_m \psi_{m,g,k}^{(1),n+1}(s) + \\ &\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \frac{1}{4\pi}\sum_{k=1}^{\#tracks} A_{k,m}\int_0^{\Delta s_{k,m}} ds b_i(s)\left(C_v\frac{T_{PWL}^{(0),n+1}-T_{PWL}^{(0),n}}{\Delta t^n} - Q_{PWL}\right) = 0\end{aligned} \tag{4.58}$$

Next, this equation is summed for all cells that touch the vertex, v_i, at which b_i is centered. In doing so, the edge net currents will cancel due to the shared edges between neighboring cells. Therefore this summation yields the following equation.

$$\sum_{cells \in v_i} \left[\begin{array}{l} \sum_{g=1}^{G} \sum_{m=1}^{M} w_m J_m \sum_{k=1}^{\#tracks} A_{k,m} \int_0^{\Delta s_{k,m}} ds b_i(s) \frac{1}{4\pi c} \frac{B_{g,k}^{(0),n+1} - B_{g,k}^{(0),n}}{\Delta t^n} - \\ \sum_{g=1}^{G} \sum_{m=1}^{M} w_m J_m \sum_{k=1}^{\#tracks} A_{k,m} \int_0^{\Delta s_{k,m}} ds \vec{\nabla} b_i \bullet \vec{\Omega}_m \psi_{m,g,k}^{(1),n+1}(s) + \\ \sum_{g=1}^{G} \sum_{m=1}^{M} w_m J_m \frac{1}{4\pi} \sum_{k=1}^{\#tracks} A_{k,m} \int_0^{\Delta s_{k,m}} ds b_i(s) \left(\begin{array}{l} C_v \frac{T_{PWL}^{(0),n+1} - T_{PWL}^{(0),n}}{\Delta t^n} \\ -Q_{PWL} \end{array} \right) \end{array} \right] = 0 \qquad (4.59)$$

The summation over groups can also be applied to the Planckian, yielding the following expression, in which the T^4 unknowns are PWL functions.

$$\sum_{cells \in v_i} \left[\begin{array}{l} \sum_{m=1}^{M} w_m J_m \sum_{k=1}^{\#tracks} A_{k,m} \int_0^{\Delta s_{k,m}} ds b_i(s) \frac{a}{4\pi} \frac{\left(T_{PWL}^{(0),n+1}\right)^4 - \left(T_{PWL}^{(0),n}\right)^4}{\Delta t^n} - \\ \sum_{g=1}^{G} \sum_{m=1}^{M} w_m J_m \sum_{k=1}^{\#tracks} A_{k,m} \int_0^{\Delta s_{k,m}} ds \vec{\nabla} b_i \bullet \vec{\Omega}_m \psi_{m,g,k}^{(1),n+1}(s) + \\ \sum_{g=1}^{G} \sum_{m=1}^{M} w_m J_m \frac{1}{4\pi} \sum_{k=1}^{\#tracks} A_{k,m} \int_0^{\Delta s_{k,m}} ds b_i(s) \left(\begin{array}{l} C_v \frac{T_{PWL}^{(0),n+1} - T_{PWL}^{(0),n}}{\Delta t^n} \\ -Q_{PWL} \end{array} \right) \end{array} \right] = 0 \qquad (4.60)$$

This equation will be returned to shortly with a definition for the weighted first-order current that appears.

The $O(1)$ terms from Eqs. (4.39) and (4.37) yield the following expressions when Eq. (4.39) is summed over groups.

$$\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=1}^{\#tracks} A_{k,m} \int_0^{\Delta s_{k,m}} dsb_i(s) \left[\frac{\partial \psi_{m,g,k}^{(0),n+1}}{\partial s} + \left[\sigma_{a,g} \left(\psi_{m,g,k}^{n+1} - \frac{1}{4\pi} B_{g,k}^{n+1} \right) \right]^{(1)} \right] = 0 \tag{4.61}$$

$$\sum_{g=1}^{G} \left[\sigma_{a,g} \frac{1}{4\pi} \left(\phi_{cell,g,j}^{n+1} - B_{cell,g,j}^{n+1} \right) \right]^{(1)} = 0 \tag{4.62}$$

From Eq. (4.62) and the least-squares definition of ϕ_g^{n+1}, the second term in Eq. (4.61) is zero; therefore, the following expression is left from the *O*(1) terms:

$$\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=1}^{\#tracks} A_{k,m} \int_0^{\Delta s_{k,m}} dsb_i(s) \frac{\partial \psi_{m,g,k}^{(0),n+1}}{\partial s} = 0 . \tag{4.63}$$

Using the relations from Eqs. (4.52), (4.53), and (4.55), Eq. (4.63) becomes the following:

$$\begin{aligned} &\sum_{g=1}^{G}\sum_{m=1}^{M} w_m \sum_{k=1}^{\#tracks} b_i(s) \sum_{e\in cell} \partial A_{k,m,e} \left(\vec{e}_n \bullet \vec{\Omega}_m \right) \psi_{m,g,k}^{(0),n+1}(s) \Big|_0^{\Delta s_{k,m}} - \\ &\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=1}^{\#tracks} A_{k,m} \int_0^{\Delta s_{k,m}} ds \vec{\nabla} b_i \bullet \vec{\Omega}_m \psi_{m,g,k}^{(0),n+1}(s) = 0 \end{aligned} . \tag{4.64}$$

Also using the result that the leading order intensity is isotropic from Eq. (4.45), and assuming the same tracks are used for opposing angles of the symmetric quadrature set for each group, the integral over *s* in the second term of Eq. (4.64) will cancel for opposing angles, therefore becoming zero. This cancellation leaves the following expression to be analyzed.

$$\sum_{g=1}^{G}\sum_{m=1}^{M} w_m \sum_{k=1}^{\#tracks} \sum_{e\in cell} \partial A_{k,m,e} b_i(s) \left(\vec{e}_n \bullet \vec{\Omega}_m \right) \psi_{m,g,k}^{(0),n+1}(s) \Big|_0^{\Delta s_{k,m}} = 0 \tag{4.65}$$

Only the edges of the cell that touch vertex v_i contribute to this expression because the PWL basis function is zero on other edges. Equation (4.65) essentially means that the weighted integral over the edges of a cell should yield a zero leading-order net current. These edge terms will bring in unknowns from neighboring cells due to these net currents.

Equation (4.65) couples all of the unknowns for all vertices in the problem and brings in the incident intensities on the boundary. For example, considering a vertex on the boundary of the problem, Figure 9 shows the unknowns in red that are included in the two equations corresponding to the two basis functions that are centered at one vertex equation on the boundary (shown in purple). With these coupled equations we have a system of unknowns and equations that is equal to the number of basis functions in the problem.

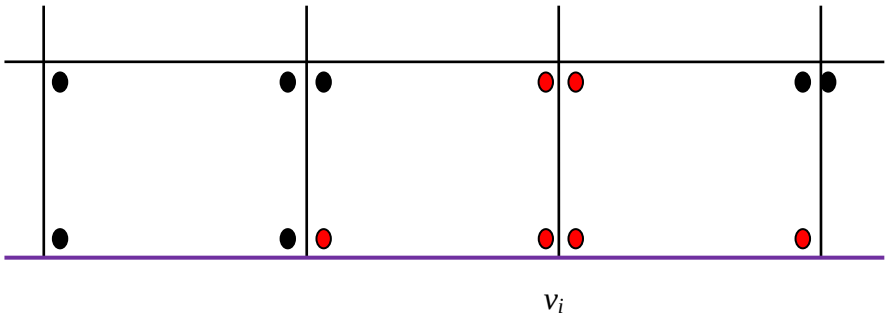


Figure 9. Unknowns in a Boundary Vertex Equation

We refer to the paper by Adams[40] for proof that with DFEM's, if the incident partial currents are continuous around the domain boundary, then the leading-order Planckian

unknowns are continuous at every vertex, including the interior and the boundary, and can be defined as a single unknown at each vertex. This proof is applicable to this PWL-LC method assuming a fine track spacing. Below we show that continuity can be lost if track spacing is not fine.

If we assume that the leading-order Planckian is continuous at each interior vertex, then differences of Planckians across interior surfaces vanish and the equation for a boundary vertex becomes the following equation.

$$\sum_{g=1}^{G}\left[\begin{array}{l}\displaystyle\sum_{\vec{e}_n\bullet\vec{\Omega}_m>0} w_m \sum_{e\in bdry}\sum_{k=1}^{\#\text{tracks on } e} \partial A_{k,m,e} b_{i,k}\left(\vec{e}_n\bullet\vec{\Omega}_m\right)\frac{1}{4\pi}B_g^{(0),n+1}(s) \\ \displaystyle -\sum_{\vec{e}_n\bullet\vec{\Omega}_m<0} w_m \sum_{e\in bdry}\sum_{k=1}^{\#\text{tracks on } e} \partial A_{k,m,e} b_{i,k}\left|\left(\vec{e}_n\bullet\vec{\Omega}_m\right)\right|\psi_{m,g,k,inc}^{n+1}(s)\end{array}\right]=0 \qquad (4.66)$$

Except for corners of the spatial domain, each boundary cell has only one boundary surface (referred to as an "edge"). Thus, the sum over edges vanishes. If the same tracks are used for opposing angles then the direction sums can be combined. Also, the Planckian is a PWL function and the incident partial current is assumed to be a PWL function as well. It follows (under the assumption mentioned above that the solution is continuous in the interior) that the following system is satisfied on each boundary cell surface.

$$\sum_{g=1}^{G}\sum_{\vec{e}_n\bullet\vec{\Omega}_m<0} w_m\left|\left(\vec{e}_n\bullet\vec{\Omega}_m\right)\right|\sum_{k=1}^{\#\text{tracks on } e}\partial A_{k,m,e} b_{i,k}\sum_{j=1}^{N} b_{j,k}\left[\frac{1}{4\pi}B_{g,j}^{(0),n+1}-\psi_{m,g,inc,j}^{n+1}\right]=0 \qquad (4.67)$$

Equation. (4.67) can be written in matrix notation by defining two matrices, $\left[M_e\right]$ and $\left[C_{m,e}\right]$, with the following elements,

$$\begin{aligned} C_{m,e,ij} &\equiv \sum_{k=1}^{\#\text{tracks on } e} \partial A_{k,m,e} b_{i,k} b_{j,k} \\ M_{e,ij} &\equiv \sum_{\vec{e}_n \bullet \vec{\Omega}_m < 0} w_m \left| \left(\vec{e}_n \bullet \vec{\Omega}_m \right) \right| C_{m,e,ij} \end{aligned} . \tag{4.68}$$

Rewriting Eq. (4.67) with these matrices, the following equation is found,

$$\frac{1}{4\pi} \left[M_e \right] \sum_{g=1}^{G} \vec{B}_g^{(0),n+1} = \sum_{\vec{e}_n \bullet \vec{\Omega}_m < 0} w_m \left| \left(\vec{e}_n \bullet \vec{\Omega}_m \right) \right| \left[C_{m,e} \right] \sum_{g=1}^{G} \vec{\psi}_{m,g,inc}^{n+1} , \tag{4.69}$$

where

$$\begin{aligned} \vec{B}_g^{(0),n+1} &= \begin{bmatrix} B_{g,1}^{(0),n+1} \\ B_{g,2}^{(0),n+1} \end{bmatrix}, \\ \vec{\psi}_{m,g,inc}^{n+1} &= \begin{bmatrix} \psi_{m,g,inc,1}^{n+1} \\ \psi_{m,g,inc,2}^{n+1} \end{bmatrix} \end{aligned} . \tag{4.70}$$

Here we have used "1" and "2" to represent the two PWL unknowns associated with the boundary cell-edge of a 2D problem. In 3D, the vectors would have a length equal to the number of vertices of the given boundary cell-face. Matrix inversion produces the equation for the leading-order Planckian on the boundary.

$$\sum_{g=1}^{G} \vec{B}_g^{(0),n+1} = 4\pi \left[M_e \right]^{-1} \sum_{\vec{e}_n \bullet \vec{\Omega}_m < 0} w_m \left| \left(\vec{e}_n \bullet \vec{\Omega}_m \right) \right| \left[C_{m,e} \right] \sum_{g=1}^{G} \vec{\psi}_{m,g,inc}^{n+1} \tag{4.71}$$

If the $\left[C_{m,e} \right]$ matrices are the same for all *m*, which is the case for fine track spacing, then Eq. (4.71) simplifies greatly and the leading-order solution satisfies a Marshak boundary condition at each boundary vertex,

$$\sum_{g=1}^{G} \vec{B}_g^{(0),n+1} = \sum_{\vec{e}_n \bullet \vec{\Omega}_m < 0} w_m \frac{4 \left| \left(\vec{e}_n \bullet \vec{\Omega}_m \right) \right|}{\rho} \sum_{g=1}^{G} \vec{\psi}_{m,g,inc}^{n+1} , \tag{4.72}$$

where,

$$\rho = \sum_{\vec{e}_n \bullet \vec{\Omega}_m < 0} w_m \frac{\left| \left(\vec{e}_n \bullet \vec{\Omega}_m \right) \right|}{\pi} \cong 1 \quad . \tag{4.73}$$

In this case, if the incident partial currents are continuous at all boundary vertices, then the leading-order Planckian (and thus the temperature) will also be continuous. However, if the $\left[C_{m,e} \right]$ matrices are different for different *m*, then Eq. (4.71) does not simplify and the leading-order solution in a boundary cell at a boundary vertex depends on the incident intensities at all of that cell's boundary vertices.

The leading-order solution in a neighboring boundary cell at the same boundary vertex depends on the incident intensities at all of its vertices, and thus the neighboring solutions at a given boundary vertex may be different. This means that continuity can be lost, even if incident partial currents are PWL and continuous, if track spacing is not fine. This loss of continuity without fine track spacing on a boundary surface is an issue that will be brought up later and has not been resolved in this research.

Next, the definition for the weighted first-order current from Eq. (4.59) is found. Multiplying Eq. (4.33) by $\frac{db_i}{ds}$ and a quadrature weight and summing over angles and groups yields the following $O(1)$ terms.

$$\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=1}^{\#tracks} A_{k,m} \int_0^{\Delta s_{k,m}} ds\vec{\Omega}_m \bullet \vec{\nabla} b_i(s) \left[\begin{array}{l} \dfrac{\partial \psi_{m,g,k}^{(0),n+1}(s)}{\partial s} + \\ \sigma_{a,g}^{(0)} \left(\psi_{m,g,k}^{(1),n+1}(s) - \dfrac{1}{4\pi} B_{g,k}^{(1),n+1}(s) \right) + \\ \sigma_{a,g}^{(1)} \left(\psi_{m,g,k}^{(0),n+1}(s) - \dfrac{1}{4\pi} B_{g,k}^{(0),n+1}(s) \right) \end{array} \right] = 0 \qquad (4.74)$$

The term multiplying $\sigma_{a,g}^{(1)}$ is zero based upon previous findings. Also, assuming the opacity is constant in the cell it can be divided and taken outside the integral. Doing so leads to the following equation:

$$\begin{array}{l} \sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=1}^{\#tracks} A_{k,m} \dfrac{1}{\sigma_{a,g}^{(0)}} \int_0^{\Delta s_{k,m}} ds\vec{\Omega}_m \bullet \vec{\nabla} b_i(s) \dfrac{\partial \psi_{m,g,k}^{(0),n+1}(s)}{\partial s} + \\ \sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=1}^{\#tracks} A_{k,m} \int_0^{\Delta s_{k,m}} ds\vec{\Omega}_m \bullet \vec{\nabla} b_i(s) \left(\psi_{m,g,k}^{(1),n+1}(s) - \dfrac{1}{4\pi} B_{g,k}^{(1),n+1}(s) \right) = 0 \end{array} . \qquad (4.75)$$

The Planckian can be expressed in terms of the PWL basis, therefore, Eq. (4.75) becomes the following:

$$\begin{array}{l} \sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=1}^{\#tracks} A_{k,m} \dfrac{1}{\sigma_{a,g}^{(0)}} \int_0^{\Delta s_{k,m}} ds\vec{\Omega}_m \bullet \vec{\nabla} b_i(s) \dfrac{\partial \psi_{m,g,k}^{(0),n+1}(s)}{\partial s} + \\ \sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=1}^{\#tracks} A_{k,m} \int_0^{\Delta s_{k,m}} ds\vec{\Omega}_m \bullet \vec{\nabla} b_i(s) \left(\psi_{m,g,k}^{(1),n+1}(s) - \dfrac{1}{4\pi} \sum_{j=1}^{N} B_{cell,g,j}^{(1),n+1} b_j(s) \right) = 0 \end{array} . \qquad (4.76)$$

With this substitution the integral over *s* in the Planckian term is only over basis function terms. Isolating this term from Eq. (4.76) and rearranging, the following expression is found.

$$-\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=1}^{\#tracks} A_{k,m} \frac{1}{4\pi} \sum_{j=1}^{N} B_{cell,g,j}^{(1),n+1} \int_0^{\Delta s_{k,m}} ds \vec{\Omega}_m \bullet \vec{\nabla} b_i(s) b_j(s) = 0 \tag{4.77}$$

Again assuming the same track is used for opposing angles in this integral for the symmetric quadrature set, this integral over *s* will cancel for opposing angles. Therefore this term becomes zero in Eq. (4.76) and the following terms are left.

$$\begin{aligned} &\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=1}^{\#tracks} A_{k,m} \int_0^{\Delta s_{k,m}} ds \vec{\Omega}_m \bullet \vec{\nabla} b_i(s) \psi_{m,g,k}^{(1),n+1}(s) = \\ &-\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=1}^{\#tracks} A_{k,m} \frac{1}{\sigma_{a,g}^{(0)}} \int_0^{\Delta s_{k,m}} ds \vec{\Omega}_m \bullet \vec{\nabla} b_i(s) \frac{\partial \psi_{m,g,k}^{(0),n+1}(s)}{\partial s} \end{aligned} \tag{4.78}$$

Next, substituting the expression for the leading-order intensity, and rearranging terms in the integral leads to the following equation.

$$\begin{aligned} &\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=1}^{\#tracks} A_{k,m} \int_0^{\Delta s_{k,m}} ds \vec{\Omega}_m \bullet \vec{\nabla} b_i(s) \left(\psi_{m,g,k}^{(1),n+1}(s)\right) = \\ &-\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=1}^{\#tracks} A_{k,m} \frac{1}{\sigma_{a,g}^{(0)}} \int_0^{\Delta s_{k,m}} ds \vec{\nabla} b_i(s) \bullet \vec{\Omega}_m \vec{\Omega}_m \bullet \frac{1}{4\pi} \vec{\nabla} B_{g,k}^{(0),n+1}(s) \end{aligned} \tag{4.79}$$

Rewriting the Planckian in terms of the PWL basis, Eq. (4.79) can be written as the following expression.

$$\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=1}^{\#tracks} A_{k,m} \int_0^{\Delta s_{k,m}} ds\vec{\Omega}_m \bullet \vec{\nabla} b_i(s)\left(\psi_{m,g,k}^{(1),n+1}(s)\right) =$$
$$-\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \frac{1}{4\pi}\sum_{j=1}^{N} B_{g,cell,j}^{(0),n+1} \sum_{k=1}^{\#tracks} A_{k,m} \frac{1}{\sigma_{a,g}^{(0)}} \int_0^{\Delta s_{k,m}} ds\vec{\nabla} b_i(s) \bullet \vec{\Omega}_m \vec{\Omega}_m \bullet \vec{\nabla} b_j(s) \tag{4.80}$$

This summation over tracks can be split into a summation over tracks on sides in the cell. Adding this summation and rearranging terms leads to the following expression.

$$\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=1}^{\#tracks} A_{k,m} \int_0^{\Delta s_{k,m}} ds\vec{\Omega}_m \bullet \vec{\nabla} b_i(s)\left(\psi_{m,g,k}^{(1),n+1}(s)\right) =$$
$$-\sum_{g=1}^{G} \frac{1}{4\pi}\frac{1}{\sigma_{a,g}^{(0)}} \sum_{l=1}^{\#sides}\sum_{j=1}^{N} B_{g,cell,j}^{(0),n+1} \sum_{m=1}^{M} w_m J_m \sum_{k=1}^{\#tracks\ on\ l} A_{kl,m} \int_0^{\Delta s_{kl,m}} ds\vec{\nabla} b_{i,l} \bullet \vec{\Omega}_m \vec{\Omega}_m \bullet \vec{\nabla} b_{j,l} \tag{4.81}$$

Equation (4.81) has used the fact that the gradient of each basis function is constant on each side of the cell, so the notation $\vec{\nabla} b_{j,l}$ represents the value of the gradient of the basis function on the l^{th} side. In this expression, the integral over *s* evaluates to the length of the track on this side. Therefore, a track volume on the side for a given angle can be defined as given in Eq. (4.82).

$$V_{l,m} \equiv J_m \sum_{k=1}^{\#tracks\ on\ l} A_{kl,m} \int_0^{\Delta s_{kl,m}} ds \tag{4.82}$$

A Rosseland mean opacity can be defined on a side. Recall that the gradient of the Planckian on a side is a constant for all positions on the side. With this in mind, we can rewrite the group summation in Eq. (4.81), where we have written the Planckian in terms of its gradient on a side.

$$\sum_{g=1}^{G}\frac{1}{\sigma_{a,g}^{(0)}}\vec{\nabla}B_{g,l}^{(0),n+1}=\sum_{g=1}^{G}\frac{ac}{\sigma_{a,g}^{(0)}}\frac{\left|\vec{\nabla}B_{g}^{(0),n+1}\right|_l}{\left|\vec{\nabla}T^{(0),n+1}\right|_l}\vec{\nabla}T^{(0),n+1}\Big|_l=$$
$$\sum_{g=1}^{G}\frac{ac}{\sigma_{a,g}^{(0)}}\frac{\left|\vec{\nabla}B_{g}^{(0),n+1}\right|_l}{\left|\vec{\nabla}\left(T^{(0),n+1}\right)^4\right|_l}\vec{\nabla}\left(T^{(0),n+1}\right)^4\Big|_l \tag{4.83}$$

Here we have taken advantage of the identities:

$$\vec{\nabla}B_{g,l}^{(0),n+1}=\frac{dB_{g,l}^{(0),n+1}}{dT_l^{(0),n+1}}\vec{\nabla}T^{(0),n+1}\Big|_l=\frac{dB_{g,l}^{(0),n+1}}{dT_l^{(0),n+1}}\frac{dT_l^{(0),n+1}}{d\left(T_l^{(0),n+1}\right)^4}\vec{\nabla}\left(T^{(0),n+1}\right)^4\Big|_l, \tag{4.84}$$

which show that all three gradient vectors point in the same direction. Expression (4.83) can be manipulated and simplified which leads to the following expression involving the Rosseland mean,

$$\sum_{g=1}^{G}\frac{1}{\sigma_{a,g}^{(0)}}\vec{\nabla}B_{g,l}^{(0),n+1}=\frac{ac}{\sigma_{R,l}^{(0)}}\vec{\nabla}\left(T^{(0),n+1}\right)^4\Big|_l, \tag{4.85}$$

where,

$$\sigma_{R,l}^{(0)}\equiv\frac{\sum_{g=1}^{G}\frac{1}{\sigma_{a,g}^{(0)}}\frac{\left|\vec{\nabla}B_{g}^{(0),n+1}\right|_l}{\left|\vec{\nabla}T^{(0),n+1}\right|_l}}{\sum_{g=1}^{G}\frac{\left|\vec{\nabla}B_{g}^{(0),n+1}\right|_l}{\left|\vec{\nabla}T^{(0),n+1}\right|_l}},$$
$$\vec{\nabla}\left(T^{(0),n+1}\right)^4\Big|_l=\left|\vec{\nabla}\left(T^{(0),n+1}\right)^4\right|_l\frac{\vec{\nabla}T^{(0),n+1}\Big|_l}{\left|\vec{\nabla}T^{(0),n+1}\right|_l} \tag{4.86}$$

Substituting this definition and the definition of Eq. (4.82) into Eq. (4.81), this equation becomes the following expression.

$$\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=1}^{\#tracks} A_{k,m} \int_0^{\Delta s_{k,m}} ds \vec{\Omega}_m \bullet \vec{\nabla} b_i(s)\left(\psi_{m,g,k}^{(1),n+1}(s)\right) = \\ -\frac{1}{4\pi}\sum_{l=1}^{\#sides}\frac{ac}{\sigma_{R,l}^{(0)}}\sum_{j=1}^{N}\left(T_{cell,j}^{(0),n+1}\right)^4 \vec{\nabla} b_{i,l} \bullet \sum_{m=1}^{M} w_m V_{l,m} \vec{\Omega}_m \vec{\Omega}_m \bullet \vec{\nabla} b_{j,l} \tag{4.87}$$

It is helpful to define a tensor as given in Eq. (4.88).

$$\vec{\vec{\tau}}_l \equiv \frac{\sum_{m=1}^{M} w_m V_{l,m} \vec{\Omega}_m \vec{\Omega}_m}{\frac{1}{4\pi}\sum_{m=1}^{M} w_m V_{l,m}} \tag{4.88}$$

For standard methods with a symmetric quadrature set, this term would yield $4\pi/3$ multiplied by the identity tensor because the cross terms cancel in the quadrature summation. However, for LC methods, the cross terms in this angular summation do not cancel because the track volumes on a side are different for each angle given a finite track spacing. Therefore, this tensor will be "contaminated" by other terms representing the contribution from these cross terms. Equation (4.89) gives the definition of Eq. (4.88) with these contamination terms.

$$\vec{\vec{\tau}}_l \equiv \frac{\sum_{m=1}^{M} w_m V_{l,m} \vec{\Omega}_m \vec{\Omega}_m}{\frac{1}{4\pi}\sum_{m=1}^{M} w_m V_{l,m}} \equiv \frac{4\pi}{3}\vec{\vec{I}} + \delta\vec{\vec{\tau}}_l \tag{4.89}$$

Putting this definition into Eq. (4.87) gives the following expression. From this expression, it easy to see that it yields a stiffness matrix with some contamination terms.

$$\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=1}^{\#tracks} A_{k,m} \int_0^{\Delta s_{k,m}} ds\vec{\Omega}_m \bullet \vec{\nabla} b_i(s)\left(\psi_{m,g,k}^{(1),n+1}(s)\right) =$$
$$-\frac{1}{4\pi}\sum_{l=1}^{\#sides}\frac{ac}{\sigma_{R,l}^{(0)}}\sum_{j=1}^{N}\left(T_{cell,j}^{(0),n+1}\right)^4 \vec{\nabla} b_{i,l} \bullet \sum_{m=1}^{M} w_m V_{l,m}\left[\frac{4\pi}{3}\vec{\vec{I}} + \delta\vec{\vec{\tau}}_l\right]\bullet\vec{\nabla} b_{j,l} \tag{4.90}$$

This stiffness matrix is defined in the expression given in Eq. (4.91). The temperature can also be written in vector form given in Eq. (4.92).

$$\left[K\right]_{ij,l} \equiv \sum_{m=1}^{M}\frac{1}{4\pi} w_m V_{l,m} \vec{\nabla} b_{i,l} \bullet \left[\vec{\vec{I}} + \frac{3}{4\pi}\delta\vec{\vec{\tau}}_l\right]\bullet\vec{\nabla} b_{j,l} \tag{4.91}$$

$$\left(\vec{T}_{cell}^{(0),n+1}\right)^4 = \begin{bmatrix} \left(T_{cell,1}^{(0),n+1}\right)^4 \\ \vdots \\ \left(T_{cell,j}^{(0),n+1}\right)^4 \end{bmatrix} \tag{4.92}$$

With this definition, the O(ε) current term for cells in the interior becomes the following.

$$\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=1}^{\#tracks} A_{k,m} \int_0^{\Delta s_{k,m}} ds\vec{\Omega}_m \bullet \vec{\nabla} b_i(s)\left(\psi_{m,g,k}^{(1),n+1}(s)\right) =$$
$$-\frac{1}{3}\sum_{l=1}^{\#sides}\frac{ac}{\sigma_{R,l}^{(0)}}\left[K\right]_l\left(\vec{T}_{cell}^{(0),n+1}\right)^4 \tag{4.93}$$

Putting the expression for the first-order current into Eq. (4.60) yields the following i^{th} equation for interior vertices.

$$\sum_{cells\in v_i}\left[\begin{array}{l}\sum_{m=1}^{M} w_m J_m \sum_{k=1}^{\#tracks} A_{k,m} \int_0^{\Delta s_{k,m}} dsb_i(s)\,\frac{a}{4\pi}\,\frac{\left(T_{PWL}^{(0),n+1}\right)^4-\left(T_{PWL}^{(0),n}\right)^4}{\Delta t^n}- \\ -\frac{1}{3}\sum_{l=1}^{\#sides}\frac{ac}{\sigma_{R,l}^{(0)}}[K]_l\left(\vec{T}_{cell}^{(0),n+1}\right)^4+ \\ \sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \frac{1}{4\pi}\sum_{k=1}^{\#tracks} A_{k,m}\int_0^{\Delta s_{k,m}} dsb_i(s)\begin{pmatrix} C_v\frac{T_{PWL}^{(0),n+1}-T_{PWL}^{(0),n}}{\Delta t^n} \\ -Q_{PWL}\end{pmatrix}\end{array}\right]=0 \qquad (4.94)$$

A track-based mass matrix can also be defined for the remaining terms in Eq. (4.94). Considering the last terms in this equation, substituting the PWL relations for the temperatures and source, the following expression is found.

$$\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \frac{1}{4\pi}\sum_{k=1}^{\#tracks} A_{k,m}\int_0^{\Delta s_{k,m}} dsb_i(s)\begin{pmatrix} C_v\frac{\sum_{j=1}^{N} T_{cell,j}^{(0),n+1}b_j(s)-\sum_{j=1}^{N} T_{cell,j}^{(0),n}b_j(s)}{\Delta t^n} \\ -\sum_{j=1}^{N} Q_{cell,j}b_j(s)\end{pmatrix} \qquad (4.95)$$

A track-based mass matrix term can now be defined as given in Eq. (4.96).

$$[A]_{ij}\equiv\frac{1}{4\pi}\sum_{l=1}^{\#sides}\sum_{m=1}^{M} w_m J_m \sum_{k=1}^{\#tracks\ on\ l} A_{kl,m}\int_0^{\Delta s_{kl,m}} dsb_i(s)b_j(s) \qquad (4.96)$$

Using this definition, expression (4.95) becomes the following expression, where the temperature and source coefficients have been written in vector form.

$$\begin{gathered}\sum_{g=1}^{G}\frac{C_v}{\Delta t^n}[A]\left(\vec{T}_{cell}^{(0),n+1}-\vec{T}_{cell}^{(0),n}\right)-\sum_{g=1}^{G}[A]\vec{Q}_{cell}, \\ \vec{T}_{cell}^{(0),n+1}=\begin{bmatrix}T_1^{(0),n+1}\\ \vdots \\ T_N^{(0),n+1}\end{bmatrix},\vec{T}_{cell}^{(0),n}=\begin{bmatrix}T_1^{(0),n}\\ \vdots \\ T_N^{(0),n}\end{bmatrix},\vec{Q}_{cell}=\begin{bmatrix}Q_1\\ \vdots \\ Q_N\end{bmatrix}\end{gathered} \qquad (4.97)$$

Putting this matrix definition into Eq. (4.94) leads to the i^{th} equation for an interior vertex as given in Eq. (4.98).

$$\sum_{cells \in v_i} \left[\begin{array}{l} \dfrac{a}{\Delta t^n}[A]\left(\left(\vec{T}_{cell}^{(0),n+1}\right)^4 - \left(\vec{T}_{cell}^{(0),n}\right)^4\right) - \dfrac{1}{3}\sum_{l=1}^{\#sides} \dfrac{ac}{\sigma_{R,l}^{(0)}}[K]_l \left(\vec{T}_{cell}^{(0),n+1}\right)^4 + \\ \sum_{g=1}^{G} \dfrac{C_v}{\Delta t^n}[A]\left(\vec{T}_{cell}^{(0),n+1} - \vec{T}_{cell}^{(0),n}\right) - \sum_{g=1}^{G}[A]\vec{Q}_{cell} \end{array} \right] = 0 \qquad (4.98)$$

This is a reasonable continuous FEM discretization of the correct equilibrium diffusion equation, with continuous PWL basis functions, if the leading-order temperatures are continuous. This continuity is achieved with fine track spacing and continuous incident partial currents on boundaries. If track spacing is coarse or incident currents are not continuous, then the leading-order solution is not continuous, and it is more difficult to interpret the accuracy of the discretization shown here. The reference by Adams[40] discusses this in more detail. This discussion and numerical experience suggests that even in problems for which the analysis cannot prove continuity, the solution in the interior of thick diffusive regions tends toward the solution of the continuous FEM equations, with discontinuities and other inaccuracies largely confined to boundary cells. This is tested in section 6 below.

Next, this balance equation is considered for a cell on the boundary on the problem. This portion of the analysis will determine the boundary conditions that go along with the interior equations (Eq. (4.98)). When considering cells on the boundary of the problem, the surface terms that previously canceled when analyzing the first-order current in the problem interior will need to be considered. Returning to Eq. (4.78), the first-order track-summed current will be defined as the following for ease of notation:

$$J_{tracks}^{(1),i} \equiv \sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=1}^{\#tracks} A_{k,m} \int_0^{\Delta s_{k,m}} ds \vec{\Omega}_m \bullet \vec{\nabla} b_i(s) \psi_{m,g,k}^{(1),n+1}(s). \quad (4.99)$$

With this definition, Eq. (4.78) can now be written as the following:

$$J_{tracks}^{(1),i} = -\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=1}^{\#tracks} A_{k,m} \frac{1}{\sigma_{a,g,k}^{(0)}} \int_0^{\Delta s_{k,m}} ds \vec{\Omega}_m \bullet \vec{\nabla} b_i(s) \frac{\partial \psi_{m,g,k}^{(0),n+1}(s)}{\partial s}. \quad (4.100)$$

The summation over tracks is split over sides and the derivative over *s* is rewritten to give Eq. (4.101).

$$J_{tracks}^{(1),i} = -\sum_{g=1}^{G} \frac{1}{\sigma_{a,g}^{(0)}} \sum_{l=1}^{\#sides}\sum_{m=1}^{M} w_m J_m \sum_{k=1}^{\#tracks\ on\ l} A_{kl,m} \int_0^{\Delta s_{kl,m}} ds \vec{\Omega}_m \bullet \vec{\nabla} b_i(s) \vec{\Omega}_m \bullet \vec{\nabla} \psi_{m,g,k}^{(0),n+1} \quad (4.101)$$

The leading-order intensity will now be written in the following form:

$$\psi_{m,g,k}^{(0),n+1}(s) = \psi_{m,g,k}^{(0),n+1}(s) - \frac{B_k^{(0),n+1}(s)}{4\pi} + \frac{B_k^{(0),n+1}(s)}{4\pi}. \quad (4.102)$$

Substituting this expression into Eq. (4.101), this new equation is considered for a cell that is on the boundary of the problem but with a basis function, b_i, centered at a non-boundary vertex.

$$J_{tracks}^{(1),i} = -\sum_{g=1}^{G} \frac{1}{\sigma_{a,g}^{(0)}} \sum_{l=1}^{\#sides}\sum_{m=1}^{M} w_m J_m \sum_{k=1}^{\#tracks\ on\ l} A_{kl,m} \rightarrow$$
$$\int_0^{\Delta s_{kl,m}} ds \vec{\Omega}_m \bullet \vec{\nabla} b_i(s) \vec{\Omega}_m \bullet \vec{\nabla} \left[\begin{array}{l} \psi_{m,g,k}^{(0),n+1}(s) - \\ \frac{B_k^{(0),n+1}(s)}{4\pi} + \\ \frac{B_k^{(0),n+1}(s)}{4\pi} \end{array} \right] \quad (4.103)$$

For a side in the cell that does not touch the boundary, the leading-order flux is isotropic and the previous expression simplifies to give the same stiffness matrix and Rosseland mean opacity that have been defined previously.

For sides of the boundary cell that do touch the problem boundary, the $\psi_{m,g,k}^{(0),n+1}(s) - \frac{B_k^{(0),n+1}(s)}{4\pi}$ term in Eq. (4.103) is not zero and must be explicitly considered, while the $+\frac{B_k^{(0),n+1}(s)}{4\pi}$ term contributes to the usual stiffness matrix. The result is:

$$J_{tracks}^{(1),i} = -\frac{1}{3}\sum_{l=1}^{\#\text{sides}} \frac{ac}{\sigma_{R,l}^{(0)}}\left[K\right]_l \left(\vec{T}_{cell}^{(0),n+1}\right)^4 - \sum_{g=1}^{G}\frac{1}{\sigma_{a,g}^{(0)}}\sum_{l\in bdry}\sum_{m=1}^{M} w_m J_m \sum_{k=1}^{\#\text{tracks on } l_{dry}} A_{kl,m}\int_0^{\Delta s_{kl,m}} ds\vec{\nabla}b_i(s)\bullet\vec{\Omega}_m\vec{\Omega}_m\bullet\vec{\nabla}\left[\begin{array}{l}\psi_{m,g,k}^{(0),n+1}(s) - \\ \frac{B_{g,k}^{(0),n+1}(s)}{4\pi}\end{array}\right]. \quad (4.104)$$

The basis function, b_i, in this expression is not centered at a vertex on this side, so the gradient of this basis function on this side is a constant multiplied by the normal of the boundary edge. This property is denoted in Eq. (4.105).

$$J_{tracks}^{(1),i} = -\frac{1}{3}\sum_{l=1}^{\#\text{sides}} \frac{ac}{\sigma_{R,l}^{(0)}}\left[K\right]_l \left(\vec{T}_{cell}^{(0),n+1}\right)^4 - \sum_{g=1}^{G}\frac{1}{\sigma_{a,g}^{(0)}}\sum_{l\in bdry}\sum_{m=1}^{M} w_m J_m \sum_{k=1}^{\#\text{tracks on } l_{bdry}} \left\{\begin{array}{l}\left|\vec{\nabla}b_{il}\right| A_{kl,m}\left(-\vec{e}_l\bullet\vec{\Omega}_m\right)... \\ \int_0^{\Delta s_{kl,m}} ds\vec{\Omega}_m\bullet\vec{\nabla}\left[\begin{array}{l}\psi_{m,g,k}^{(0),n+1}(s) - \\ \frac{B_{g,k}^{(0),n+1}(s)}{4\pi}\end{array}\right]\end{array}\right\} \quad (4.105)$$

The following variable is defined to denote the outward edge normal dotted with the direction variable as given in Eq. (4.106).

$$\mu_{m,l} \equiv \vec{e}_l \bullet \vec{\Omega}_m \tag{4.106}$$

With this notation, Eq. (4.105) becomes the following expression. The integral over *s* of $\frac{\partial \psi}{\partial s}$ can be evaluated and turns into a summation over the beginning and ending position of the tracks on this side. Also, the Jacobian and area of each track along the edge of the side is related to the length along this edge as was given in Eq. (4.55).

$$J_{tracks}^{(1),i} = -\frac{1}{3}\sum_{l=1}^{\#\text{sides}} \frac{ac}{\sigma_{R,l}^{(0)}} \left[K\right]_l \left(\vec{T}_{cell}^{(0),n+1}\right)^4 +$$

$$\sum_{g=1}^{G} \frac{1}{\sigma_{a,g}^{(0)}} \sum_{l \in bdry} \sum_{m=1}^{M} w_m \sum_{e \in l_{bdry}} \sum_{k=1}^{\#\text{tracks on } e} \left|\vec{\nabla} b_{il}\right| \partial A_{kle,m} \left|\mu_{m,l}\right| \mu_{m,l} \left[\left(\psi_{m,g,k}^{(0),n+1} - \frac{B_{g,k}^{(0),n+1}}{4\pi} \right)_{s=\Delta s_{kl,m}} - \left(\psi_{m,g,k}^{(0),n+1} - \frac{B_{g,k}^{(0),n+1}}{4\pi} \right)_{s=0} \right] \tag{4.107}$$

For tracks entering this side from an edge of the side that is not on the boundary, it is known that the leading-order flux is isotropic, therefore these tracks cancel and only the boundary edge of this side is left.

$$J_{tracks}^{(1),i} = -\frac{1}{3}\sum_{l=1}^{\#\text{sides}} \frac{ac}{\sigma_{R,l}^{(0)}} \left[K\right]_l \left(\vec{T}_{cell}^{(0),n+1}\right)^4 +$$

$$\sum_{g=1}^{G} \frac{1}{\sigma_{a,g}^{(0)}} \sum_{l \in bdry} \sum_{m=1}^{M} w_m \sum_{k=1}^{\#\text{tracks on } e_{l,bdry}} \left|\vec{\nabla} b_{il}\right| \partial A_{kle,m} \left|\mu_{m,l}\right| \mu_{m,l} \left[\left(\psi_{m,g,k}^{(0),n+1} - \frac{B_{g,k}^{(0),n+1}}{4\pi} \right)_{s=\Delta s_{kl,m}} - \left(\psi_{m,g,k}^{(0),n+1} - \frac{B_{g,k}^{(0),n+1}}{4\pi} \right)_{s=0} \right] \tag{4.108}$$

It is also known that for the tracks exiting on this boundary edge, the intensity is isotropic and therefore these terms cancel as shown in Eq. (4.108). With this cancellation only the incoming terms on this boundary edge remain as given in Eq. (4.109), where the incident intensity is given and the Planckian on the boundary (which satisfies a Marshak boundary condition given in Eq. (4.72)) is written as $B_{g,M,bdry,k}$.

$$J_{tracks}^{(1),i} = -\frac{1}{3}\sum_{l=1}^{\#\text{sides}} \frac{ac}{\sigma_{R,l}^{(0)}}\left[K\right]_l \left(\vec{T}_{cell}^{(0),n+1}\right)^4 + \sum_{g=1}^{G}\frac{1}{3\sigma_{a,g}^{(0)}}\sum_{l\in bdry}\left|\vec{\nabla}b_{il}\right|\left[\sum_{\mu_{m,l}<0} w_m \sum_{k=1}^{\#\text{tracks on e}_{l,\text{bdry}}} \partial A_{kle,m} 3\mu_{m,l}^2 \left[\psi_{m,g,k,l,e,inc}^{n+1} - \frac{B_{g,M,bdry,k}^{(0),n+1}}{4\pi}\right]\right] \tag{4.109}$$

In order to be able to write the boundary term in a way that allows interpretation, the definition of the weighted first-order current involving the stiffness matrix is returned to and integration by parts is applied for the side that is on the boundary of the problem in a cell. Rewriting Eq. (4.109) with the term the stiffness matrix came from, the following terms are present.

$$\begin{aligned}
&\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=1}^{\#tracks} A_{k,m} \int_0^{\Delta s_{k,m}} ds\vec{\Omega}_m \bullet \vec{\nabla} b_i\left(s\right)\psi_{m,g,k}^{(1),n+1}\left(s\right) = \\
&-\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=1}^{\#tracks} A_{k,m}\frac{1}{\sigma_{a,g}^{(0)}} \int_0^{\Delta s_{k,m}} ds\vec{\Omega}_m \bullet \vec{\nabla} b_i\left(s\right)\frac{\partial\psi_{m,g,k}^{(0),n+1}\left(s\right)}{\partial s} \\
&+\sum_{g=1}^{G}\frac{1}{3\sigma_{a,g}^{(0)}}\sum_{l\in bdry}\left|\vec{\nabla}b_{il}\right|\left[\sum_{\mu_{m,l}<0} w_m \sum_{k=1}^{\#\text{tracks on e}_{\text{bdry}}} \partial A_{kle,m} 3\mu_{m,l}^2 \left[\psi_{m,g,k,l,e,inc}^{n+1} - \frac{B_{g,M,bdry,k}^{(0),n+1}}{4\pi}\right]\right]
\end{aligned} \tag{4.110}$$

The leading-order intensity in the *s* integral is isotropic and can be written in terms of the leading-order Planckian.

$$\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=1}^{\#tracks} A_{k,m} \int_0^{\Delta s_{k,m}} ds\vec{\Omega}_m \bullet \vec{\nabla} b_i(s)\psi_{m,g,k}^{(1),n+1}(s) =$$
$$-\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=1}^{\#tracks} A_{k,m} \frac{1}{\sigma_{a,g}^{(0)}} \int_0^{\Delta s_{k,m}} ds\vec{\Omega}_m \bullet \vec{\nabla} b_i(s) \frac{1}{4\pi} \frac{\partial B_g^{(0),n+1}(s)}{\partial s} \tag{4.111}$$
$$+\sum_{g=1}^{G} \frac{1}{3\sigma_{a,g}^{(0)}} \sum_{l\in bdry} \left|\vec{\nabla} b_{il}\right| \left[\sum_{\mu_{m,l}<0} w_m \sum_{k=1}^{\#\text{tracks on } e_{bdry}} \partial A_{kle,m} 3\mu_{m,l}^2 \left[\psi_{m,g,k,l,e,inc}^{n+1} - \frac{B_{g,bdry}^{(0),n+1}}{4\pi} \right] \right]$$

The *s* integral is integrated by parts.

$$\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=1}^{\#tracks} A_{k,m} \int_0^{\Delta s_{k,m}} ds\vec{\Omega}_m \bullet \vec{\nabla} b_i(s)\psi_{m,g,k}^{(1),n+1}(s) =$$
$$-\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=1}^{\#tracks} A_{k,m} \frac{1}{\sigma_{a,g}^{(0)}} \left\{ \left[\vec{\Omega}_m \bullet \vec{\nabla} b_i \frac{B_g^{(0),n+1}}{4\pi} \right]_0^{\Delta s_{k,m}} - \int_0^{\Delta s_{k,m}} ds \frac{\partial^2 b_i}{\partial s^2} \frac{B_g^{(0),n+1}}{4\pi} \right\} \tag{4.112}$$
$$+\sum_{g=1}^{G} \frac{1}{3\sigma_{a,g}^{(0)}} \sum_{l\in bdry} \left|\vec{\nabla} b_{il}\right| \left[\sum_{\mu_{m,l}<0} w_m \sum_{k=1}^{\#\text{tracks on } e_{bdry}} \partial A_{kle,m} 3\mu_{m,l}^2 \left[\psi_{m,g,k,l,e,inc}^{n+1} - \frac{B_{g,bdry}^{(0),n+1}}{4\pi} \right] \right]$$

The first term in the second line of Eq. (4.112) turns into a summation over the edges of this side which means the relation defined in Eq. (4.55) can be applied and the gradient of the basis function turns into a constant multiplied by the edge normal. The Planckian in the cell has been denoted by an interior (int) subscript and the Planckian on the boundary edge has been denoted by a Marshak boundary subscript (*M,bdry*). The summation over angles can be split into incoming and outgoing angles on the boundary side.

$$\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=1}^{\#tracks} A_{k,m} \int_0^{\Delta s_{k,m}} ds \vec{\Omega}_m \bullet \vec{\nabla} b_i(s) \psi_{m,g,k}^{(1),n+1}(s) =$$

$$-\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=1}^{\#\text{tracks}\notin e_{bdry}} A_{k,m} \frac{1}{\sigma_{a,g}^{(0)}} \left\{ \left[\vec{\Omega}_m \bullet \vec{\nabla} b_i \frac{B_{g,\text{int},k}^{(0),n+1}}{4\pi} \right]_0^{\Delta s_{k,m}} - \int_0^{\Delta s_{k,m}} ds \frac{\partial^2 b_i}{\partial s^2} \frac{B_{g,\text{int},k}^{(0),n+1}}{4\pi} \right\}$$

$$-\sum_{g=1}^{G} \sum_{l\in bdry} \left| \vec{\nabla} b_{il} \right| \frac{1}{\sigma_{a,g}^{(0)}} \left[\begin{array}{l} \sum_{\mu_{m,l}<0} w_m \sum_{k=1}^{\#\text{tracks}\in e_{bdry}} \partial A_{kle,m} \left|\mu_{m,l}\right| \mu_{m,l} \left[\frac{B_{g,\text{int},k}^{(0),n+1}}{4\pi} - \frac{B_{g,M,bdry,k}^{(0),n+1}}{4\pi} \right] - \\ \sum_{\mu_{m,l}>0} w_m \sum_{k=1}^{\#\text{tracks}\in e_{bdry}} \partial A_{kle,m} \left|\mu_{m,l}\right| \mu_{m,l} \left[\frac{B_{g,M,bdry,k}^{(0),n+1}}{4\pi} - \frac{B_{g,\text{int},k}^{(0),n+1}}{4\pi} \right] \end{array} \right]$$

$$+\sum_{g=1}^{G} \frac{1}{3\sigma_{a,g}^{(0)}} \sum_{l\in bdry} \left| \vec{\nabla} b_{il} \right| \left[\sum_{\mu_{m,l}<0} w_m \sum_{k=1}^{\#\text{tracks}\in e_{bdry}} \partial A_{kle,m} 3\mu_{m,l}^2 \left[\psi_{m,g,k,l,e,inc}^{n+1} - \frac{B_{g,M,bdry}^{(0),n+1}}{4\pi} \right] \right] \tag{4.113}$$

If the same tracks are used for opposing angles in the symmetric quadrature set, the incoming and outgoing angles can be combined leading to the following expression.

$$\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=1}^{\#tracks} A_{k,m} \int_0^{\Delta s_{k,m}} ds \vec{\Omega}_m \bullet \vec{\nabla} b_i(s) \psi_{m,g,k}^{(1),n+1}(s) =$$

$$-\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=1}^{\#\text{tracks}\notin e_{bdry}} A_{k,m} \frac{1}{\sigma_{a,g}^{(0)}} \left\{ \left[\vec{\Omega}_m \bullet \vec{\nabla} b_i \frac{B_{g,\text{int},k}^{(0),n+1}}{4\pi} \right]_0^{\Delta s_{k,m}} - \int_0^{\Delta s_{k,m}} ds \frac{\partial^2 b_i}{\partial s^2} \frac{B_{g,\text{int},k}^{(0),n+1}}{4\pi} \right\}$$

$$-\sum_{g=1}^{G} \frac{1}{3\sigma_{a,g}^{(0)}} \sum_{l\in bdry} \left| \vec{\nabla} b_{il} \right| \left[\sum_{\mu_{m,l}<0} w_m \sum_{k=1}^{\#\text{tracks}\in e_{bdry}} \partial A_{kle,m} 3\mu_{m,l}^2 \left[\frac{2B_{g,\text{int},k}^{(0),n+1}}{4\pi} - \frac{2B_{g,M,bdry,k}^{(0),n+1}}{4\pi} \right] \right] \tag{4.114}$$

$$+\sum_{g=1}^{G} \frac{1}{3\sigma_{a,g}^{(0)}} \sum_{l\in bdry} \left| \vec{\nabla} b_{il} \right| \left[\sum_{\mu_{m,l}<0} w_m \sum_{k=1}^{\#\text{tracks}\in e_{bdry}} \partial A_{kle,m} 3\mu_{m,l}^2 \left[\psi_{m,g,k,l,e,inc}^{n+1} - \frac{B_{g,M,bdry,k}^{(0),n+1}}{4\pi} \right] \right]$$

Combining terms leads to the expression given in Eq. (4.115).

$$\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=1}^{\#tracks} A_{k,m} \int_0^{\Delta s_{k,m}} ds\vec{\Omega}_m \bullet \vec{\nabla} b_i(s)\psi_{m,g,k}^{(1),n+1}(s) =$$

$$-\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=1}^{\#\text{tracks}\not\in e_{\text{bdry}}} A_{k,m} \frac{1}{\sigma_{a,g}^{(0)}} \left\{ \left[\vec{\Omega}_m \bullet \vec{\nabla} b_i \frac{B_{g,\text{int},k}^{(0),n+1}}{4\pi} \right]_0^{\Delta s_{k,m}} - \int_0^{\Delta s_{k,m}} ds \frac{\partial^2 b_i}{\partial s^2} \frac{B_{g,\text{int},k}^{(0),n+1}}{4\pi} \right\}$$

$$-\sum_{g=1}^{G} \frac{1}{3\sigma_{a,g}^{(0)}} \sum_{l\in bdry} \left| \vec{\nabla} b_{il} \right| \left[\sum_{\mu_{m,l}<0} w_m \sum_{k=1}^{\#\text{tracks}\in e_{\text{bdry}}} \partial A_{kle,m} 3\mu_{m,l}^2 \frac{2B_{g,\text{int},k}^{(0),n+1}}{4\pi} \right] \quad (4.115)$$

$$+\sum_{g=1}^{G} \frac{1}{3\sigma_{a,g}^{(0)}} \sum_{l\in bdry} \left| \vec{\nabla} b_{il} \right| \left[\sum_{\mu_{m,l}<0} w_m \sum_{k=1}^{\#\text{tracks}\in e_{\text{bdry}}} \partial A_{kle,m} 3\mu_{m,l}^2 \left[\psi_{m,g,k,l,e,inc}^{n+1} + \frac{B_{g,M,bdry,k}^{(0),n+1}}{4\pi} \right] \right]$$

A "variational" boundary Planckian term can be defined for a boundary edge as given in Eq. (4.116).

$$-\sum_{g=1}^{G} \frac{1}{3\sigma_{a,g}^{(0)}} \sum_{l\in bdry} \left| \vec{\nabla} b_{il} \right| \sum_{\mu_{m,l}<0} w_m \sum_{k=1}^{\#\text{tracks}\in e_{\text{bdry}}} \partial A_{kle,m} 3\mu_{m,l}^2 \left[\frac{2B_{g,\text{int},k}^{(0),n+1}}{4\pi} - \frac{2B_{g,V,bdry,k}^{(0),n+1}}{4\pi} \right] \equiv$$

$$-\sum_{g=1}^{G} \frac{1}{3\sigma_{a,g}^{(0)}} \sum_{l\in bdry} \left| \vec{\nabla} b_{il} \right| \sum_{\mu_{m,l}<0} w_m \sum_{k=1}^{\#\text{tracks}\in e_{\text{bdry}}} \partial A_{kle,m} 3\mu_{m,l}^2 \left[\begin{array}{l} \frac{2B_{g,\text{int},k}^{(0),n+1}}{4\pi} - \\ \frac{B_{g,M,bdry,k}^{(0),n+1}}{4\pi} - \psi_{m,g,k,l,e,inc}^{n+1} \end{array} \right] \quad (4.116)$$

Since the gradient of each basis function that is not on this boundary edge has the same constant value on this edge, Eq. (4.116) shows that only an average value on each edge is required for a boundary condition. With this in mind, a variational average Planckian on a boundary edge is defined using Eq. (4.116) and is written as the following expression.

$$\sum_{g=1}^{G} \frac{1}{3\sigma_{a,g}^{(0)}} \left\langle B_{g,V,bdry}^{(0),n+1} \right\rangle_{e_{bdry}} \equiv$$
$$\frac{\sum_{g=1}^{G} \frac{1}{3\sigma_{a,g}^{(0)}} \sum_{\mu_{m,l}<0} w_m \sum_{k=1}^{\#\text{tracks}\in e_{\text{bdry}}} \partial A_{kle,m} 3\mu_{m,l}^2 \left[\frac{B_{g,M,bdry,k}^{(0),n+1}}{4\pi} + \psi_{m,g,k,l,e,inc}^{n+1} \right]}{\frac{1}{4\pi} \sum_{\mu_{m,l}<0} w_m \sum_{k=1}^{\#\text{tracks}\in e_{\text{bdry}}} \partial A_{kle,m} 3\mu_{m,l}^2} \tag{4.117}$$

Given fine track spacing, this variational average Planckian is approximately equal to the expected and proper boundary condition as given in Eq. (4.118).

$$\sum_{g=1}^{G} \frac{1}{3\sigma_{a,g}^{(0)}} \left\langle B_{g,V,bdry}^{(0),n+1} \right\rangle_{e_{bdry}} \approx \sum_{g=1}^{G} \frac{1}{3\sigma_{a,g}^{(0)}} \sum_{\mu_{m,l}<0} w_m \left(3\mu_{m,l}^2 + \frac{2|\mu_{m,l}|}{\rho} \right) \left\langle \psi_{m,g,inc}^{n+1} \right\rangle_{e_{bdry}} \tag{4.118}$$

A group-average Rosseland mean opacity can be defined using this average variational Planckian on the boundary edge similar to Eq. (4.86). This average opacity is defined in Eq. (4.119).

$$\sigma_{R,l_{bdry}}^{(0)} \equiv \frac{\sum_{g=1}^{G} \frac{1}{\sigma_{a,g}^{(0)}} \frac{\left| \vec{\nabla} \left\langle B_{g,V,bdry}^{(0),n+1} \right\rangle \right|_{l_{bdry}}}{\left| \vec{\nabla} T^{(0),n+1} \right|_{l_{bdry}}}}{\sum_{g=1}^{G} \frac{\left| \vec{\nabla} \left\langle B_{g,V,bdry}^{(0),n+1} \right\rangle \right|_{l_{bdry}}}{\left| \vec{\nabla} T^{(0),n+1} \right|_{l_{bdry}}}} \tag{4.119}$$

Here the gradient term involving $\left\langle B_{g,V,bdry}^{(0),n+1} \right\rangle$ is defined to use the "variational" B values, as defined in Eq. (4.118), for each PWL basis function centered at a vertex on the boundary face, while using the unknown leading-order B values at interior vertices. Using this definition, the average variational boundary Planckian is written as the following.

$$\sum_{g=1}^{G}\frac{1}{\sigma_{a,g}^{(0)}}\vec{\nabla}\left\langle B_{g,V,bdry}^{(0),n+1}\right\rangle_{l_{bdry}}=\frac{ac}{\sigma_{R,e_{bdry}}^{(0)}}\vec{\nabla}\left(T_V^{(0),n+1}\right)^4\Big|_{l_{bdry}} \tag{4.120}$$

With this defined gradient of the variational Planckian on the boundary edge, the simplifications that were performed on Eq. (4.112) can be undone to arrive at an expression for the weighted first-order current that yields the stiffness matrix that has been previously defined and the Rosseland mean opacity defined in Eq. (4.119). Note that the temperature in this expression actually includes a small contamination by the Marshak boundary Planckian term which comes from the second term obtained from integration by parts in Eq. (4.112). For a cell on the boundary, this weighted first-order current can be written as given in Eq. (4.121).

$$J_{tracks,bdrycell}^{(1),i}=-\frac{1}{3}\sum_{l=1}^{\#sides}\frac{ac}{\sigma_{R,l}^{(0)}}\left[K\right]_l\left(\tilde{\vec{T}}_{cell,V}^{(0),n+1}\right)^4 \tag{4.121}$$

Therefore, it can be seen from Eq. (4.121) and Eq. (4.118) that the boundary condition seen by interior cells is different from the Marshak boundary condition defined previously.

Based upon this weighted first-order current analysis, it has been shown that for cells on the boundary, this first-order current must be split into a summation over the sides in the cell. For a side edge that does not touch a boundary vertex, the previously defined stiffness matrix term in Eq. (4.91) holds. For side edges that touch a boundary vertex, but are not on the actual problem boundary edge, the Marshak boundary condition is included. For a side edge that is on the problem boundary edge, the incident angular intensity is included. Also, it is noted that the boundary condition seen by

interior vertices is different from the Marshak boundary condition and is more accurate for this diffusion limit. Therefore, Eq. (4.98) has been fully defined for all cells in the problem beyond the first time step and the boundary condition seen by the interior cells is different from the standard Marshak condition that the leading-order solution satisfies.

Finally, the leading-order solution must be defined on the initial time step. The analog of Eq. (4.98) for the first time step is given in Eq. (4.122). It is assumed in this equation that the initial temperature condition can be written in terms of the PWL basis.

$$\sum_{cells\in v_i}\left[\begin{array}{l}\dfrac{a}{\Delta t^n}[A]\left(\vec{T}_{cell}^{(0),n+1}\right)^4 - \\ \displaystyle\sum_{g=1}^{G}\frac{1}{4\pi c}\sum_{m=1}^{M}w_m J_m \sum_{k=1}^{\#tracks} A_{k,m}\int_0^{\Delta s_{k,m}} dsb_i(s)\frac{\psi_{m,g,k}^{i}}{\Delta t^n} - \\ \dfrac{1}{3}\displaystyle\sum_{l=1}^{\#sides}\frac{ac}{\sigma_{R,l}^{(0)}}[K]_l\left(\vec{T}_{cell}^{(0),n+1}\right)^4 + \\ \displaystyle\sum_{g=1}^{G}\frac{C_v}{\Delta t^n}[A]\left(\vec{T}_{cell}^{(0),n+1}-\vec{T}_{cell}^{i}\right)-\sum_{g=1}^{G}[A]\vec{Q}_{cell}\end{array}\right]=0 \qquad (4.122)$$

If the initial intensity is Planckian at the initial temperature, then the following would be true and Eq. (4.122) reverts to Eq. (4.98).

$$\begin{array}{l}\displaystyle\sum_{g=1}^{G}\frac{1}{4\pi c}\sum_{m=1}^{M}w_m J_m \sum_{k=1}^{\#tracks} A_{k,m}\int_0^{\Delta s_{k,m}} dsb_i(s)\frac{\psi_{m,g,k}^{i}}{\Delta t^n} = \\ \dfrac{1}{4\pi c\Delta t^n}\displaystyle\sum_{m=1}^{M}w_m J_m \sum_{k=1}^{\#tracks} A_{k,m}\int_0^{\Delta s_{k,m}} dsb_i(s)ac\sum_{j=1}^{N}\left(T_{cell,j}^{i}\right)^4 b_j(s) = \\ \dfrac{a}{\Delta t^n}[A]\left(\vec{T}_{cell}^{i}\right)^4\end{array} \qquad (4.123)$$

In general the following initial condition (denoted with the superscript *n*) must be met in each cell to make Eq. (4.122) reproduce Eq. (4.98) for the first time step.

$$\left\{\frac{a}{4\pi}[A]\left(\vec{T}_{cell}^{(0),n}\right)^4+\sum_{g=1}^{G}C_v[A]\left(\vec{T}_{cell}^{(0),n}\right)\right\}_{n=0}=$$
$$\frac{1}{4\pi c}\sum_{g=1}^{G}\sum_{m=1}^{M}w_m J_m\sum_{k=1}^{\#tracks}A_{k,m}\int_0^{\Delta s_{k,m}}dsb_i(s)\frac{\psi_{m,g,k}^{i}}{\Delta t^n}+ \qquad (4.124)$$
$$\sum_{g=1}^{G}C_v[A]\left(\vec{T}_{cell}^{i}\right)$$

This discretized initial condition is very similar to the exact initial condition from the analytic analysis.

This analysis of the PWL-LC method has shown that the leading-order PWL-LC solution satisfies a reasonable discretization of the equilibrium diffusion equation with associated appropriately discretized boundary conditions and defined initial conditions. This diffusion equation given in Eq. (4.98), also defined on the boundary and the initial time step, is analogous to a PWL continuous FEM discretization in space with fully-implicit differencing in time. Note that the assumption of fine track spacing is needed to enforce continuity of unknowns at vertices.

4.4 PWL-LC Steady-State Asymptotic Analysis in the Diffusion Limit

This section analyzes the discretized steady-state radiative-transfer equations in the thick diffusion limit using the PWL-LC method for spatial discretization in (x, y). This analysis will be very similar to the previous time-dependent analysis in section 4.3. The details of this analysis can be found in section A.1 of Appendix A.

The resulting balance equation that the leading-order PWL-LC method satisfies is given is shown here for discussion.

$$\sum_{cells\in v_i}\left[-\frac{1}{3}\sum_{l=1}^{\#sides}\frac{ac}{\sigma_{R,l}^{(0)}}\left[K\right]_l\left(\vec{T}_{cell}^{(0)}\right)^4-\sum_{g=1}^{G}\left[A\right]\vec{Q}_{cell}\right]=0 \tag{4.125}$$

This balance equation is a reasonable discretized equilibrium diffusion equation and is similar to a steady-state DFEM discretized diffusion equation. This balance equation holds for all cells with an added term for boundary cells that brings in the incident flux on sides of the cell that touch the problem boundary. It can be seen that in the limit of steady-state, the time-dependent discretized diffusion equation limits to this steady-state equation with this PWL-LC method. Taking the limit of Eq. (4.98) as the time-step becomes large yields the following expression.

$$\lim_{\Delta t^n\to\infty}\left[\sum_{cells\in v_i}\left[\begin{array}{l}\frac{a}{\Delta t^n}\left[A\right]\left(\left(\vec{T}_{cell}^{(0),n+1}\right)^4-\left(\vec{T}_{cell}^{(0),n}\right)^4\right)-\\ \frac{1}{3}\sum_{l=1}^{\#sides}\frac{ac}{\sigma_{R,l}^{(0)}}\left[K\right]_l\left(\vec{T}_{cell}^{(0)}\right)^4+\\ \sum_{g=1}^{G}\frac{C_v}{\Delta t^n}\left[A\right]\left(\vec{T}_{cell}^{(0),n+1}-\vec{T}_{cell}^{(0),n}\right)-\sum_{g=1}^{G}\left[A\right]\vec{Q}_{cell}\end{array}\right]\right]=0 \tag{4.126}$$

Evaluating this limit, it is clear that Eq. (4.126) limits to the steady-state result given in Eq. (4.125) for this PWL-LC method. It is also clear that this steady-state equation is analogous to the analytic steady-state result in the diffusion limit. The boundary conditions that held for the time-dependent discretized LC diffusion equation also hold for this PWL-LC method in steady-state. Therefore, the boundary condition seen by the interior cells is defined by a variational boundary condition (given in Eq. (4.120)), which is different than the leading-order Marshak condition satisfied on the boundary.

4.5 STLC Asymptotic Analysis in the Diffusion Limit

This section summarizes an asymptotic analysis of the scaled discretized radiative transfer equations given in Eqs. (4.27) and (4.28) using the STLC method with various source approximations. The STLC discretized transport equation for a given track, *k*, is shown in Eq. (4.127).

$$\frac{\partial \psi_{m,g,k}(s)}{\partial s} + \frac{\sigma_{a,g}(T(s))}{\varepsilon}\psi_{m,g,k}(s) = \frac{\sigma_{a,g}(T(s))}{\varepsilon}\frac{1}{4\pi}B_{g,k}(T(s)) \qquad (4.127)$$

It is noted that the material energy equation does not couple spatial points. Therefore, in the following source approximations, this property will be preserved. The dependence of the temperature in time will be chosen based upon the source approximation used.

Due to the scaling applied to give the thick diffusion limit, the time axis of the (*x*, *y*, *vt*) cell is stretched, therefore the number of tracks starting on the spatial cell boundary in a given time step becomes infinite as $\varepsilon \to 0$. This stretching and increase in number of tracks is the same as what was outlined in the diffusion limit analysis of the STLC method in $(x, t)^3$. There are four types of tracks that can be defined for an (*x*, *y*, *vt*) cell. These four types of tracks in a cell are defined as the following:

1) tracks entering and exiting out of a spatial boundary,
2) tracks entering and exiting from the two time boundaries,
3) tracks entering on a time boundary and exiting out of a spatial boundary,
4) track entering on a spatial boundary and exiting out of a time boundary.

As $\varepsilon \to 0$, the term $\frac{c\Delta t_n}{\varepsilon}$ becomes large, so the space-time cell is dominated by tracks of type 1. The tracks of types 3 and 4 will make up an $O(\varepsilon)$ fraction of the total tracks in a

cell because their volume is $O(\varepsilon)$ of the total volume of the *x*-*y*-*vt* cell. Tracks of type 2 should vanish given a fixed quadrature set with no directions along the *z*-axis and a sufficiently small ε because they are a significantly smaller fraction of cell volume, so they do not play a role in the following analysis.

The analysis in the subsections below proceeds with the previously postulated power series expansions of the unknowns as well as the assumed symmetric quadrature set and its previously outlined properties. Each subsection uses a different functional form for the source approximation in the radiative transfer equations based upon the approximations outlined in section 3.3.

4.5.1 PWC-STLC

The analysis in this section proceeds with the source in a space-time cell being approximated as constant in time and space. Again, the details of this analysis can be found in section A.2 of Appendix A. The results of this detailed analysis show that this method fails in the thick diffusion limit. The following equation is found to govern the PWC-STLC leading-order intensity.

$$\sum_{g=1}^{G}\sum_{e\in\text{cell}}\left[\begin{array}{l}\displaystyle\sum_{\vec{e}_n\bullet\vec{\Omega}_m>0} w_m \sum_{k\in\text{type1}} \partial A_{k,m,e}\left(\vec{e}_n\bullet\vec{\Omega}_m\right)\frac{B^{(0)}_{g,cell}}{4\pi} \\ \displaystyle -\sum_{\vec{e}_n\bullet\vec{\Omega}_m<0} w_m \sum_{k\in\text{type1}} \partial A_{k,m,e}\left|\vec{e}_n\bullet\vec{\Omega}_m\right|\frac{B^{(0)}_{g,neighbor}}{4\pi}\end{array}\right]=0 \qquad (4.128)$$

From this equation, it can be seen that the equation for a cell will include the leading-order Planckian of the given cell and all of its neighboring cells. Therefore, for all *J* cells, there is a set of *J* equations with *J* unknowns where each of these equations is linearly independent. These equations are a discretization of the Laplacian operator

(possibly an inaccurate one, depending on the mesh) that is invertible. Therefore, the leading-order scalar intensity for this PWC-STLC method is determined by the spatial grid and the boundary conditions of the problem (which enter in place of the neighbor terms in Eq. (4.128)) and does not depend on any physical properties in this thick diffusion limit. This result is expected based upon past research[21,40] and shows that this PWC-STLC method fails in this thick diffusion limit.

Although the steady-state PWC-LC thick diffusion limit analysis is not explicitly included in this dissertation, the results for this PWC-STLC apply to the PWC-LC method as well. Therefore, the PWC-LC method fails in the thick diffusion limit because the leading-order solution only depends on boundary conditions.

4.5.2 PWL-C-STLC

The analysis in this section proceeds with the source in a space-time cell being approximated as PWL in space and constant in time. Details of this analysis are given in section A.3 of Appendix A. The resulting diffusion equation satisfied by the leading-order PWL-C-STLC solution is given in Eq. (4.129).

$$\sum_{cell\in v_i}\left[\begin{array}{l}\dfrac{a}{\Delta t^n}\left(\left[M\right]^{n+1}\left(\vec{T}_{cell}^{(0),n+1/2}\right)^4-\left[M\right]^n\left(\vec{T}_{cell}^{(0),n-1/2}\right)^4\right)- \\ \dfrac{1}{3}\sum_{l=1}^{\#\text{sides}}\dfrac{ac}{\sigma_{R,l}^{(0)}}\left[K\right]_l\left(\vec{T}_{cell}^{(0),n+1/2}\right)^4+ \\ \sum_{g=1}^{G}\dfrac{C_v}{\Delta t^n}\left[A\right]\left(\vec{T}_{cell}^{(0),n+1/2}-\vec{T}_{cell}^{(0),n-1/2}\right)-\sum_{g=1}^{G}\left[A\right]\vec{Q}_{cell}\end{array}\right]=0\,. \qquad (4.129)$$

This equation is very similar to a fully implicit time-differenced and PWLD spatially discretized diffusion equation. Note that due to the use of finite track spacing, the track-based areas of the beginning and ending time boundaries of an *x-y-vt* cell are usually

different, and this causes the M matrices to differ at time n and $n+1$, in general, in the equation above. This is the reason why this method may oscillate around a steady state solution when considering a time-dependent problem in the steady-state limit.

Overall this analysis has shown that the leading-order solution of the PWL-C-STLC method satisfies a reasonably discretized diffusion equation that is similar to a PWLD discretized diffusion equation with fully-implicit time-differencing. The main differences between these two diffusion equations come from the track-based volume and area summations performed in the cell. The boundary condition seen by the interior cells is the same as was found from the analysis of the PWL-LC method which is expected and this boundary condition is more accurate than the Marshak condition that occurs at the boundary of the problem.

4.5.3 PWL-L-STLC

The analysis in this section proceeds with the source in a cell being approximated as PWL in space with one added basis that is linear in time as defined in Eq. (3.23). The details of this analysis can be found in section A.4 of Appendix A. The resulting balance equation that the leading-order solution satisfies is given in Eq. (4.130), defined for $i = 1,\ldots,N$,

$$\sum_{cells\in v_i}\left[\begin{array}{l}\frac{a}{\Delta t^n}A^{n+1}\left[M\right]^{n+1}\left[\left(\vec{T}_{cell}^{(0),n+1/2}\right)^4+\left(\vec{T}_z^{(0),t}\right)^4\right]-\\ \frac{a}{\Delta t^n}A^{n}\left[M\right]^{n}\left[\left(\vec{T}_{cell}^{(0),n-1/2}\right)^4+\left(\vec{T}_z^{(0),n}\right)^4-\left(\vec{T}_z^{(0),n-1/2}\right)^4\right]-\\ \frac{1}{3}\sum_{l=1}^{\#sides}\frac{ac}{\sigma_{R,l}^{(0)}}\left[K\right]_l\left(\vec{T}_{cell}^{(0),n+1/2}\right)^4+\\ \sum_{g=1}^{G}\frac{C_v}{\Delta t^n}\left[A\right]\left(\vec{T}_{cell}^{(0),n+1/2}-\vec{T}_{cell}^{(0),n}+\vec{T}_z^{(0),t}\right)-\\ \sum_{g=1}^{G}\left[A\right]\vec{Q}_{cell}\end{array}\right]=0 \qquad (4.130)$$

where,

$$\begin{array}{l}\left(\vec{T}_z^{(0),t}\right)^4\equiv\left(\vec{T}_z^{(0),n+1}\right)^4-\left(\vec{T}_z^{(0),n+1/2}\right)^4,\\ \vec{T}_z^{(0),t}\equiv\vec{T}_z^{(0),n+1}-\vec{T}_z^{(0),n+1/2}\end{array}. \qquad (4.131)$$

This balance equation along with the defined temperature at the center of the space-time cell given in Eq. (4.132) and the unphysical result for the time-slope of the Planckian given in Eq. (4.133), form the system of N+2 unknowns on a cell (where N is the number of vertices in the spatial cell) that the leading-order solution satisfies.

$$\begin{array}{l}\sum_{m=1}^{M}w_m J_m\sum_{k=type1}A_{k,m}\int_0^{\Delta s_{k,m}}ds\sum_{j=1}^{N}b_j T_{cell,j}^{(0),n+1/2}\equiv\\ T_z^{(0),n+1/2}\sum_{m=1}^{M}w_m J_m\sum_{k=type1}A_{k,m}\int_0^{\Delta s_{k,m}}ds\sum_{j=1}^{N}b_j\end{array} \qquad (4.132)$$

$$ac\sum_{e\in cell}\left[\rho_3 A_e\left(T_z^{t,cell,(0)}\right)^4-\rho_3 A_e\left(T_z^{t,neigh,(0)}\right)^4\right]=0 \qquad (4.133)$$

The "neighbor" terms in Eq. (4.133) are defined in terms of the time-slope of the incident partial current on boundary surfaces, which is what couples the leading-order time-slope terms to the boundary conditions. Without the extra time-slope terms in Eq.

(4.130), it resembles a fully-implicit DFEM discretized balance equation. These extra time-slope terms can make the solution to a given problem more or less accurate than a fully-implicit method. Since these time slope terms are not determined by anything physical, they will most likely lead to a less accurate solution. Note that if the given boundary condition is PWC in time, these extra time slope terms go to zero to leading order.

Overall it has been found that this PWL-L-STLC method produces a diffusion equation that is similar to a fully-implicit discretized diffusion equation in the diffusion limit with an added unphysical time-slope term. The same unphysical condition governing the time-slope of the Planckian has been found for this (x, y, t) PWL-L-STLC method as it was in (x, t)[3]. This flaw in the time-slope will cause this method to not perform well for certain problems in the thick diffusion limit and it is not recommended to use this method for most highly diffusive time-dependent problems.

4.5.4 PPWL-STLC

In this method, the source in a cell is approximated as PWL in space with each spatial-basis coefficient being linear in time, therefore leading to 2N unknowns per cell where N is the number of vertices of the polygonal spatial cell. With this basis, the PPWL-STLC scaled transport equation for a cell can be written using the least-squares procedure as given in Eq. (4.134), for $i = 1,\ldots, 2N$.

$$\sum_{m=1}^{M} w_m J_m \sum_{k=1}^{\#\text{ tracks}} A_{k,m} \int_0^{\Delta s_{k,m}} b_i(s) \left[\frac{\partial \psi_{m,g,k}}{\partial s} + \frac{\sigma_{a,g}}{\varepsilon} \psi_{m,g,k} - \frac{\sigma_{a,g}}{\varepsilon} \frac{1}{4\pi} B_{g,k} \right] ds = 0 \quad (4.134)$$

The temperature for this source approximation will be considered to be linear and discontinuous. Therefore, the temperature at a given spatial point in a time step can be written in terms of the temperatures at the beginning and end of the time step, using the linear basis in time as given in Eq. (4.135),

$$T_j(t) = T_j\,\alpha_n(t) + T_{j+N}\alpha_{n+1}(t), \qquad j = 1,\ldots,N, \tag{4.135}$$

where α_n and α_{n+1} represent the linear basis functions in time given in Eq. (4.136).

$$\begin{aligned} \alpha_n(t) &= \frac{t^{n+1} - t}{\Delta t} \\ \alpha_{n+1}(t) &= \frac{t - t^n}{\Delta t} = 1 - \alpha_n(t) \end{aligned} \tag{4.136}$$

The scaled material energy equation that holds for any spatial point is given in Eq. (4.137).

$$\varepsilon C_v \frac{\partial T_j(t)}{\partial t} = \sum_{g=1}^{G} \frac{1}{4\pi} \frac{\sigma_{a,g}}{\varepsilon} \left[\phi_{g,j}(t) - B_{g,j}(t) \right] + \varepsilon Q_j \tag{4.137}$$

The details of this analysis can be found in section A.5 of Appendix A. The main results of this analysis are that the leading-order PPWL-STLC solution satisfies a set of balance equations given in Eqs. (4.138) and (4.139) for $i = 1{:}2N$, that hold for all cells given the first-order weighted current on boundary cells defined in Eqs. (A.257) and (A.258) and the initial condition given by Eq. (A.261).

$$\sum_{cell\in v_i}\left[\begin{array}{l}\dfrac{a}{\Delta t^n}[A_0]\left[\begin{array}{l}\left(\left(\vec{T}_{cell}^{(0),n+1}\right)^{4,avg}+\left(\vec{T}_{cell}^{(0),n+1}\right)^{4,slope}\right)-\\ \left(\left(\vec{T}_{cell}^{(0),n}\right)^{4,avg}+\left(\vec{T}_{cell}^{(0),n}\right)^{4,slope}\right)\end{array}\right]+\\ \dfrac{C_v}{\Delta t^n}\sum_{g=1}^{G}[A_0]\left[\begin{array}{l}\left(\vec{T}_{cell}^{(0),avg,n+1}+\vec{T}_{cell}^{(0),slope,n+1}\right)-\\ \left(\vec{T}_{cell}^{(0),avg,n}+\vec{T}_{cell}^{(0),slope,n}\right)\end{array}\right]-\\ \dfrac{1}{3}\sum_{l=1}^{\#sides}\dfrac{ac}{\sigma_{R,l}^{(0)}}[K]_l\left(\vec{T}_{cell}^{(0),n+1}\right)^{4,avg}-\sum_{g=1}^{G}[A_0]\left(\vec{Q}_{cell}^{avg}\right)\end{array}\right]=0 \qquad (4.138)$$

$$\sum_{cells\in v_i}\left[\begin{array}{l}\dfrac{a}{\Delta t^n}[A_0]\left[\begin{array}{l}\left(\left(\vec{T}_{cell}^{(0),n+1}\right)^{4,avg}+\left(\vec{T}_{cell}^{(0),n+1}\right)^{4,slope}\right)+\\ \left(\left(\vec{T}_{cell}^{(0),n}\right)^{4,avg}+\left(\vec{T}_{cell}^{(0),n}\right)^{4,slope}\right)-2\left(\vec{T}_{cell}^{(0)}\right)^{4,avg}\end{array}\right]+\\ \dfrac{C_v}{\Delta t^n}\sum_{g=1}^{G}[A_0]\left[\begin{array}{l}\left(\vec{T}_{cell}^{(0),avg,n+1}+\vec{T}_{cell}^{(0),slope,n+1}\right)+\\ \left(\vec{T}_{cell}^{(0),avg,n}+\vec{T}_{cell}^{(0),slope,n}\right)-2\vec{T}_{cell}^{(0),avg}\end{array}\right]-\\ \dfrac{1}{9}\sum_{l=1}^{\#sides}[K]_l\left\{\begin{array}{l}\dfrac{ac}{\sigma_{R,l}^{(0),n+1}}\left(\vec{T}_{cell}^{(0),n+1}\right)^{4,slope}+\\ \left(\dfrac{ac}{\sigma_{R,l}^{(0),n+1}}-\dfrac{ac}{\sigma_{R,l}^{(0)}}\right)\left(\vec{T}_{cell}^{(0),n+1}\right)^{4,avg}\end{array}\right\}+\\ \sum_{g=1}^{G}[A_0]\dfrac{1}{3}\vec{Q}_{cell}^{slope}\end{array}\right]=0 \qquad (4.139)$$

These balance equations form a set of equations consistent with an analogous PPWL-DFEM discretized diffusion equation that can be solved for the 2*N* temperature unknowns. Treating the time variable as linear for each spatial position should provide an accurate solution for most time-dependent problems in the thick diffusion limit on arbitrary polygons and polyhedra. As was the case with the finite-differenced PWL-LC method (section 4.3 above), without fine track spacing, loss of continuity occurs, even

with continuous incident partial currents on the boundary. However, in practice this loss of continuity appears to affect only the boundary cells, with the PPWL-STLC solution in the interior of a thick diffusive region remaining accurate and essentially continuous. See the results in section 6 below for more details.

4.6 PPWL-LC Method in X-Y-Z

The analysis that has been done in the previous section for the PWL-LC method can also be extended to a PPWL-LC method for prism cells in (x, y, z). This prismatic geometry is used in many deterministic nuclear reactor models. From section 3.4, the basis for this PPWL-LC is PWL in (x, y) with each $N/2$ (x, y) basis being linear in z, where N is the number of vertices in the 3D cell. The asymptotic analysis for this PPWL-LC method would be similar to the PWL-LC method with some parallels to the PPWL-STLC method analysis. It should be noted that the computational work required for this PPWL-LC method is much more tractable than the general PWL-LC method, not only due to being able to take advantage of the sets of tracks in the z-dimension with the same geometry, but also because calculations only need to be done on $N/2$ sides instead of $6N$ sides.

5. SOLVING THE TRANSPORT EQUATION WITH PWL-LC AND STLC

This section presents the implementation logistics for solving the transport equation using the PWL-LC and STLC methods. It explains the basic algorithms and solution procedure for each of these methods as well as an explanation of the implementation of the PWL-LC and STLC methods in PDT at Texas A&M University.

5.1 Algorithms

This section outlines the basic algorithms for solving the transport equation with the LC and STLC methods that are used in PDT. The algorithms that will be described below assume a cell-based transport sweep is applied for each angular direction. A traditional cell-based transport sweep refers to an inversion of the streaming plus collision terms in the transport equation in which the problem is solved starting with the incoming boundary to the outgoing boundary in the direction of particle travel, given a previous iterate for the collisional source. The scheduling of the transport sweep is dependent on the quadrature set and geometry of the problem. This sweep schedule is determined by a built-in scheduler in PDT, which will not be discussed in this work. It will be noted that LC methods offer the potential to use track-based sweeping algorithms instead of cell-based sweeping algorithms, and will be briefly outlined in the recommendations section.

5.1.1 Newton Problems

First, the basic algorithms for solving a neutron transport problem will be outlined briefly. This algorithm is independent of the spatial and time discretization methods chosen to solve a problem. It is also independent of the chosen iterative solver

routine, such as Richardson iteration, GMRES, Conjugate Gradient, etc[41]. The solving algorithm outlined below begins in PDT after the problem input has been read and the setup of geometry, materials, quadrature, energy structure, and discretization and solver types has been performed. The execution of the solving algorithm in PDT begins with the call to the PDT_execute() routine in driver.cc. Figure 10 shows a brief outline of the path this execute call follows.

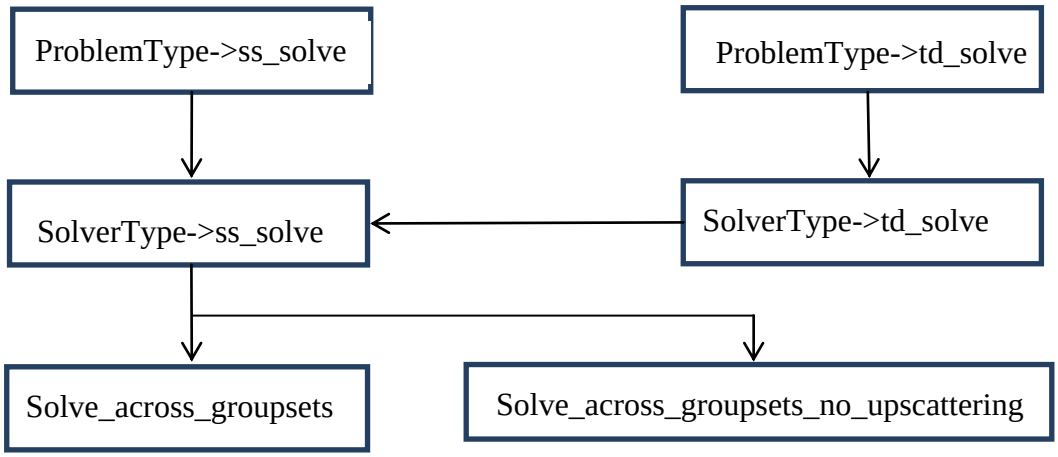


Figure 10. PDT Execution up to Solving

A groupset in PDT refers to a set of energy groups and angles that have been defined by the input to be grouped together for ease of computation and to coincide with the physics of the problem.

The details of the iteration algorithms implemented in PDT are not given in this work[42]. It is noted that the LC and STLC methods can both use standard iteration schemes such as Richardson, Conjugate Gradient, and GMRES[41] to solve the transport

equation to a given tolerance. The convergence criterion, based upon a given tolerance, is different for each iteration scheme but usually involves the calculation of the norm of a residual. The inequality that defines convergence on the norm of a residual is given in Eq. (5.1), where *l* denotes the iteration number, *j* denotes the element, *g* denotes the energy group, and *n* denotes the scattering moment.

$$\frac{\left\|\phi^{l+1}-\phi^{l}\right\|_{L2}}{\left\|\phi^{0}\right\|_{L2}}=\frac{\sqrt{\sum_{j}^{\#\text{elements}}\sum_{g=1}^{G}\sum_{n=1}^{\#\text{moments}}\left(\phi_{j,g,n}^{l+1}-\phi_{j,g,n}^{l}\right)^{2}}}{\sqrt{\sum_{j}^{\#\text{elements}}\sum_{g=1}^{G}\sum_{n=1}^{\#\text{moments}}\left(\phi_{j,g,n}^{0}\right)^{2}}}<tol \tag{5.1}$$

Besides a convergence tolerance on the residual from these iteration schemes, the solution can also be converged point-wise by comparing the maximum relative point-wise difference between successive solutions to a given tolerance. The inequality used to define relative point-wise convergence of the solution is given in Eq. (5.2) for the flux, where *l* denotes the iteration number and *j* denotes the element. This inequality must hold for all energy groups and scattering moments.

$$\frac{\left|\phi_{j}^{l+1}-\phi_{j}^{l+1}\right|}{\max\left(\left|\phi_{j}^{l+1}\right|,\left|\phi_{j}^{l}\right|\right)}<tol\left|\phi_{j}^{l+1}\right| \tag{5.2}$$

Pseudocode for the point-wise convergence check for a within-groupset solve on one processor is given in Figure 11. The point-wise convergence check across groupsets is very similar with an added loop over groupsets.

Pointwise Flux Local Convergence Check:
Converged = 1;
Max_PW_change = 0;
Loop over cells
  Loop over Elements (j=1:Nelem)
    Loop over groups in groupset (g)
      Loop over angular moments (m)
        Maxv = Max($\left|\phi_{j,g,m}^{l+1}\right|$, $\left|\phi_{j,g,m}^{l}\right|$)
        Max_PW_change = Max(Max_PW_change, $\left|\phi_{j,g,m}^{l+1}-\phi_{j,g,m}^{l}\right|/\text{Maxv}$)
        IF ($\left|\phi_{j,g,m}^{l+1}-\phi_{j,g,m}^{l}\right| > tol\left|\phi_{j,g,0}^{l+1}\right|$)
          Converged = 0
        End
      End
    End
  End

Figure 11. Point-wise Local Convergence Check on Scalar Flux

Next, the algorithm for the PWC-LC and PWL-LC methods using cell-based sweeps is considered for a steady-state problem. Consider a steady-state neutronics problem for either (x, y) or (x, y, z) geometry. The system that is solved for each cell in this problem involves the least-squares procedure given in Eq. (2.10) and Eq. (2.38). Note that the PWC-LC method can also use these equations, but with only one basis function equal to unity, to find the average flux in a cell. The set of equations found through the least-squares procedure can be written in matrix notation. This is shown below for the (x, y, z) equations. Rearranging Eq. (2.38) leads to the following expression, where g refers to one energy group:

$$\begin{aligned}\sum_{m=1}^{M} w_m \sum_{k=1}^{\#\text{tracks}} \Delta\omega_{k,m}\Delta u_{k,m} \int_0^{\Delta s_{k,m}} ds b_{cell,i}(s)\frac{1}{4\pi}\sum_{j=1}^{N}\phi_{g,cell,j}^{SS,l+1} b_{cell,j}(s) = \\ \sum_{m=1}^{M} w_m \sum_{k=1}^{\#\text{tracks}} \Delta\omega_{k,m}\Delta u_{k,m} \int_0^{\Delta s_{k,m}} ds b_{cell,i}(s)\Psi_{m,g,k}^{l}(s),\ i=1...N\end{aligned} \quad . \tag{5.3}$$

Recall the angular flux along a track is defined as given in Eq. (2.31) and depends on the total source in the cell, which uses the scalar flux from the previous iteration as denoted by the *l* superscript. The basis functions and the scalar flux coefficients can be written as vectors as given in Eq. (5.4) and Eq. (5.5).

$$\vec{b} \equiv \begin{bmatrix} b_{cell,i}(s) \\ \vdots \\ b_{cell,N}(s) \end{bmatrix} \tag{5.4}$$

$$\vec{\phi}_{g,cell}^{SS,l+1} \equiv \begin{bmatrix} \phi_{g,cell,i}^{SS,l+1}(s) \\ \vdots \\ \phi_{g,cell,N}^{SS,l+1}(s) \end{bmatrix} \tag{5.5}$$

The following system is found by substituting these vectors into Eq. (5.3).

$$\begin{aligned}\sum_{m=1}^{M} w_m \sum_{k=1}^{\#\text{tracks}} \Delta\omega_{k,m}\Delta u_{k,m} \int_0^{\Delta s_{k,m}} ds \frac{1}{4\pi}\vec{b}\vec{b}^{T}\vec{\phi}_{g,cell}^{SS,l+1} = \\ \sum_{m=1}^{M} w_m \sum_{k=1}^{\#\text{tracks}} \Delta\omega_{k,m}\Delta u_{k,m} \int_0^{\Delta s_{k,m}} ds \vec{b}\Psi_{m,g,k}^{l}(s)\end{aligned} \tag{5.6}$$

The following matrices and vectors can be defined for this system.

$$\left[M_{LC}\right] \equiv \sum_{m=1}^{M} w_m \sum_{k=1}^{\#\text{tracks}} \Delta\omega_{k,m}\Delta u_{k,m} \int_0^{\Delta s_{k,m}} ds \frac{1}{4\pi}\vec{b}\vec{b}^{T} \tag{5.7}$$

$$\vec{T}_{LC} \equiv \sum_{m=1}^{M} w_m \sum_{k=1}^{\#\,\text{tracks}} \Delta\omega_{k,m} \Delta u_{k,m} \int_0^{\Delta s_{k,m}} ds \vec{b} \Psi^{l}_{m,g,k}(s) \tag{5.8}$$

With these defined matrices, Eq. (5.6) becomes the following system of equations for each cell.

$$\left[M_{LC}\right] \vec{\phi}^{SS,l+1}_{g,cell} = \vec{T}_{LC} \tag{5.9}$$

This system is solved with Gaussian Elimination for each energy group on each cell to find the least-squares scalar flux at the current iteration. The details of how the integration over *s* is performed along each track to contribute to the matrix and vector values are given in the "sweepchunk" implementation sections below. Performing a transport sweep allows for the accumulation of terms in the matrices and right-hand sides given above. Therefore, the steady-state LC and STLC methods require one matrix inversion for each energy group per cell in one transport sweep. This number is less than the number required for a PWLD method in PDT which is one matrix inversion for each angle and group per cell in one transport sweep.

Solving a time-dependent LC problem requires defining a time discretization scheme. For the time-dependent LC method outlined in section 2.3 (termed the "FD-P' method), a theta scheme is used to define the angular flux. This family of schemes ranges from fully explicit, with $\theta = 0$, to fully implicit, with $\theta = 1$. The angular flux at the midpoint of a time step with this scheme can be written as given in Eq. (2.12), reiterated below.

$$\psi_m^{n+1/2}(\vec{r}) = \theta\psi_m^{n+1}(\vec{r}) + (1-\theta)\psi_m^n(\vec{r}) \tag{5.10}$$

Each time step requires solution of a steady-state LC problem with a modified total source and modified total cross section, as shown in section 2.3. Therefore the same least-squares algorithm defined above is used to find the scalar flux at the end of each transport sweep during each time step. In order to form the initial condition for the next time step, a consistent representation of the angular flux from the previous time step must be found. If using the PWL-LC method, a PWL form of the angular flux at the end of a time step must be found to use for ψ_m^n in the next time step. This can be done easily by using the matrices defined during the sweep of a cell. Since in the FD-P method, a cell's angular flux on time-step boundaries is assumed to be represented in terms of the basis chosen, it can be found by the same least-squares procedure used for the scalar flux. Equation (5.11) shows the representation of the angular flux at the end of a time step in terms of the chosen basis for the time-dependent LC method.

$$\psi_m^{n+1} = \sum_{j=1}^{N} \psi_{m,j}^{n+1} b_{cell,j} \tag{5.11}$$

The coefficients in this representation can again be found with the least-squares procedure for this LC method as given in Eq. (5.12).

$$\sum_{k=1}^{\#\text{ tracks}} J_m A_{k,m} \int_0^{\Delta s_{k,m}} ds \vec{b}\vec{b}^T \vec{\psi}_m^{n+1} = \sum_{k=1}^{\#\text{ tracks}} J_m A_{k,m} \int_0^{\Delta s_{k,m}} ds \vec{b} \Psi_{m,k}(s) \tag{5.12}$$

The same matrices defined previously but for only one angle can be defined here as well.

$$\left[M_{mLC}\right] \equiv J_m \sum_{k=1}^{\#\text{ tracks}} A_{k,m} \int_0^{\Delta s_k} ds \vec{b}\vec{b}^T \tag{5.13}$$

$$\vec{T}_{mLC} \equiv J_m \sum_{k=1}^{\#\text{ tracks}} A_{k,m} \int_0^{\Delta s_k} ds \vec{b} \Psi_{m,g,k}(s) \tag{5.14}$$

With these matrices, the least-squares procedure to find the angular flux at the end of a time step can be written as given in Eq. (5.15).

$$\left[M_{mLC}\right] \vec{\psi}_m^{n+1} = \vec{T}_{mLC} \tag{5.15}$$

This system is solved for each cell within the sweep for a current time step and the angular flux values are stored to use in the next time step. Note that this finite-differenced LC method requires a matrix inversion for each angle and group per cell, like the PWLD methods, in order to form the projected angular flux at the end of the time step. It still requires the one matrix inversion per group per cell to find the least-squares flux as well.

The algorithm to find the least-squares flux with the STLC method is similar to the LC algorithm. Each time step is solved separately. The least-squares procedure for the PPWL-STLC method is given in Eq. (3.25). This equation can hold for any STLC method by changing the basis used. Note that this equation has the same form as Eq. (5.3). Therefore, the same matrix system can be defined and solved as given in Eq. (5.9) by accumulating values through a transport sweep of the current time step. For the STLC methods, the scalar flux values found correspond to the basis used in the source approximation. Therefore, for the PPWL-STLC method these scalar flux values are

found at the spatial corners of a cell at the beginning and end of the time step (which form the corners of the *x*-*y*-*vt* cell).

This least-squares flux, from the LC and STLC methods, is iterated upon using an iteration procedure such as Richardson, GMRES, or Conjugate Gradient. The total source that is included in the representation of the angular flux along a track uses the flux from the previous iteration. Therefore, Eq. (5.9) can be rewritten to account for the iteration step, *l*.

$$\left[M_{LC}\right]\vec{\phi}_{g,cell}^{SS,(l+1)} = \vec{T}_{LC}^{(l)} \tag{5.16}$$

5.1.2 Thermal Radiation Problems

The algorithm used in PDT to solve a radiation transport problem with an LC or STLC method is now briefly outlined. A radiative transport problem is usually time-dependent and requires an outer iteration on the temperature of the material. Figure 12 shows brief pseudocode for the basic algorithm used for solving LC and STLC radiation transport problems in PDT. The same sweepchunks for a neutronics problem for the LC and STLC methods are used to solve for the absorption rate density for a given temperature (because these sweepchunks solve for the scalar intensity in a cell).

The PWC-STLC method uses the same RadSolver as other methods ,with some slight modifications to accommodate the different time discretization, because the discretized radiation transport equations with this method are the same as finite-differenced LC radiation transport equations as was seen in the analysis of section 4.5.1. The PPWL-STLC method uses a separate RadSolver to solve the radiative transfer

equations. The detailed equations for the PPWL-STLC radiation transport solver as well as the implementations of both RadSolvers are given in Appendix B.

```
Initialize values (T, φ, ARD, etc.)
For n=1:max_time_steps
        Set local T^(n+1/2), T^(n+1) by extrapolation
        While !PW_Temp_convergence
                Calculate source values
                Perform steady-state solve for ARD (until converged)
                Update T using ARD
                Check for PW_Temp_convergence
        End
        Update values (T, ψ_m) for next time step
End
```

Figure 12. Pseudocode for LC and STLC Radiative Transfer Problems

5.2 Implementation

This section presents how the LC and STLC algorithms are implemented in PDT. In the current implementation using the PDT-PTTL version of the code, these methods both use cell-based transport sweeps. The scheduling of these cell-based sweeps will not be addressed in this paper. A brief summary of how tracks are identified and related in PDT is presented first.

For each quadrature direction, rays are given a global ID in the problem in order to define the tracks that pass through each cell. In (x, y) the ID of each ray is given by one integer index denoted by the letter k for each quadrature direction denoted by the

letter *m*. Equation (5.17) shows the relationship between this index and the ω position of a ray.

$$\omega = k\Delta\omega_{k,m} \tag{5.17}$$

Using this relation and the coordinate system relation previously defined, the equation of a ray in this domain can be defined as given in Eq. (5.18). In this equation, the origin of the rotated (u, ω) coordinate system can be offset from the origin of the (x, y) coordinate system by an amount denoted by x_0 and y_0.

$$y = \left(x - x_0\right)\tan\gamma_m + \frac{k\Delta\omega_{k,m}}{\left|\cos\gamma_m\right|} + y_0 = \left(x - x_0\right)\frac{\eta_m}{\mu_m} + k\Delta\omega_{k,m}\left|\frac{\sqrt{\mu_m^2 + \eta_m^2}}{\mu_m}\right| + y_0 \tag{5.18}$$

In (x, y, z) and (x, y, t) the ID of each ray is given by two integer indices corresponding to *i* and *j* to yield a global ID given by *k*. The relationship between these indices and the track spacings is given in Eq. (5.19) (refer to section 2.1 and 3.2 for coordinate system definitions).

$$\begin{aligned} \omega &= i\Delta\omega_{k,m} \\ u &= j\Delta u_{k,m} \end{aligned} \tag{5.19}$$

Equation (5.20) defines how the global ID of a ray is found using these two indices, where *min* and *max* refers to the minimum and maximum track index value for this angle across the full problem domain.

$$
\begin{aligned}
k &= j - j_{\min} + \left(i - i_{\min}\right)\left(j_{\max} - j_{\min} + 1\right) \\
i &= i_{\min} + \left\lfloor \frac{k}{\left(j_{\max} - j_{\min} + 1\right)} \right\rfloor \\
j &= k + j_{\min} - \left(i - i_{\min}\right)\left(j_{\max} - j_{\min} + 1\right)
\end{aligned}
\tag{5.20}
$$

The equations for this LC method relating the (x, y, z) position to the (u, ω, s) position using these track indices are given in Eq. (5.21), where again the x_0, y_0, and z_0 parameters represent the offset of the two coordinate systems.

$$
\begin{aligned}
x &= x_0 + j\Delta u_{k,m} \frac{\mu_m}{\sqrt{1-\xi_m^2}} \xi_m - i\Delta\omega_{k,m} \frac{\eta_m}{\sqrt{1-\xi_m^2}} + s\mu_m \\
y &= y_0 + j\Delta u_{k,m} \frac{\eta_m}{\sqrt{1-\xi_m^2}} \xi_m + i\Delta\omega_{k,m} \frac{\mu_m}{\sqrt{1-\xi_m^2}} + s\eta_m \\
z &= z_0 - j\Delta u_{k,m} \sqrt{1-\xi_m^2} + s\xi_m
\end{aligned}
\tag{5.21}
$$

And similarly, the equations relating the (x, y, vt) position to the (u, ω, s) position using these indices are given in Eq. (5.22).

$$
\begin{aligned}
x &= x_0 - i\Delta\omega_{k,m} \frac{\eta_m}{\sqrt{\mu_m^2 + \eta_m^2}} + \mu_m s \\
y &= y_0 + i\Delta\omega_{k,m} \frac{\mu_m}{\sqrt{\mu_m^2 + \eta_m^2}} + \eta_m s \\
vt &= vt_0 + \frac{j\Delta u_{k,m}}{\sqrt{\mu_m^2 + \eta_m^2}} + s
\end{aligned}
\tag{5.22}
$$

The details of each function and class in PDT for the LC and STLC methods are detailed in Appendix B. Below the differences for LC methods as compared to other methods in PDT is briefly discussed.

5.2.1 Sweepchunks

The next big difference that occurs for a steady-state solution in PDT is the sweepchunk. A sweepchunk performs calculations for a given set of energy, angle, and spatial unknowns. Each spatial discretization method has a separate sweepchunk which handles the solving of the transport equation. The details of the LC and STLC sweepchunks are found in Appendix B. Ultimately, the sweepchunks solve for the angular flux exiting each cell along each track and contribute to the Mlc matrix (Eq. (5.7)) and Tlc vector (Eq. (5.8)) needed to find the scalar flux solution in each cell on a given groupset and angleset during the transport sweep process. The STLC sweepchunk also finds the exiting angular flux values for each track on a space-time cell and contributes to the matrices needed to find the scalar flux solution in each cell for an STLC method.

For the LC and STLC methods, finding the least-squares scalar flux via Gauss Elimination is done after a complete sweep of the cells has been performed. The simple details of this routine to find the least-squares scalar flux solution in each cell of the current groupset are given in Appendix B.

5.2.2 Time-Dependent Solve

Another difference from other methods in PDT for STLC methods occurs for a time-dependent problem. For the PWC-STLC and PPWL-STLC methods there are several changes needed in the time-dependent solving routine to account for the incoming tracks on the time boundary of each cell. These details can be found in Appendix B. The implementation of the PPWL-STLC method as applied to a radiative

transfer problem requires the call to a separate radiative transfer solver due to treating the temperature as linear and discontinuous with 2*N* unknowns per cell. The radiative transfer equations solved with this PPWL-STLC method in PDT can be found in Appendix B. It is noted that all of these time-dependent solvers eventually call the steady-state sweepchunks for each time step in order to find the scalar flux or absorption rate density for each cell in a time step.

5.2.3 STLC Iterative Operator

The STLC methods also require a separate iterative algorithm when solving for the absorption rate density (ARD), the flux within a groupset (WGS), or the flux across a groupset (AGS). These operators are very similar to the usual iterative operators to solve for the ARD, WGS flux, and AGS flux. The main difference with the usual operators is the need to use the values at the beginning and end of the time step in a cell for the STLC methods, such as phi_n_minus (ϕ^{bot}) and phi_n_plus (ϕ^{top}), instead of the average values in a cell.

5.2.4 Reporting Values

The flux values and associated rate densities that are reported for the LC and STLC methods are the least-squares fluxes. Therefore, the reaction rates that are reported or calculated with these fluxes will cause this method to not be cell-wise conservative in the sense of the actual cell volume. The user must be aware of this fact when interpreting the results of an LC or STLC problem. See the discussion of conservation given in section 4.1 for further details.

When volume-weighted average values (e.g. scalar flux) are reported in PDT, they are determined by using actual element volumes instead of an integral over basis functions. The expression used in PDT to calculate a cell-average flux is given in Eq. (5.23). In this expression *LS* refers to the least-squares LC flux, V_j refers to the volume of an element of a cell, and V_{cell} refers to the volume of a cell.

$$\phi_{avg} = \frac{\sum_{j}^{\#\text{elements}} \phi_j^{LS} V_j}{V_{cell}} \tag{5.23}$$

The true expression that should be used to calculate a cell-average flux is given in Eq. (5.24) where *b* refers to a basis function used approximate the collision source.

$$\phi_{avg} = \frac{\sum_{j}^{\#\text{elements}} \phi_j^{LS} \int d^3 r b_j(\vec{r})}{V_{cell}} \tag{5.24}$$

For an orthogonal grid these two expressions yield the same average flux value in a cell. For a non-orthogonal grid, the cell-average flux reported using Eq. (5.23) is not the true weighted average (which is given in Eq. (5.24)). Usually the difference between these cell-average fluxes is small, however the user should be aware of this difference when reporting reaction rates with the cell-average flux value reported from PDT.

5.2.5 Computational Considerations

This section addresses the number of unknowns that need to be computed with these LC and STLC methods. It also compares these unknowns to similar DFEM methods. First the number of unknowns required to form the collision source for each of these methods is presented in Table 1. In this table, *N* refers to the number of vertices of

a polygonal cell. Note that steady-state and time-dependent methods are both shown in this table.

Table 1. Number of Unknowns to Form the Collision Source in a Cell

Unknowns Per Cell	PWLD	PWL-LC	FI-PWLD	TBDF2-PWLD	PPWL-STLC
Collision Source Unknowns	N	N	N	2N	2N

The total number of angular fluxes calculated in a problem can also be compared between these methods. For the LC and STLC methods, this number depends on the number of angles, energy groups, grid geometry, and track spacing for a problem. A comparison of the total number of angular flux values in a problem for the LC and STLC methods with PWLD methods is given for two test problems in section 6.

6. RESULTS

This section presents numerical results from various test problems using the PWL-LC method and STLC method that are implemented in the Parallel Deterministic Transport, PDT, code at Texas A&M University. It is split into subsections involving steady-state *x-y* and *x-y-z* problems as well as time-dependent problems for these spatial domains. The final subsection of this section presents results from test problems using the STLC method in *x-y-vt* space, particularly the PWC-STLC method and the PPWL-STLC method. The main method for comparison with these methods is the PWL-DFEM in *x-y* space and *x-y-z* space with fully implicit or TBDF2 time differencing.

Before presenting these numerical results, some notation is defined for ease in discussing the convergence tolerance used for these test problems in PDT. First, the L_2 norm of a function can be defined as given in Eq. (6.1).

$$\| f \|_2 \equiv \mathrm{L}_2 \text{ norm of } f \tag{6.1}$$

Using this definition, the relative residual of the l^{th} iterate is defined in Eq. (6.2). This residual is based on solving the system $A\phi = b$.

$$r_{2,rel}^{(l)} \equiv \frac{\left\| b - A\phi^{(l)} \right\|_2}{\|b\|_2} \tag{6.2}$$

A definition for the relative pointwise difference between iterates is also defined as given in Eq. (6.3).

$$d_{2,rel}^{(l)} \equiv \max_j \left| \frac{\phi_j^{(l)} - \phi_j^{(l-1)}}{\phi_j^{(l)}} \right| \tag{6.3}$$

These definitions will be used in the following subsections.

6.1 Steady-State PWL-LC Test Problems in X-Y

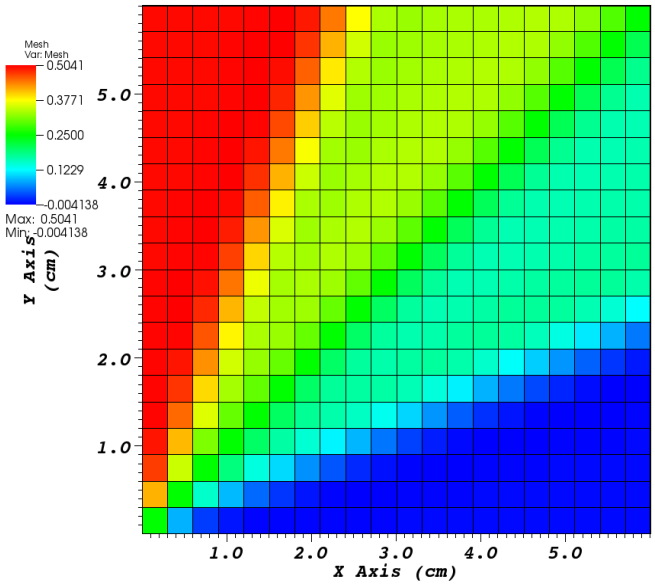
The test problems in this subsection demonstrate the properties of the PWL-LC method as applied to steady-state neutron transport problems in two dimensions, (x, y).

6.1.1 Streaming in a Vacuum

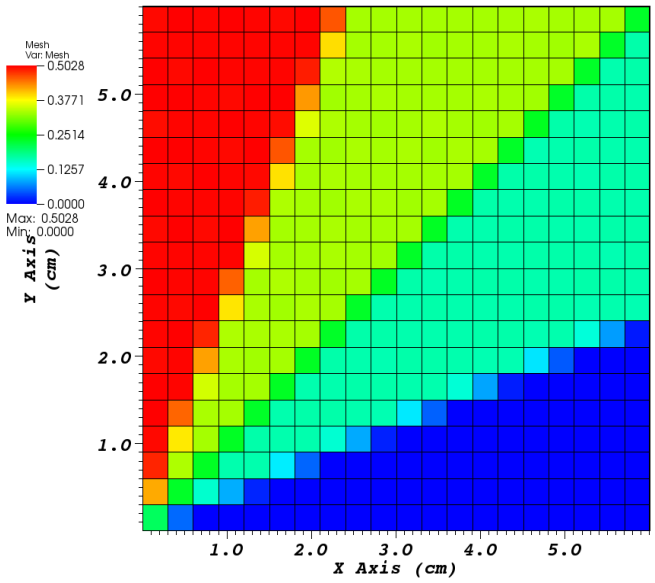
This first test problem is designed to show how the PWL-LC method compares to standard methods in a steady-state streaming problem in a vacuum. The test problem consists of a 20x20 rectangular grid of uniform width and height, $\Delta x = \Delta y = 0.3$ cm, and a fixed track spacing of $\Delta\omega = 0.05$ cm. An S_4 level-symmetric quadrature set is used in this problem with an incident beam on the left side of the problem defined in Eq. (6.4) and vacuum boundary conditions elsewhere. This beam is incident on essentially a vacuum with the total cross section of the material of σ_t = 1E-07 cm^{-1} with no scattering.

$$\psi_m^{inc}(0,y) = 2\frac{\text{n}}{\text{cm}^2\text{-s-str}}, \quad m:\mu_m > 0, \eta_m > 0 \tag{6.4}$$

Figure 13 shows the cell-average scalar flux solution using the PWL-LC method and the PWL-DFEM. As can be seen, these results demonstrate that the PWL-LC method does not spread the solution for this streaming problem, unlike the PWL-DFEM. Also note that this PWL-LC method does not find negative average scalar flux solutions in a cell for a streaming problem because this method finds the exact angular flux along each ray and uses these values to do the least-squares fit. Figure 13 also shows that the PWL-DFEM does generate some negative cell-averaged scalar fluxes for this streaming problem.



(a)



(b)

Figure 13. Cell-Average Scalar Flux for Incident Beam in a Vacuum (a) PWL-DFEM (b) PWL-LC

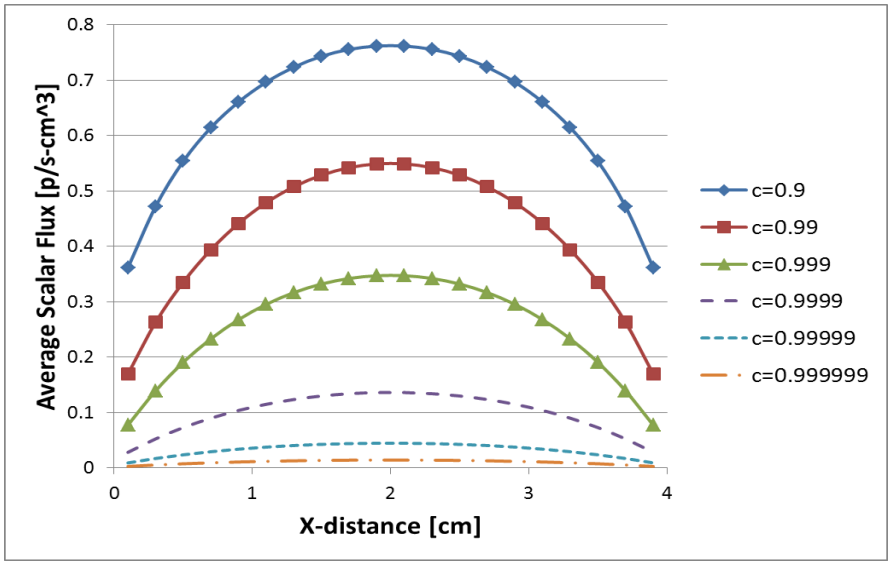
6.1.2 Diffusion Limit Symmetric Test Problem

This second test problem is designed to show the properties of the PWL-LC method in a highly diffusive problem. It consists of a multi-cell rectangular grid of uniform rectangular cells of size $\Delta x = \Delta y = 0.2$ cm and a fixed track spacing of $\Delta\omega =$ 0.01 cm. The overall width and height of the problem are $X = Y = 4$ cm. A family of transport problems is run by changing the value of ε to tend toward zero with the specifications given in Eq. (6.5). As $\varepsilon \to 0$, the problem becomes optically thick and highly scattering. Theory described in section 4 shows that the solution should approach a diffusion solution as $\varepsilon \to 0$, given the chosen specifications. GMRES is used to converge the iterations on the scattering source with no preconditioner (other than the transport sweeps)[43]. The convergence tolerance in GMRES is $r_{2,rel} < 1E\text{-}06$.

$$\begin{aligned} c &= \frac{\sigma_s}{\sigma_t} = 1-\varepsilon^2 \\ \sigma_t &= \frac{\sigma_{t0}}{\varepsilon} = \frac{0.3}{\varepsilon} cm^{-1} \\ q &= \varepsilon q_0 = 0.6\varepsilon \frac{n}{cm^3 - s} \\ \psi_m^{inc}(x,y) &= 0 \frac{n}{cm^2 - s - str} \quad m \in M \end{aligned} \tag{6.5}$$

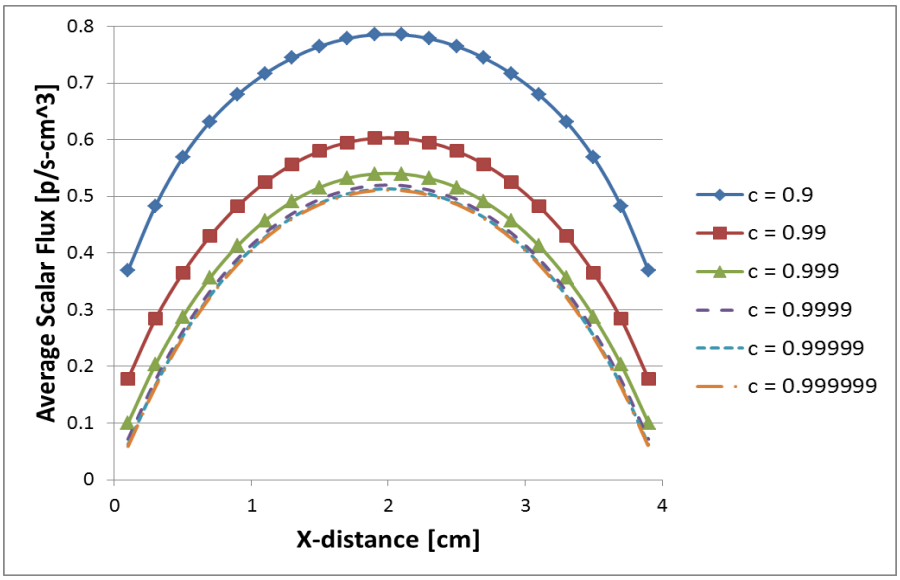
The correct transport solution for this problem approaches a diffusion solution as ε approaches zero[40,44,45]. Figure 14(a) shows a cross section of the cell-average scalar flux solution of this problem using the PWC-LC method at a position of $y = 2.1$ cm. It can be clearly seen that as $\varepsilon \to 0$, the PWC-LC solution is going toward zero, which is predicted by analysis[21]. Figure 14(b) shows a similar plot of the PWL-LC solution. As opposed to the PWC-LC solution, this PWL-LC solution approaches a fixed diffusion

solution as $\varepsilon \to 0$. The excellent asymptotic behavior observed by this method is expected based upon the analysis presented in section 4. The PWL basis functions used in this method are designed to extend this excellent behavior in the diffusion limit to polygonal and polyhedral spatial cells but no cell-wise polynomial bases can achieve this behavior on general polygons and polyhedral[23,35]. In addition to these PWL bases, other bases can achieve this good behavior including Wachspress's rational bases[33] and Warsa's piecewise polynomial bases[34].



(a)

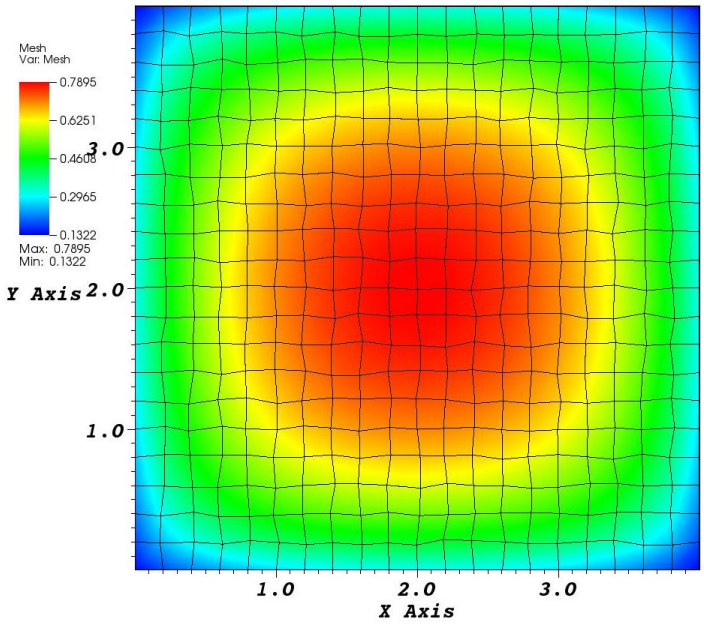
Figure 14. Cross Section of Cell-Average Scalar Flux Solution at Y = 2.1 cm (a) PWC-LC (b) PWL-LC



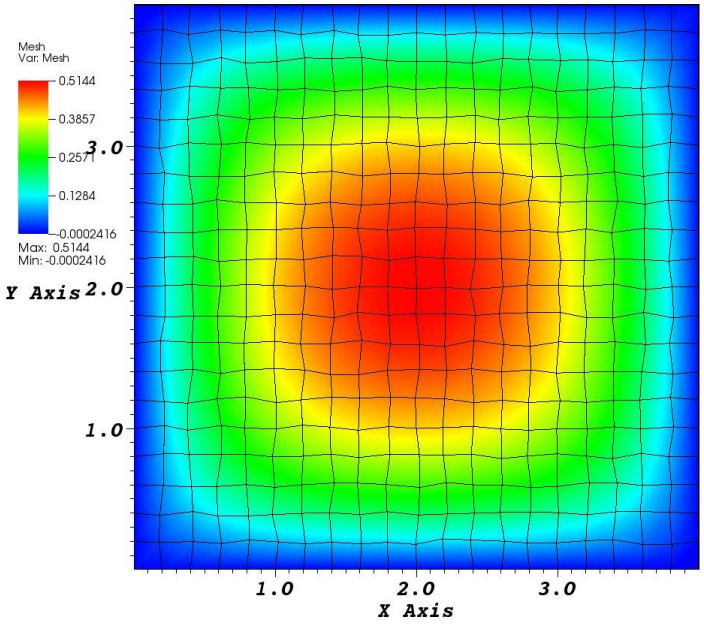
(b)

Figure 14. Continued

This family of problems was also run on a non-orthogonal grid that was produced in PDT by randomly moving the vertices of the orthogonal grid given above. Figure 15(a) and (b) show the pointwise scalar flux solution on this non-orthogonal grid using the PWL-LC method for the two test problems with scattering ratios of $c = 0.9$ and $c = 0.999999$, respectively. These figures show that the solution in the diffusion limit is almost identical to that on the rectangular grid. It is also the same result that is achieved with the PWL-DFEM method. All of this is predicted by the theory given in section 4.



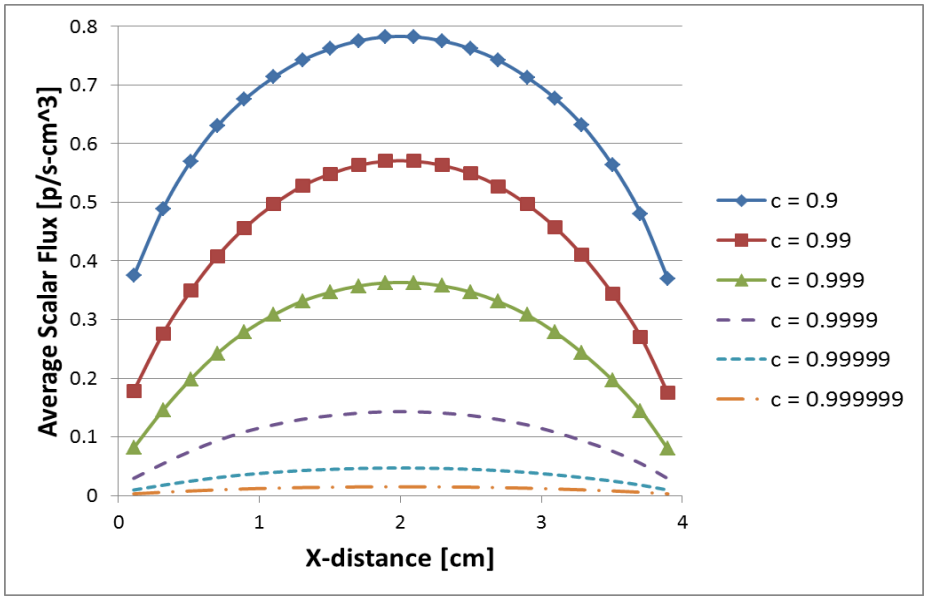
(a)



(b)

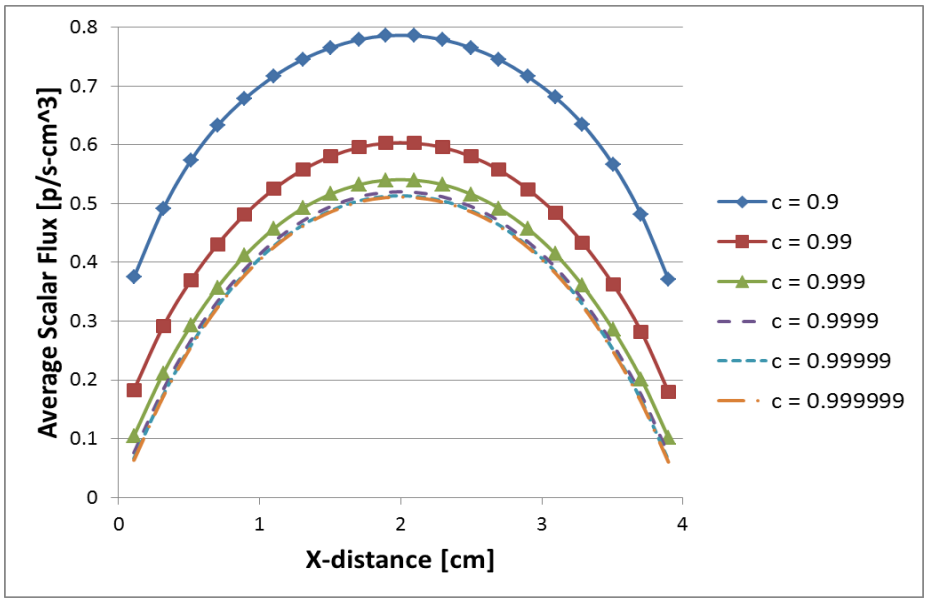
Figure 15. PWL-LC Pointwise Scalar Flux on Non-Orthogonal Grid (a) c = 0.9 (b) c = 0.999999

Figure 16 shows the cross section of the cell-average scalar flux solution using the PWC-LC and PWL-LC methods at a position of y = 2.1 cm on this non-orthogonal grid. The expected behavior for these methods is also observed on this non-orthogonal grid: the PWC-LC solution goes toward zero and the PWL-LC solution approaches a diffusion solution.



(a)

Figure 16. Cross Section of Cell-Average Scalar Flux Solution at Y ≅ 2.1 cm (non-orthogonal grid) (a) PWC-LC (b) PWL-LC



(b)

Figure 16. Continued

6.1.3 Absorbing Block – LC vs. DFEM

This third problem is designed to show a problem in which the PWL-LC method performs better than the PWL-DFEM. It consists of an optically thin material with a thick absorbing block in it. The dimensions of this problem are X = $3\sqrt{3}$ cm by Y = 3.0 cm made up of an 80x20 cell grid. The geometry and cross sections for this test problem are given in Figure 17. There is a zero fixed source in this problem with an incident isotropic flux of $\psi_m^{inc}(0,y) = 200\frac{n}{cm^2 - s - str}$ on the left boundary of the problem and vacuum boundaries on the other three boundaries. A Gauss-Legendre-Chebyshev quadrature set was used with a total of 24 polar angles and 128 azimuthal angles. The

track spacing for this PWL-LC problem is $\Delta\omega$ = 0.003 cm with an origin offset of y_0 = 1.5 cm so that the characteristic rays are symmetric for opposing angles in the problem.

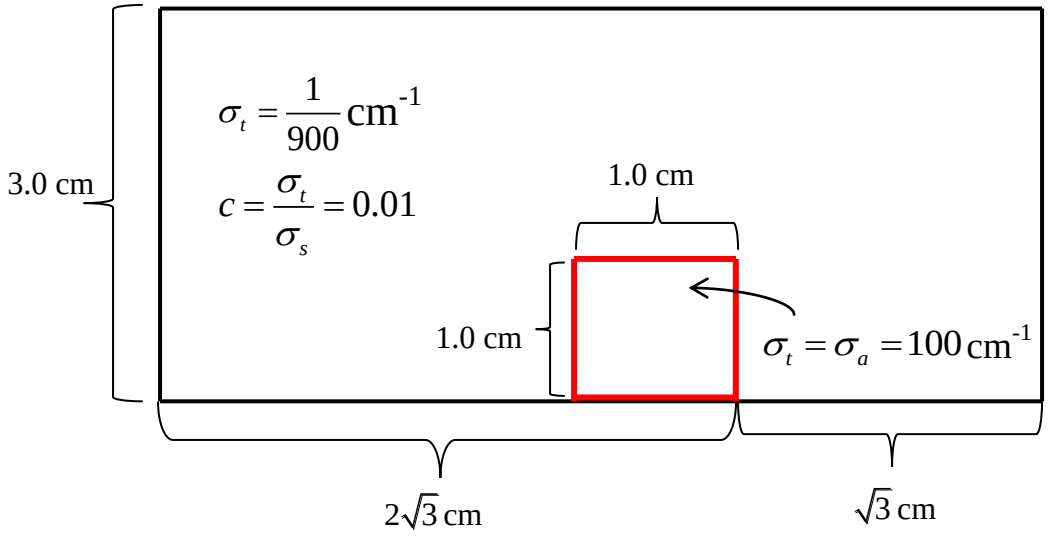
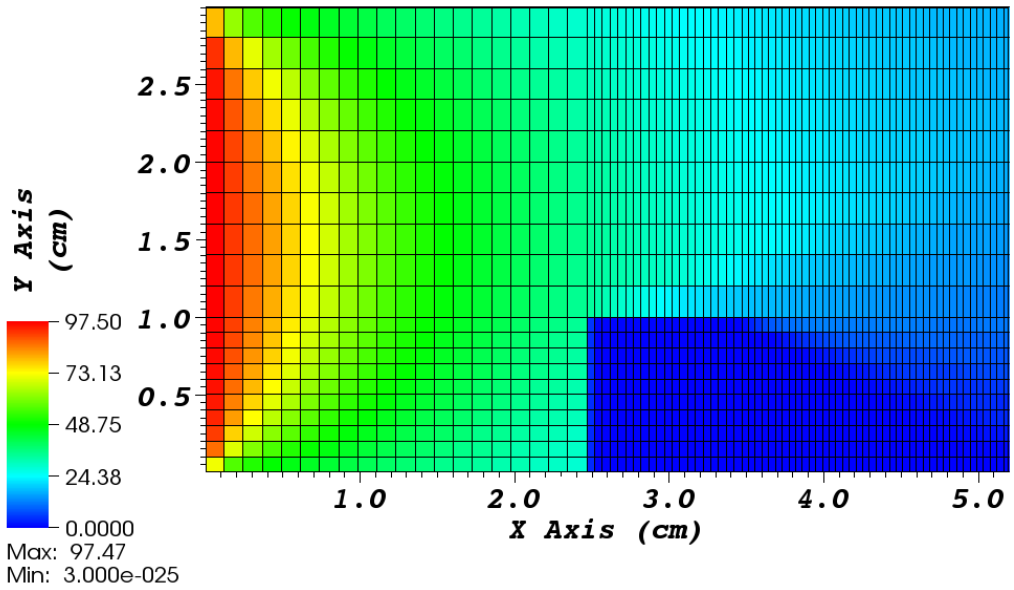
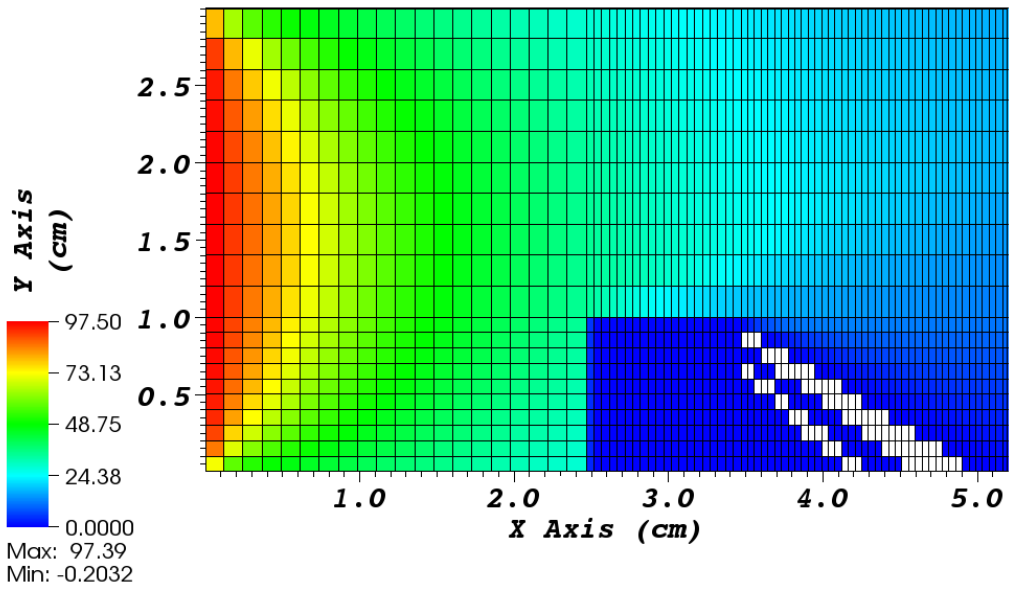


Figure 17. Geometry and Cross Section for Absorbing Block Problem

Figure 18 shows the PWL-LC cell-average scalar flux solution and the PWL-DFEM cell-average scalar flux solution plotted on the same scale for comparison. Cells in which the average scalar flux is negative are shown in white. From Figure 18(a), it can be seen that the PWL-LC method does not find any negative cell-average flux solutions for this problem, as expected because there is a very small amount of scattering. Figure 18(b) shows that the PWL-DFEM finds negative cell-average scalar flux values in the shadow region of the problem, which occurs behind the absorbing block. This oscillatory behavior comes from the PWL-DFEM projection of the angular flux, which is discontinuous in this problem, onto the PWL basis.



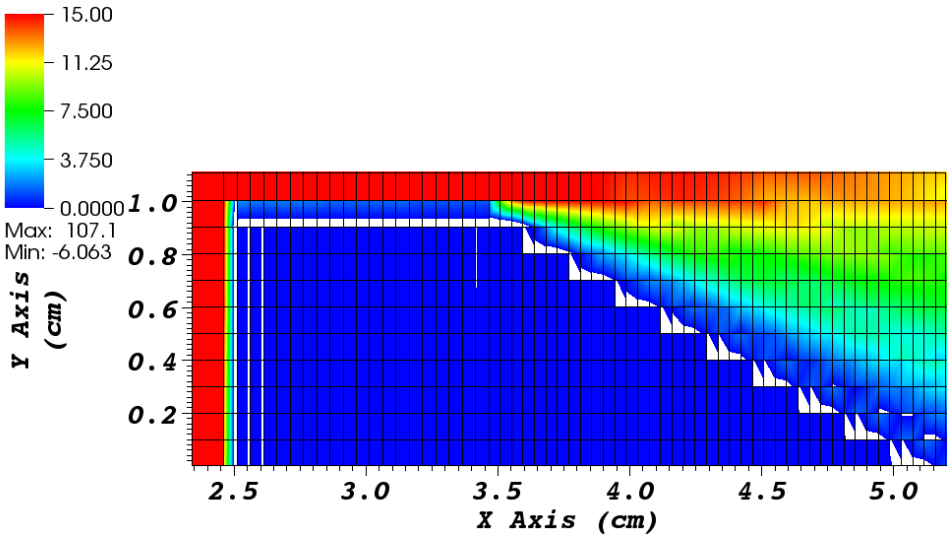
(a)



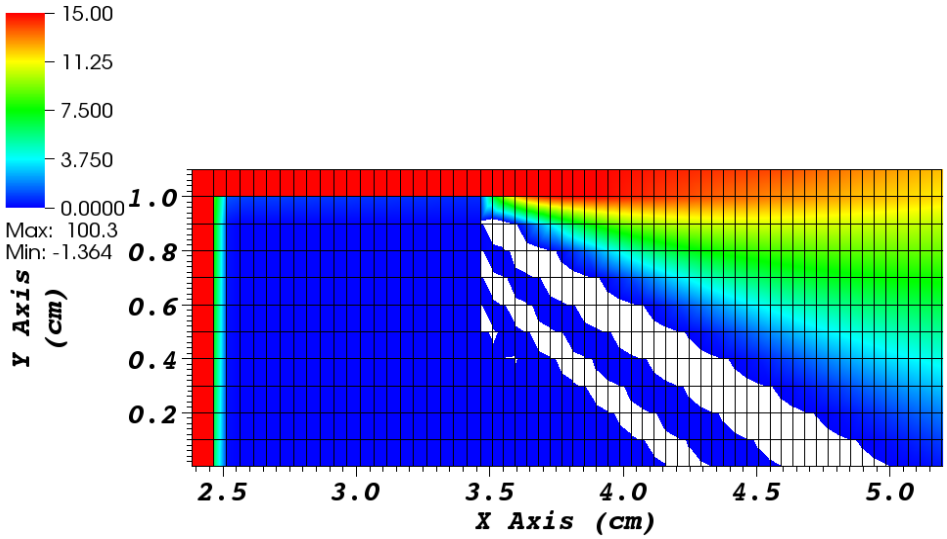
(b)

Figure 18. Cell-Average Scalar Flux Solution for Absorbing Block (a) PWL-LC (b) PWL-DFEM

Figure 19 shows the pointwise scalar flux solution, the flux plotted at the nodal values within each cell, using the PWL-LC method and the PWL-DFEM. These plots are again on the same scale and are zoomed in on the absorber block region and the shadowed region behind it. The color table used for these plots shows negative nodal values as white, therefore any other color is considered a positive flux value. The oscillatory behavior of the PWL-DFEM can clearly be seen in Figure 19(b). The negative values that are shown in Figure 19(a) for the PWL-LC method come from the least-squares projection of the solution onto the PWL basis. These negative values appear only in the cells for which there is a discontinuity and do not affect the neighboring cells because the angular flux along each track is found analytically using the assumed PWL source. Therefore, the PWL-LC method performs better on this type of problem than the PWL-DFEM.



(a)

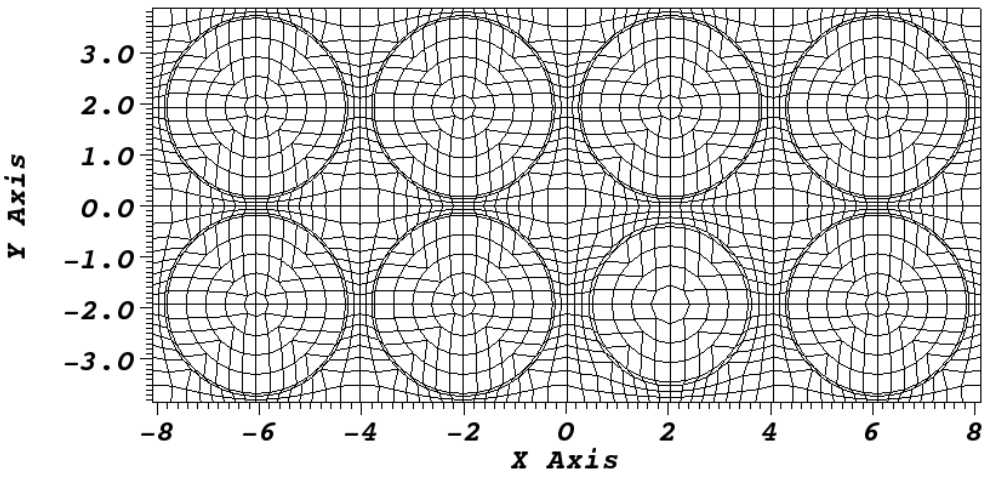


(b)

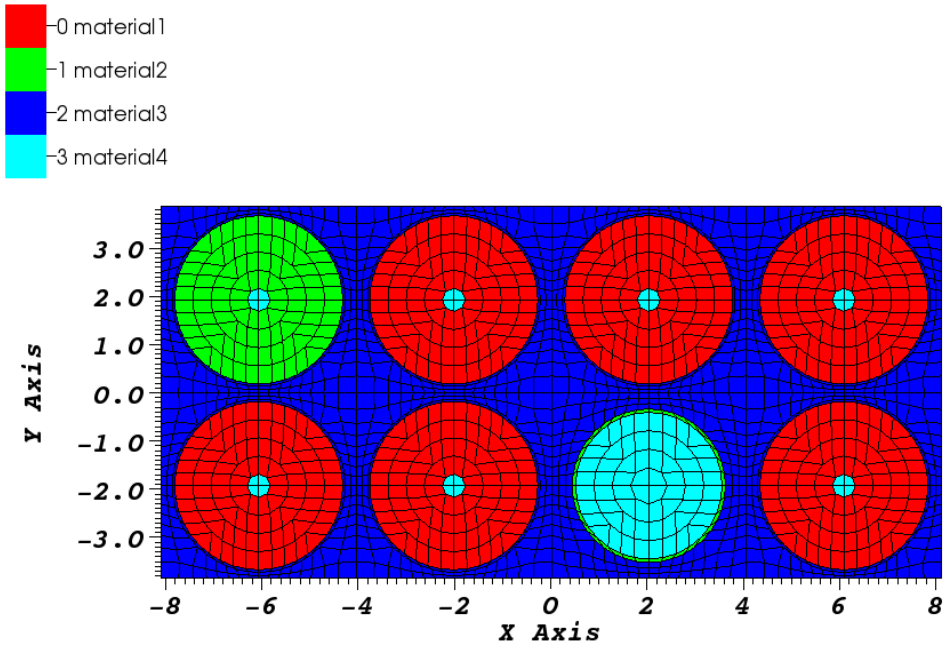
Figure 19. Pointwise Scalar Flux for Absorbing Block (a) PWL-LC (b) PWL-DFEM

6.1.4 TRIGA Test Pins

This test problem demonstrates the ability of the PWL-LC method to obtain an accurate solution on a non-orthogonal grid. It consists of a two-dimensional representation of some pins used in the TRIGA reactor at the Nuclear Science Center at Texas A&M University. The geometry and materials making up the pin layout is given in Figure 20. The pin layout is made up of two different instrumentation pins located in the top left position and third pin from the left on the bottom row with the rest of the pins representing fuel pins. The cells are all quadrilaterals formed by manipulating an orthogonal grid by moving the vertices. The *x* and *y* pitches for this pin layout are 4.05003 cm and 3.85445 cm, respectively. Two energy groups were used. Vacuum boundary conditions were used with a fixed source in the first energy group of the fuel region (material 1) of $q_{1,fixed} = 1.0$ n/cm^3-s with a zero fixed source elsewhere. The PWL-LC method and PWL-DFEM solutions were compared using this geometry and setup for a pure absorber problem and a scattering problem. Both methods used an S_2 level-symmetric quadrature set and PWL-LC used a track spacing of $\Delta\omega = 0.025$ cm and an offset of $x_0 = -8.10006$ cm.



(a)



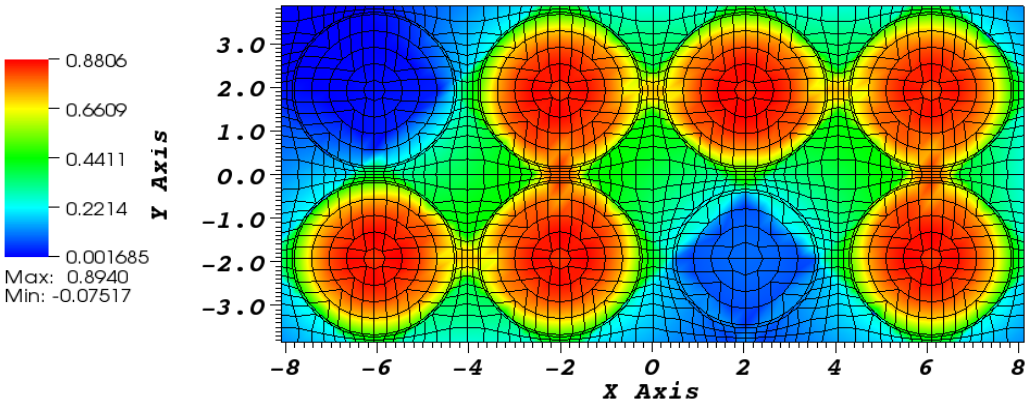
(b)

Figure 20. Mesh and Materials of TRIGA pins (a) Mesh (b) Material Layout

The first test problem using this TRIGA pin layout is a pure absorber with both groups having the same cross section in each material as given in Eq. (6.6). Richardson iteration was used to solve this problem in PDT.

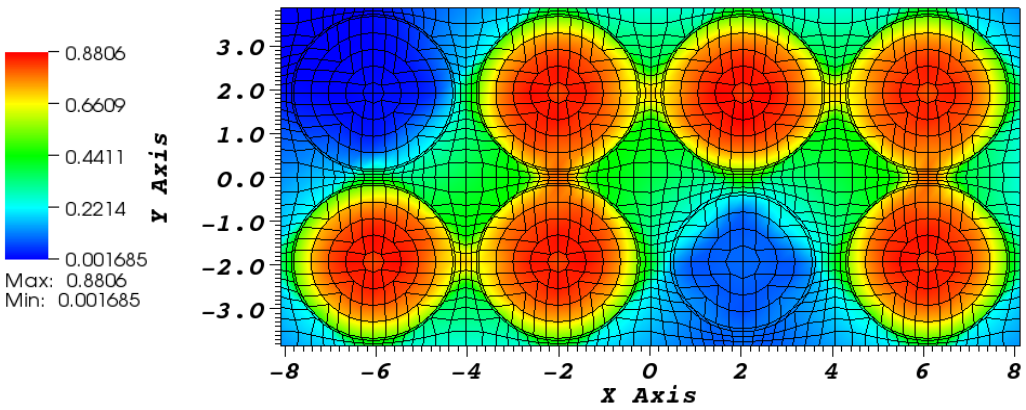
$$\begin{aligned}
\sigma_{t,\text{material1}} &= \sigma_{a,\text{material1}} = 1.0 \text{ cm}^{-1} \\
\sigma_{t,\text{material2}} &= \sigma_{a,\text{material2}} = 0.75 \text{ cm}^{-1} \\
\sigma_{t,\text{material3}} &= \sigma_{a,\text{material3}} = 0.5 \text{ cm}^{-1} \\
\sigma_{t,\text{material4}} &= \sigma_{a,\text{material4}} = 0.1 \text{ cm}^{-1}
\end{aligned} \tag{6.6}$$

Since there is no source in the second energy group and no scattering, the solution for the second energy group is zero and both methods find this solution. Figure 21 shows the pointwise scalar flux solution using the PWL-LC method and PWL-DFEM for the first energy group. Both of these plots are on the same scale, and it is noticeable that the PWL-LC method does not artificially spread the S_2 solution like the PWL-DFEM solution does.



(a)

Figure 21. Pointwise Scalar Flux Solution for Group 1 in Pure Absorbing TRIGA pins (a) PWL-LC (b) PWL-DFEM



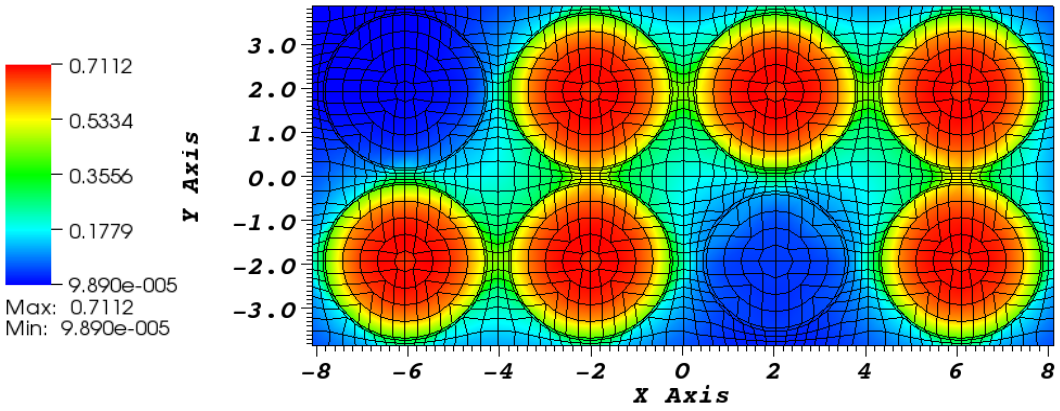
(b)

Figure 21. Continued

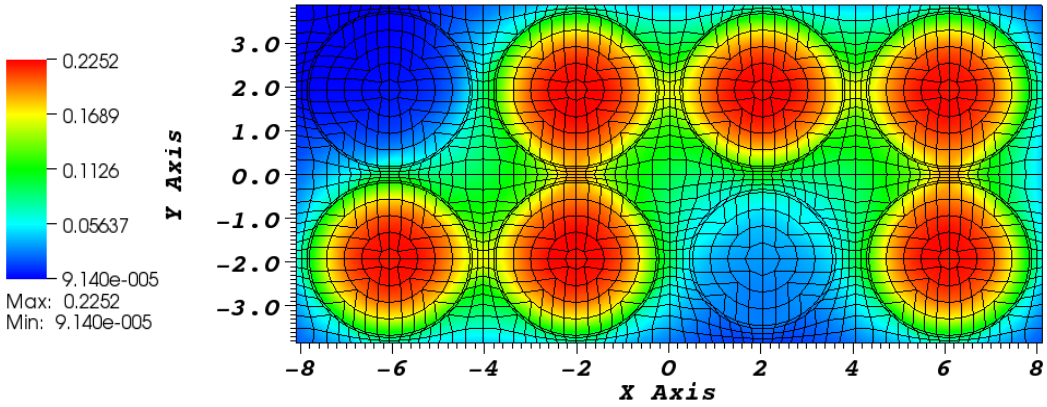
Next, this pin layout is run with isotropic scattering with both energy groups having the same cross sections with a scattering ratio of $c = \sigma_s / \sigma_t = 0.25$. Equation (6.7) gives the cross sections for each material in this scattering problem. Richardson iteration was used to converge the scattering source to a tolerance of $r_{2,rel} < 1E\text{-}05$.

$$\begin{aligned}
&\text{material 1:} \quad \sigma_t^1 = \sigma_t^2 = 2.0 \text{ cm}^{-1}, \quad \sigma_s^{1\to 2} = 0.5 \text{ cm}^{-1}, \quad \sigma_s^{2\to 1} = 0.5 \text{ cm}^{-1} \\
&\text{material 2:} \quad \sigma_t^1 = \sigma_t^2 = 1.5 \text{ cm}^{-1}, \quad \sigma_s^{1\to 2} = 0.375 \text{ cm}^{-1}, \quad \sigma_s^{2\to 1} = 0.375 \text{ cm}^{-1} \\
&\text{material 3:} \quad \sigma_t^1 = \sigma_t^2 = 1.0 \text{ cm}^{-1}, \quad \sigma_s^{1\to 2} = 0.25 \text{ cm}^{-1}, \quad \sigma_s^{2\to 1} = 0.25 \text{ cm}^{-1} \\
&\text{material 4:} \quad \sigma_t^1 = \sigma_t^2 = 0.2 \text{ cm}^{-1}, \quad \sigma_s^{1\to 2} = 0.05 \text{ cm}^{-1}, \quad \sigma_s^{2\to 1} = 0.05 \text{ cm}^{-1}
\end{aligned} \tag{6.7}$$

Figure 22 shows the pointwise scalar flux solution using the PWL-DFEM for both energy groups. As expected the solution for the first energy group is larger than for the second energy group since it contains the fixed source.



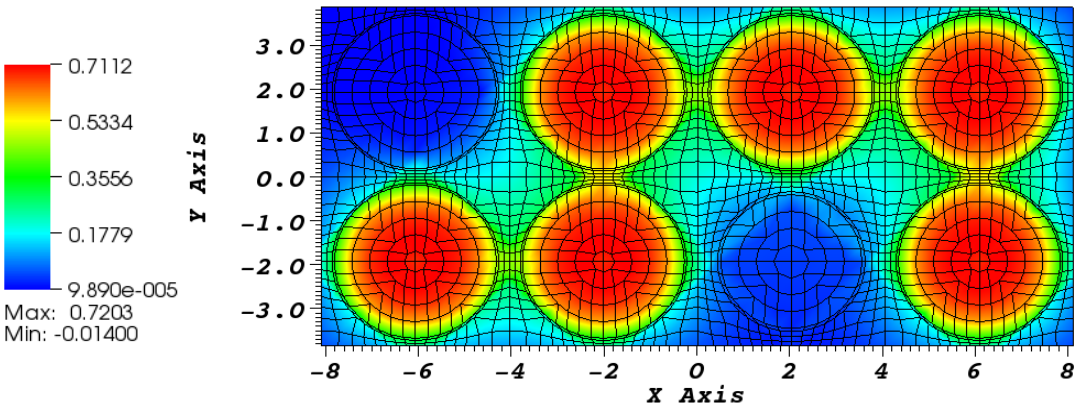
(a)



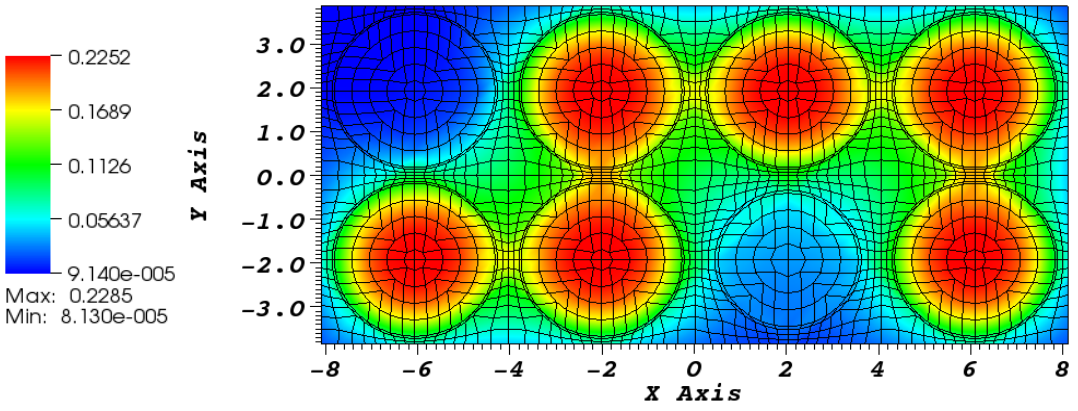
(b)

Figure 22. Pointwise Scalar Flux Solution for PWL-DFEM in Scattering TRIGA pins (a) Group 1 (b) Group 2

These results are very similar to the solution from the PWL-LC method given in Figure 23. The relative difference between these solutions is no more than 30%. This difference could be mitigated by using a smaller track spacing for the PWL-LC method. Overall, this test problem shows that the PWL-LC method can achieve an accurate solution for a scattering problem on a non-orthogonal grid.



(a)



(b)

Figure 23. Pointwise Scalar Flux Solution for PWL-LC in Scattering TRIGA pins

(a) Group 1 (b) Group 2

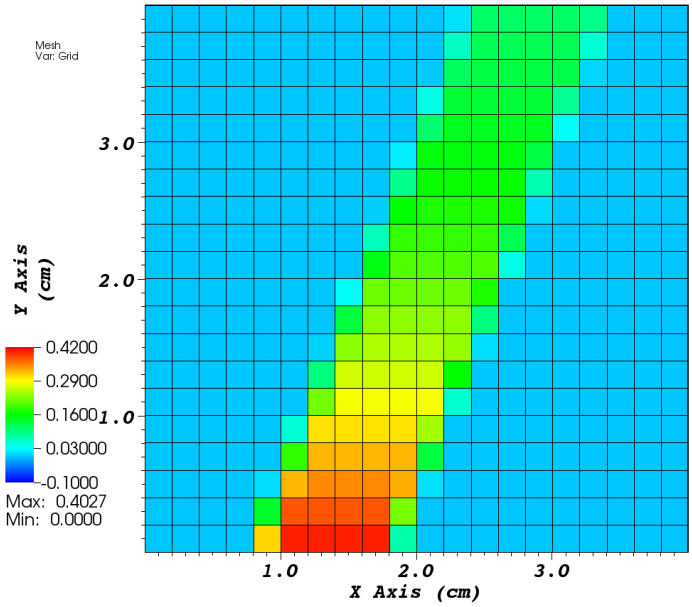
6.1.5 Streaming in a Pure Absorber

This fifth test problem is designed to show the properties of the PWL-LC method for a problem with streaming in a pure absorber. The steady-state results presented here

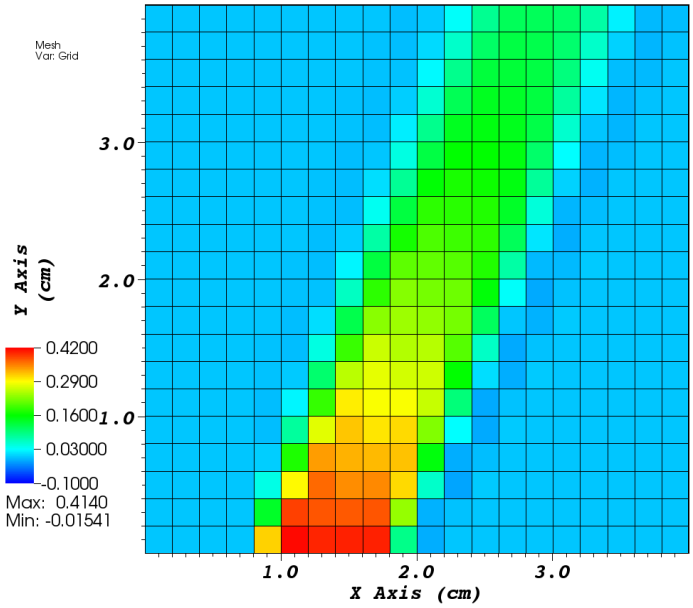
will be used for comparison with a time-dependent version of this test problem later in this section. This test problems consists of a 20x20 rectangular grid of uniform cell size, $\Delta x = \Delta y = 0.2$ cm, and a fixed track spacing of $\Delta\omega = 0.01$ cm. The total problem size for this test problem is $X = Y = 4$ cm. An S_4 level-symmetric quadrature set is used to solve this problem with the boundary condition given in Eq. (6.8) and vacuum boundaries elsewhere. The total cross section is $\sigma_t = \sigma_a = 0.3$ cm^{-1} with a zero fixed source.

$$\psi_m^{inc}(x,0) = 5\frac{n}{cm^2 - s - str}, \quad 0.8cm \le x \le 1.8cm, \quad m = 2, t > 0 \tag{6.8}$$

Figure 24 shows the steady-state cell-average scalar flux solution using the PWL-LC method and the PWL-DFEM. It is noted that since there is no scattering in this problem, the PWC-LC and PWL-LC solutions are the same for this steady-state problem. The plots given in Figure 24 are on the same scale. It is evident from these solutions that the PWL-LC method does not spread the solution and does not find negative solution values, in contrast with the PWL-DFEM.



(a)



(b)

Figure 24. Steady-State Cell-Average Scalar Fluxes for Beam in Pure Absorber (a) PWL-LC (b) PWL-DFEM

6.1.6 Linear Manufactured Solution

This final test problem shows how the 2D PWL-LC method finds the exact solution for a problem with an angular flux solution that is linear in x, y, η, and μ. The method of manufactured solutions is used to set up the proper boundary conditions and fixed source in order to produce this linear solution. The manufactured angular flux solution for the m^{th} direction is given in Eq. (6.9), where ψ_0, ψ_x, and ψ_y are constants.

$$\Psi_{mg}(x,y) = \psi_0 + \left(x - \frac{\mu_m}{\sigma_{t,g}}\right)\psi_x + \left(y - \frac{\eta_m}{\sigma_{t,g}}\right)\psi_y \tag{6.9}$$

Using this angular flux solution, the scalar flux solution can be found as given in Eq. (6.10), where a symmetric quadrature set is used whose weights sum to 4π.

$$\phi_g(x,y) = \sum_{m=1}^{M} w_m \Psi_{mg}(x,y) = 4\pi\psi_0 + 4\pi x\psi_x + 4\pi y\psi_y \tag{6.10}$$

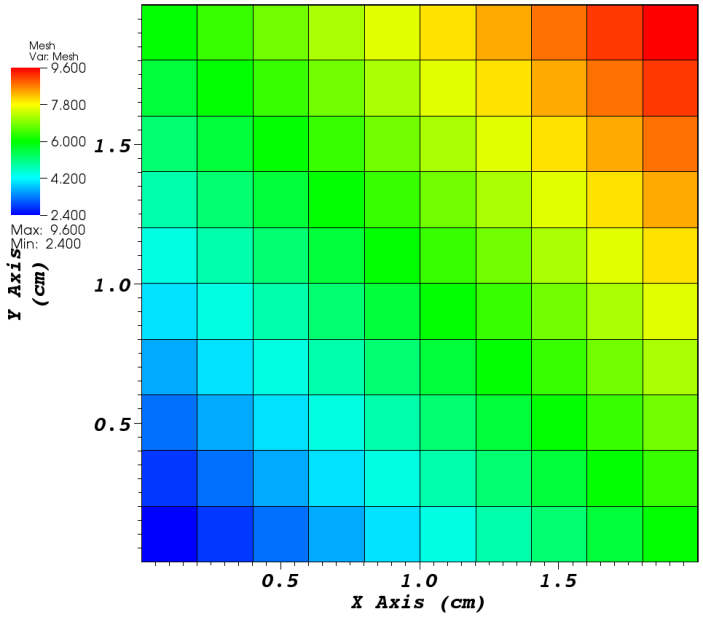
From plugging the manufactured solution into the steady-state transport equation, the fixed source that produces this solution is found to be the following:

$$q_{mg,\,fixed}(x,y) = \sigma_{a,g}\psi_0 + \sigma_{a,g}x\psi_x + \sigma_{a,g}y\psi_y, \quad \text{all } m. \tag{6.11}$$

The layout of this test problem consists of a uniform 10x10 rectangular grid with a cell size of $\Delta x = \Delta y = 0.2$ cm. The total cross section for this one-group problem is $\sigma_t = 1.0\ \text{cm}^{-1}$ with a scattering ratio of $c = 0.5$. A track spacing of $\Delta\omega = 0.02$ cm is used for this problem with an S_2 level symmetric quadrature set. The boundary condition for each track is given by Eq. (6.9) with a fixed source in each cell given by Eq. (6.11) using the following coefficients.

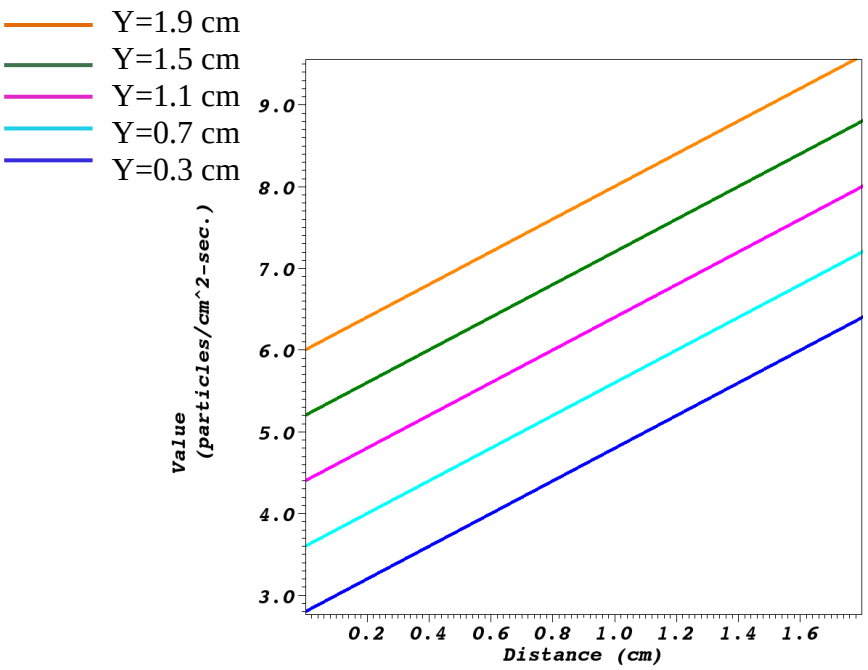
$$\psi_0 = 2.0 \quad \frac{n}{cm^2 - s - str}$$
$$\psi_x = \psi_y = 2.0 \quad \frac{n}{cm^3 - s - str} \tag{6.12}$$

Figure 25 shows the cell-average steady-state scalar flux solution using the PWL-LC method for the full problem (a) and a cross section of the solution at various *y* positions (b). As can be seen from this figure, this method produces the exact cell-averaged scalar flux solution of Eq. (6.10), as expected for this method.



(a)

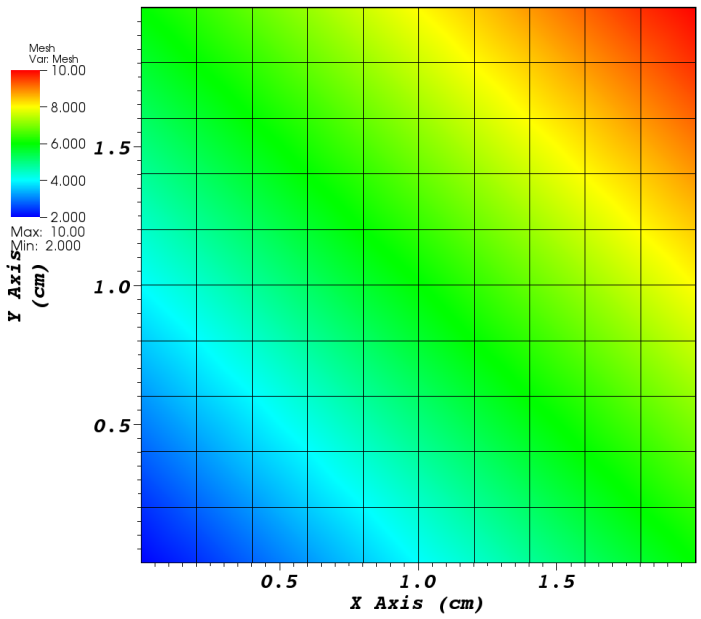
Figure 25. Steady-State Cell-Average Linear PWL-LC Scalar Flux Solution (a) Full (b) Line-Out



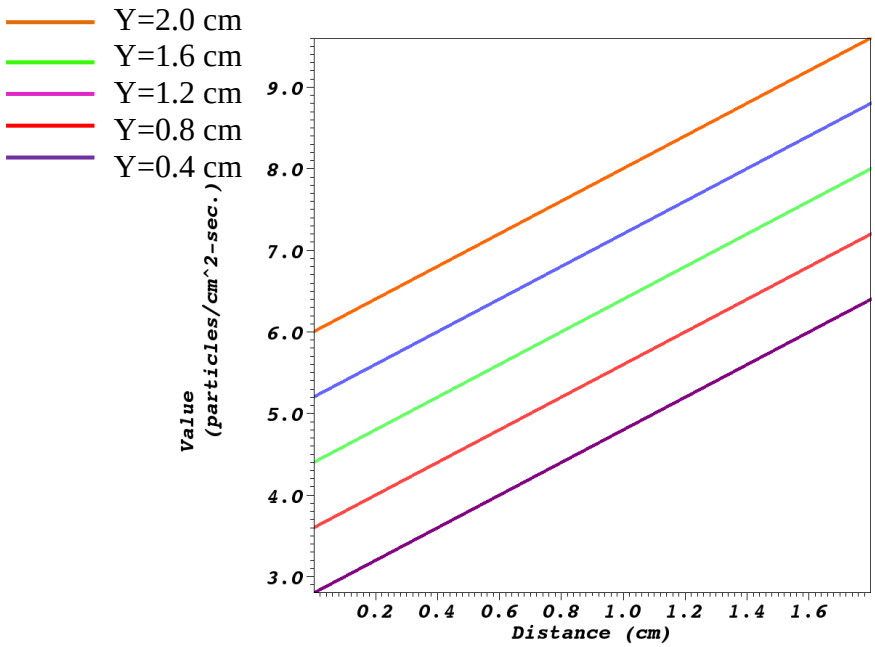
(b)

Figure 25. Continued

Figure 26 shows the pointwise steady-state scalar flux solution produced using the PWL-LC method for the full problem (a) and again at various *y* positions (b). As can be seen, this method also produces the exact linear scalar flux pointwise. Overall, this test problem has shown that since this PWL-LC method assumes a PWL scattering source representation, it can reproduce an exact linear solution for a problem in which the solution is linear in space and angle.



(a)



(b)

Figure 26. Steady-State Pointwise Linear PWL-LC Scalar Flux Solution (a) Full (b) Line-Out

6.2 Steady-State PWL-LC Test Problems in X-Y-Z

The test problem presented in this subsection shows the applicability of the PWL-LC method for steady-state transport problems in three dimensions. Since the properties of this steady-state PWL-LC method have been adequately shown through the various test problems in (x, y), this test problem is included to show that the PWL-LC method can find an accurate solution in (x, y, z) problems.

6.2.1 Reactor Pin Cells

This test problem consists of a 2x3 grid of pin cells in (x, y, z). An S_4 level-symmetric quadrature set is used along with a 7-group energy structure given in Table 2 to solve this problem. The 7-group macroscopic cross sections for the UO_2 pins and moderator (water) are taken from the C5G7 benchmark problem[46]. The control rod consists of 100% B-10. The B-10 7-group microscopic cross sections were produced using NJOY and multiplied by the number density of B-10 of $N_{B10} = 1.504E10^{23}$ atoms/cm^3 to convert to macroscopic cross sections. These cross sections for B-10 are shown in Table 3 and Table 4.

Table 2. 7-Group Energy Structure for Reactor Pin Cells

Energy Group	Lower Energy (eV)	Upper Energy (eV)
Group 1	1.35E06	1.0E07
Group 2	2.35E05	1.35E06
Group 3	1.30E02	2.35E05
Group 4	1.86E00	1.30E02
Group 5	6.0E-01	1.86E00
Group 6	1.0E-01	6.0E-01
Group 7	3.6E-03	1.0E-01

Table 3. B-10 Macroscopic Cross-Sections

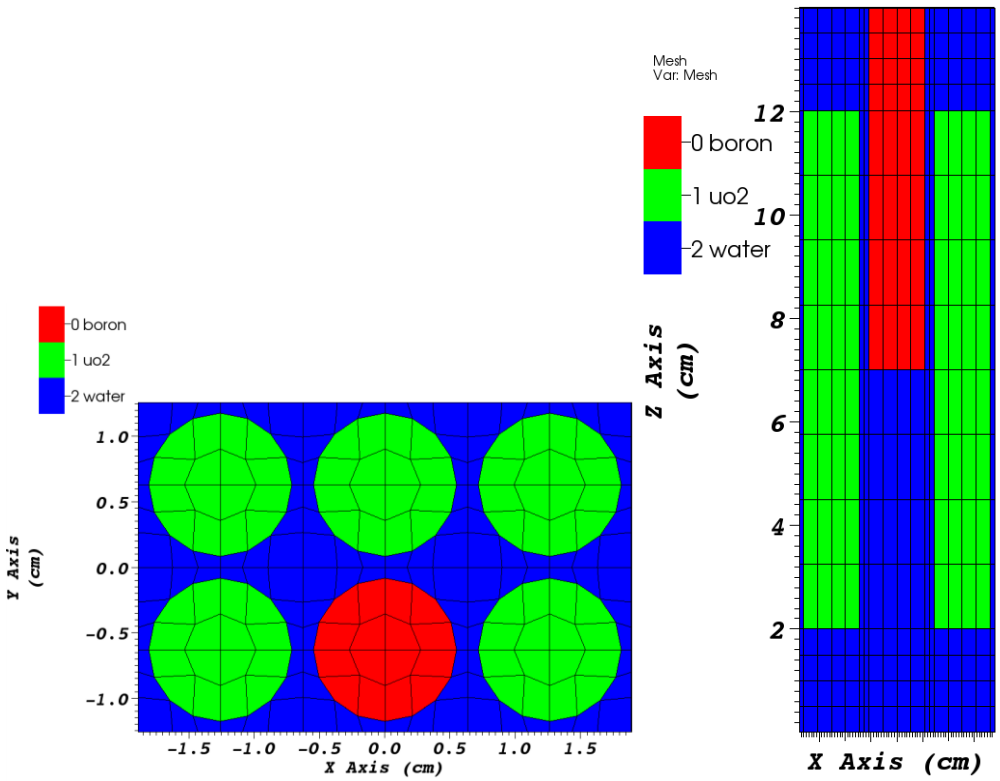
Energy Group	Total Cross-section (cm^{-1})
Group 1	2.99785E-01
Group 2	5.53041E-01
Group 3	2.06940E00
Group 4	27.06427E00
Group 5	90.91386E00
Group 6	212.3357E00
Group 7	440.75308E00

Table 4. B-10 Scattering Matrix (macroscopic cross-sections (cm^{-1}))

Energy Group	to Group 1	to Group 2	to Group 3	to Group 4	to Group 5	to Group 6	to Group 7
Group 1	1.9294E-01	4.4091E-02	9.3551E-05	0	0	0	0
Group 2	0	4.1908E-01	4.6829E-02	0	0	0	0
Group 3	0	0	3.6125E-01	5.8624E-03	0	0	0
Group 4	0	0	0	3.1272E-01	1.2807E-03	0	0
Group 5	0	0	0	0	2.7358E-01	5.3889E-02	0
Group 6	0	0	0	0	0	2.7417E-01	5.5284E-02
Group 7	0	0	0	0	0	0	3.3835E-01

The geometric and material layout of the pins is shown in Figure 27, which shows a cross section of the cells in (x, y) at z = 10 cm and a cross section of cells in (x, z) at y = -0.6 cm. The Boron control rod is moderator-followed and is inserted from z = 7.0 cm to 14.0 cm. This non-orthogonal grid is produced by moving the vertices of each cell with all resulting cells being polygonal prisms. The vertices of this 18x12x16 grid

are moved only in *x* and *y* to yield prismatic cells. Note that this mesh lines up with material boundaries. Vacuum boundary conditions are used on all problem boundaries with a spatially constant fixed source using a fission spectrum in energy in the UO_2 fuel regions (using the cross sections from the C5G7 benchmark problem[41]). The pitch in *x* and *y* for these pins is again taken from the C5G7 benchmark problem to be 1.26 cm. Track spacings for the PWL-LC method are set to $\Delta\omega = \Delta u = 0.05$ cm with a zero offset for all axes.



(a) **(b)**

Figure 27. Reactor Pin Cells Layout and Materials (a) z = 10 cm (b) y = -0.6 cm

The 7 groups in this problem were split into 3 energy groupsets (EGS) when solving this problem:

$$\text{EGS } 1 \in \{\text{Group 1, Group 2, Group 3}\},$$

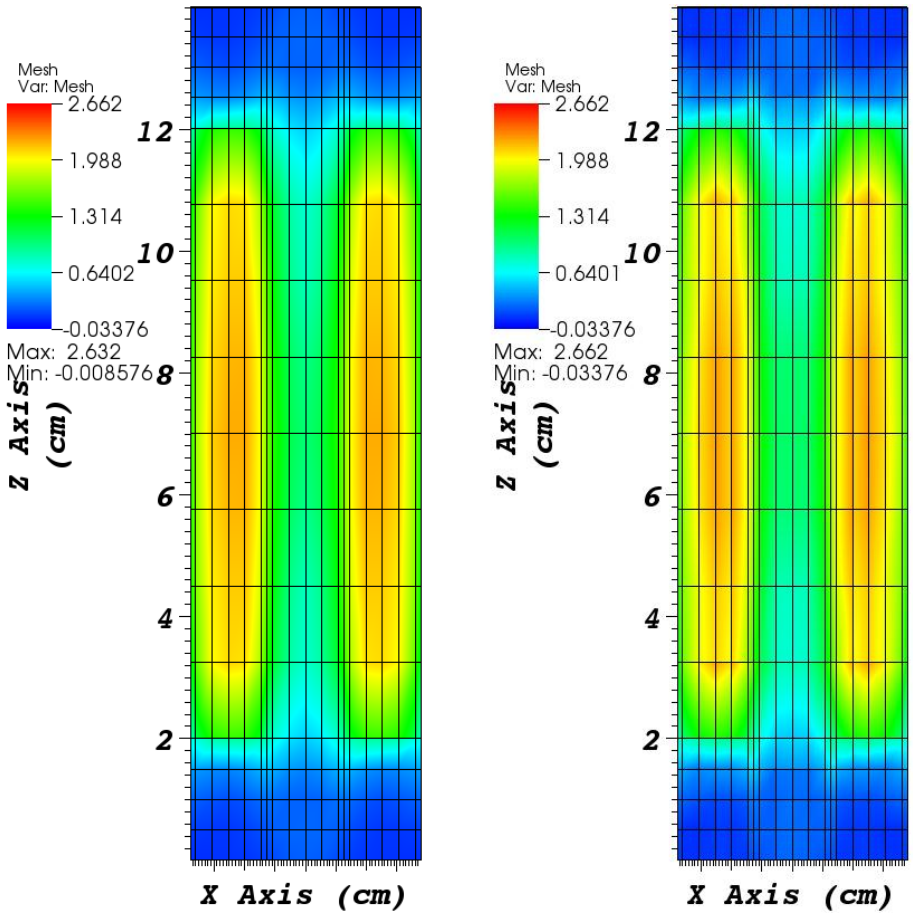
$$\text{EGS } 2 \in \{\text{Group 4, Group 5}\},$$

$$\text{EGS } 3 \in \{\text{Group 6, Group 7}\}.$$

GMRES was used to converge both of these problems with a convergence tolerance of $r_{2,rel} < 1E\text{-}07$ within each groupset. The solution in each groupset was also converged pointwise to a tolerance of $d_{2,rel} < 1E\text{-}03$. Across groupsets, the solution was converged using GMRES with the same residual tolerance, $r_{2,rel} = 1E\text{-}07$, and pointwise across groupsets with a tolerance of $d_{2,rel} < 1E\text{-}03$.

Two variations of this problem were run: one with the moderator-followed control rod inserted halfway axially and one with no control rod present (water hole). First, the scalar flux solution with no control rod present is shown using the PWL-DFEM and PWL-LC methods. There are a variety of slices of the solution that could be shown for all energy groups for comparison. Figure 28 shows a cross section of the pointwise scalar flux solution for group 1 at y = -0.6 cm, which includes the water-hole region. The PWLD and PWL-LC solutions are plotted on the same scale for ease of comparison. As expected, the flux solution for this energy group is higher in the fuel regions and lower in the water hole for both methods. From this figure it is also evident that the PWL-LC solution is very similar to the PWLD solution for this energy group, which is expected since both methods use the same collision source representation. Note that the PWL-LC method does not spread the solution as much as the PWLD method for this

highest energy group, which is also expected for this problem. The slight non-smoothness that occurs in the PWL-LC solution comes from ray effects caused by the S_4 level-symmetric quadrature set.



(a) **(b)**

Figure 28. Pointwise Group 1 Scalar Flux Solution w/o Control Rod, at y = -0.6 cm

(a) PWLD (b) PWL-LC

Figure 29 shows the solution from the PWLD and PWL-LC methods at the same *y* position for the highest energy group using an S_8 level-symmetric quadrature set. Note that the PWL-LC solution for this energy group is much smoother with this quadrature set and does not have negative flux values.

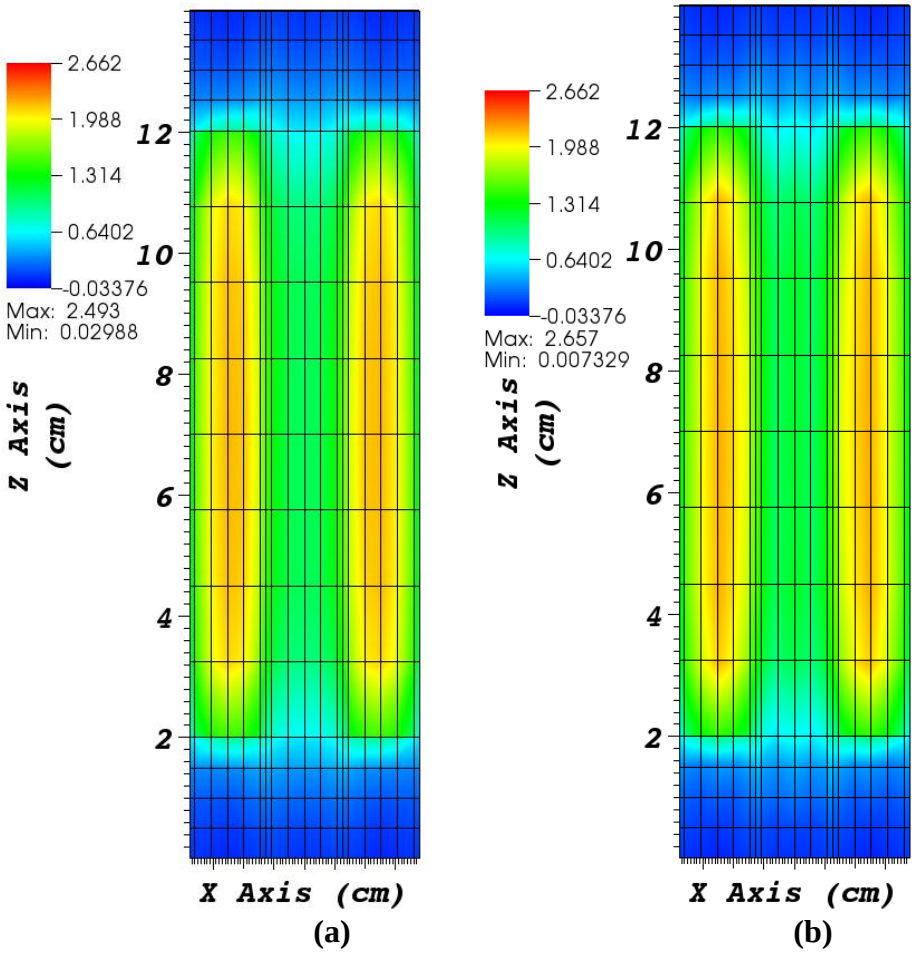
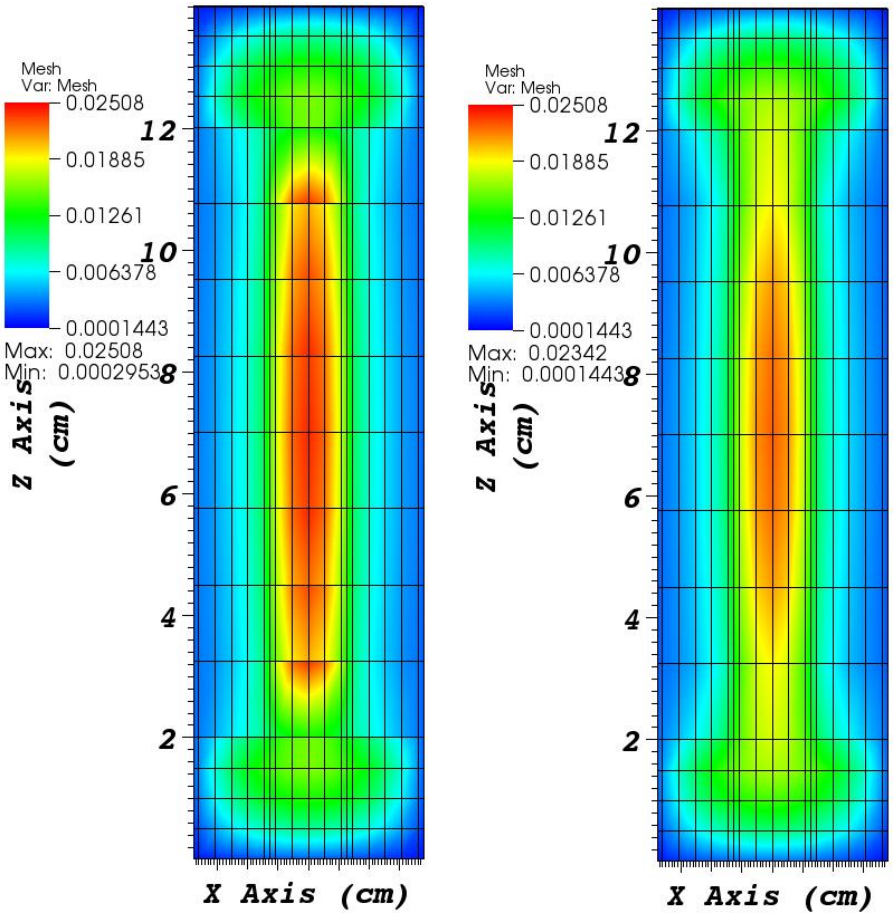


Figure 29. S_8 Pointwise Group 1 Scalar Flux Solution w/o Control Rod, at y = -0.6 cm (a) PWLD (b) PWL-LC

Figure 30 shows a cross section of the pointwise scalar flux solution for group 7 at y = -0.6 cm using an S_4 level-symmetric quadrature set. As expected, the solution in this lowest energy group peaks in the water hole. The PWL-LC method produces a much smoother scalar flux solution than the PWLD method, whose solution shows oscillations in the water hole. Thus, on this coarse axial mesh, the PWL-LC method is more accurate than the PWLD method.

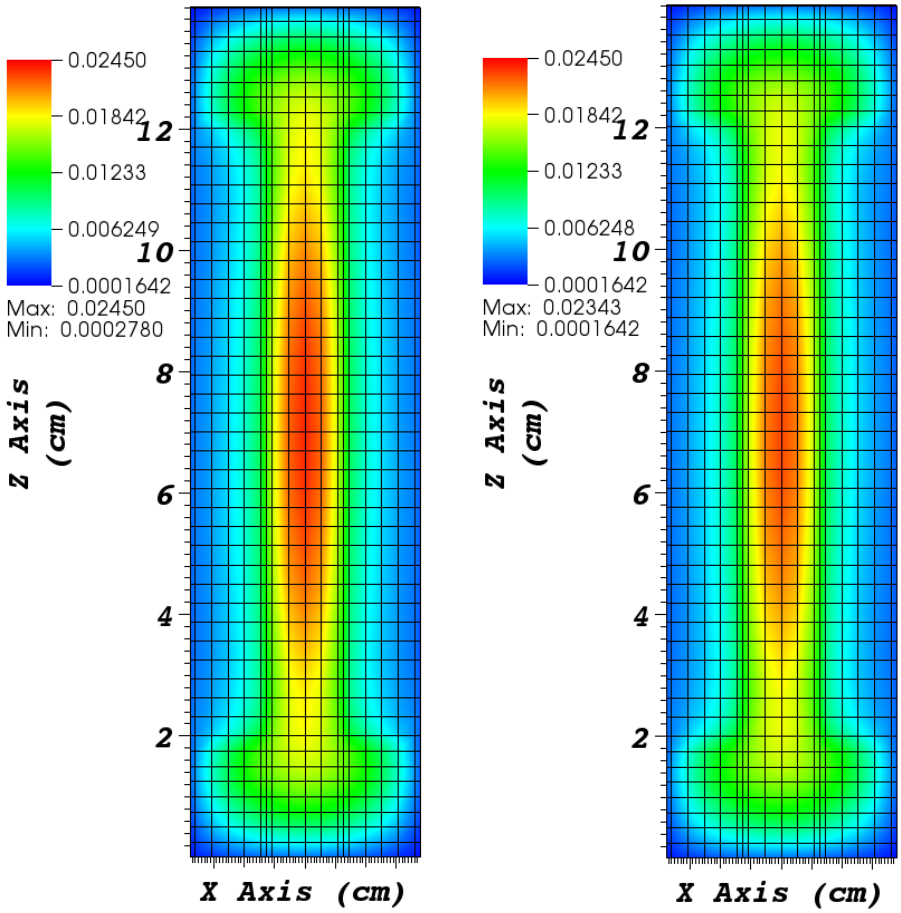


(a) **(b)**

Figure 30. Pointwise Group 7 Scalar Flux Solution w/o Control Rod, at y = -0.6 cm

(a) PWLD (b) PWL-LC

Figure 31 shows the PWLD and PWL-LC scalar flux solutions for group 7 at the same y = -0.6 cm on a grid that has been refined in the axial dimension.



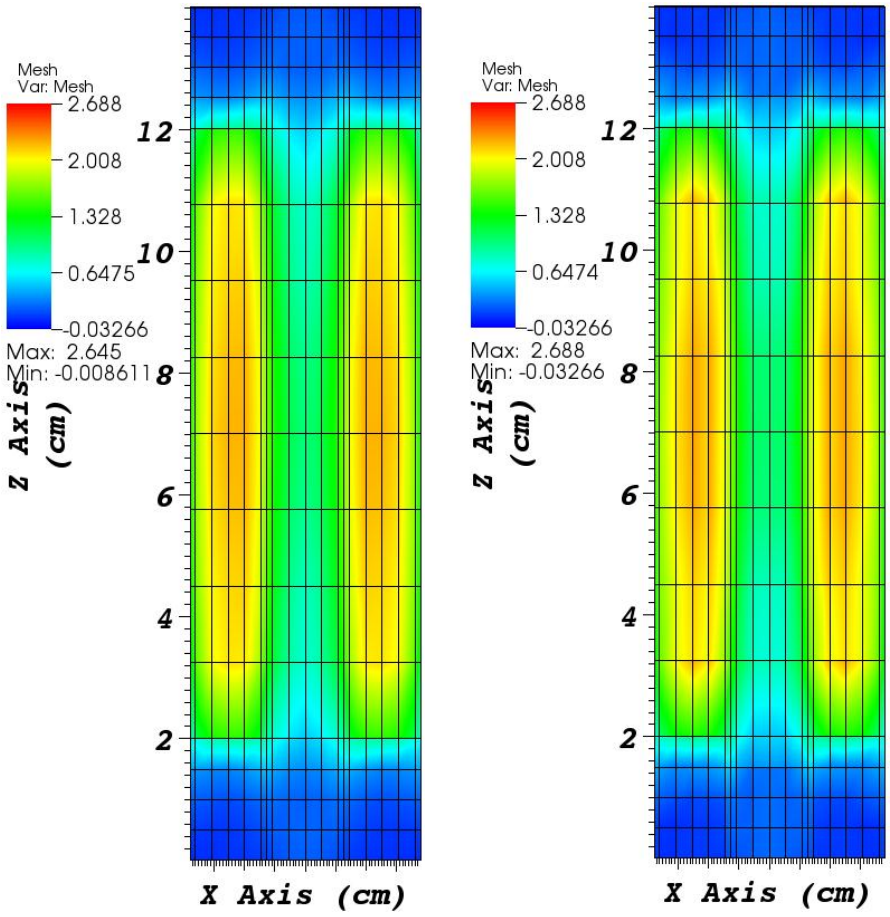
(a) **(b)**

Figure 31. Refined Pointwise Group 7 Scalar Flux Solution w/o Control Rod, at y = -0.6 cm (a) PWLD (b) PWL-LC

The PWLD method needs this refinement axially, regardless of quadrature order, to obtain the correct smooth solution for this energy group. Note that the PWL-LC solution

is smooth on this refined grid as well, similar to the smooth solution on the coarser grid shown in Figure 30. From the results of Figure 30 and Figure 31, it has been shown that the PWL-LC method can achieve an accurate solution on a coarser grid than the PWLD method, at least in some problems of interest.

Next, the scalar flux solution with the control rod inserted halfway is shown for both of these methods. Figure 32 shows a cross section of the pointwise scalar flux solution for group 1 at y = -0.6 cm using the PWLD and PWL-LC methods and an S_4 level-symmetric quadrature set. Similar to the results from Figure 28, the highest energy group scalar flux is depressed in the control rod and water regions and the PWLD and PWL-LC methods produce very similar solutions. Again, the PWL-LC method does not spread this highest energy-group scalar flux solution like the PWLD method does. Also, not shown here, the ray effects evident in the PWL-LC solution in Figure 32 diminish when an S_8 quadrature set is used.



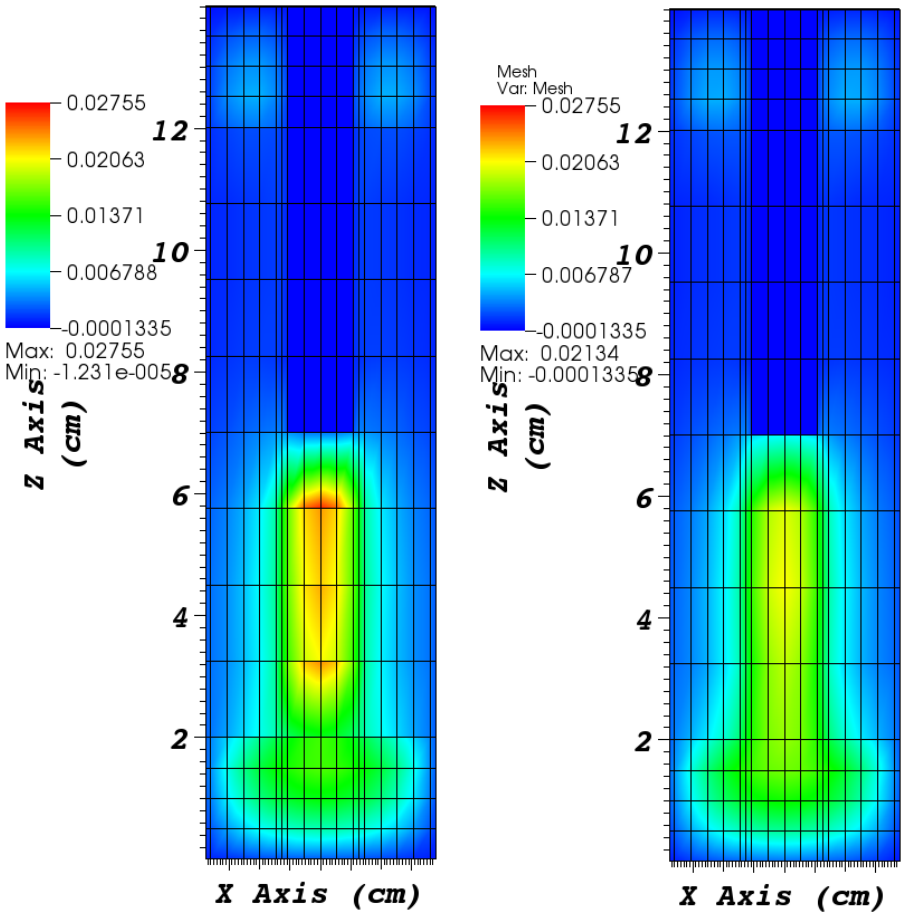
(a) **(b)**

Figure 32. Pointwise Group 1 Scalar Flux Solution w/Control Rod, at y = -0.6 cm

(a) PWLD (b) PWL-LC

A cross section of the pointwise scalar flux solutions for group 7 at y = -0.6 cm with the control rod inserted is shown in Figure 33. For both methods, this figure clearly shows that the scalar flux solution for this group is depressed in the control-rod region and peaks in the water region below as expected. Again the PWL-LC method produces a

smoother solution than the PWLD method with the PWLD method producing unphysical oscillations in the region below the control rod.



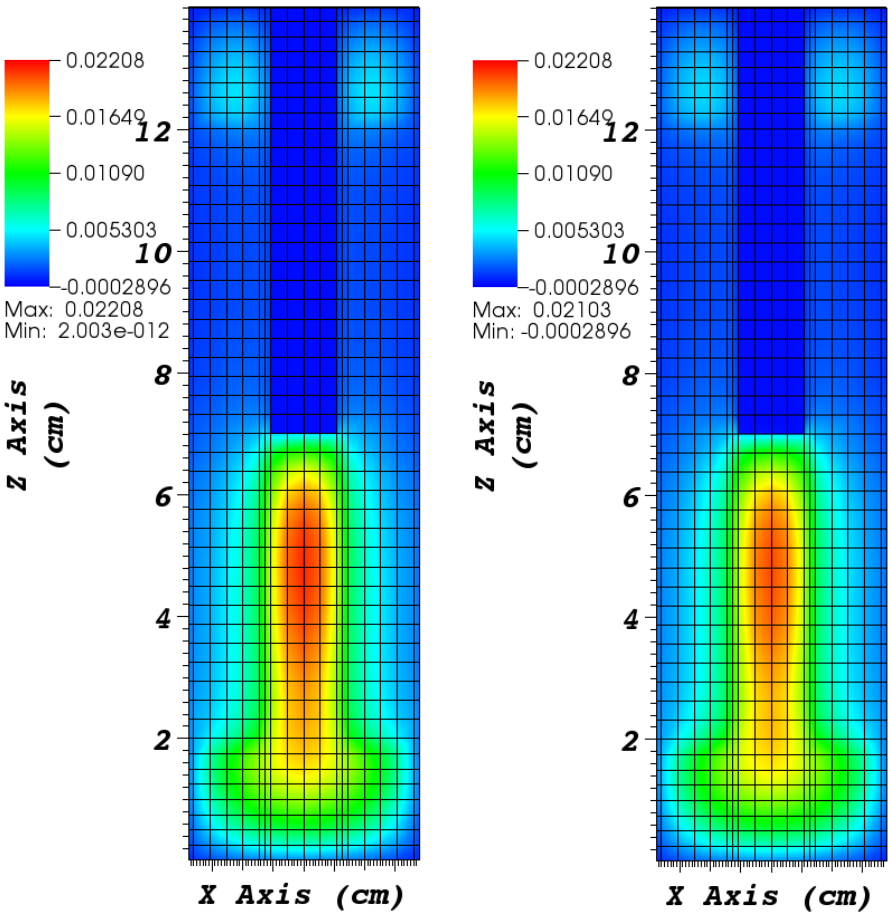
(a) **(b)**

Figure 33. Pointwise Group 7 Scalar Flux Solution w/Control Rod, at y = -0.6 cm

(a) PWLD (b) PWL-LC

Figure 34 shows a cross section of the pointwise scalar flux solutions for group 7 at y = -0.6 cm on a grid that has been refined in the axial dimension. From this figure, similar

to the results from the case in which the control rod was removed from the core, the PWLD method needs refinement axially to generate the correct (smooth) solution. Again, the PWL-LC method can find the correct solution on a coarser grid than the PWLD method.



(a) **(b)**

Figure 34. Refined Pointwise Group 7 Scalar Flux Solution w/Control Rod, at y = -0.6 cm (a) PWLD (b) PWL-LC

Overall, this test problem has shown that the PWL-LC method can achieve an accurate solution on a non-orthogonal reactor pin-cell grid in (x, y, z) and demonstrates potential accuracy advantages compared to PWLD. Table 5 gives an estimate of the total number of angular-flux unknowns that were calculated in this problem for the PWLD and PWL-LC methods on the coarse and refined grids. Note that the number of angular-flux unknowns in a problem depends on the track spacing chosen for the PWL-LC method.

Table 5. Approximate Total Angular Flux Unknowns in Reactor Pin Cell Problem

	PWLD Coarse Grid	**PWL-LC** Coarse Grid	**PWLD** Refined Grid	**PWL-LC** Refined Grid
Total Angular-Flux Unknowns	4.6449×10^6	1.8638×10^8	1.3935×10^7	2.2992×10^8

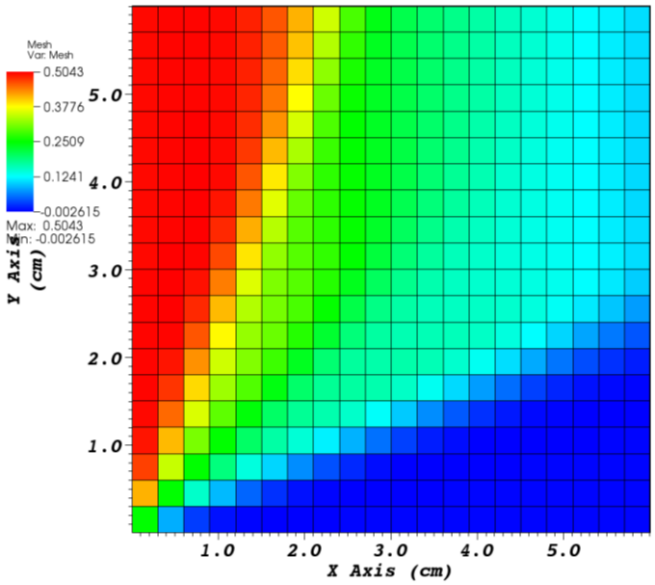
6.3 Time-Dependent PWL-LC Test Problems in X-Y

The test problems presented in this section show the properties of the PWL-LC method as applied to time-dependent problems using fully-implicit time differencing.

6.3.1 Streaming in a Vacuum

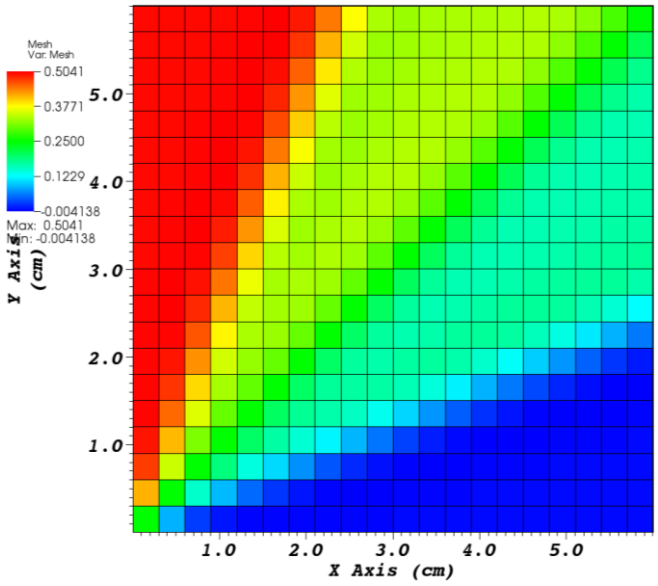
This test problem is a time-dependent version of the streaming problem in a vacuum that was presented in section 6.1.1. An initial condition of $\psi_{m,g}^{initial}\left(\vec{r}, t=0\right)=0$ is used for this fully-implicit method with a fixed time step such that $v\Delta t$ = 1.0 cm and the problem is run out to a final time of vT = 25 cm. All of the other properties defined in section 6.1.1 are used for this test problem. Figure 35 shows the PWL-DFEM solution with fully-implicit time-differencing for this test problem at an early time step and after

a steady-state has been established. This method spreads the solution and obtains negative cell-average flux solutions in some cells as was found in Figure 13(a) from section 6.1.1.



(a)

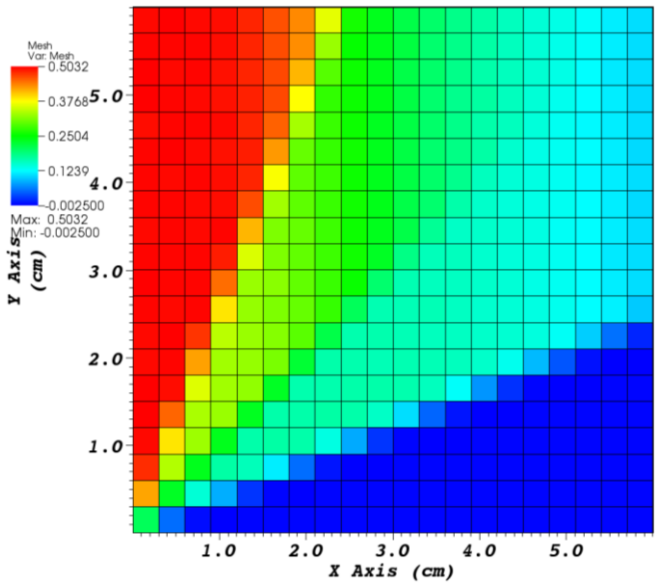
Figure 35. Cell-Average Scalar Flux for Incident Beam in a Vacuum (time-dependent) using PWL-DFEM (a) $vt \in (7.0, 8.0)cm$ **(b)** $vt \in (24.0, 25.0)cm$



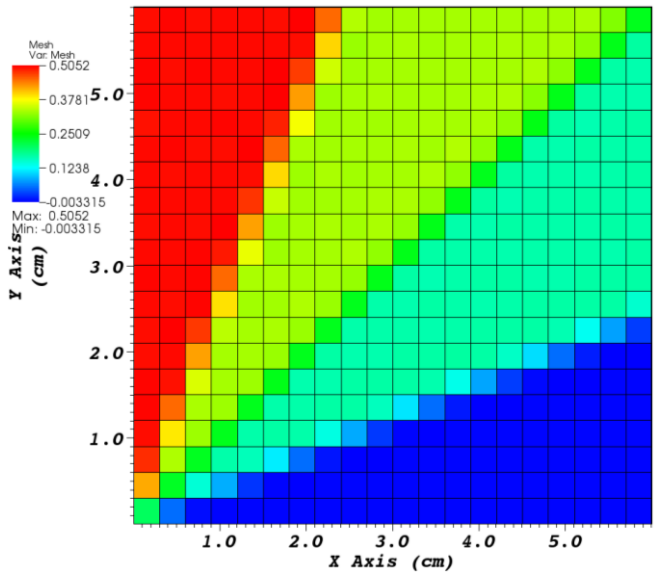
(b)

Figure 35. Continued

Figure 36 shows the cell-average scalar flux results using the PWL-LC method with fully implicit time-differencing at the same points in time as Figure 35. As expected, these results do not show as much spreading as associated with the PWL-DFEM, but the steady-state solution is closer to the PWL-DFEM solution than to the steady-state PWL-LC solution given in Figure 13. This result was predicted from the analysis of the finite-differenced PWL-LC method shown in section 2.3. Overall, this test problem has shown that the PWL-LC method with fully-implicit time-differencing does not approach the steady-state PWL-LC solution as desired, but instead approaches a PWLD solution as $v\Delta t$ becomes small.



(a)



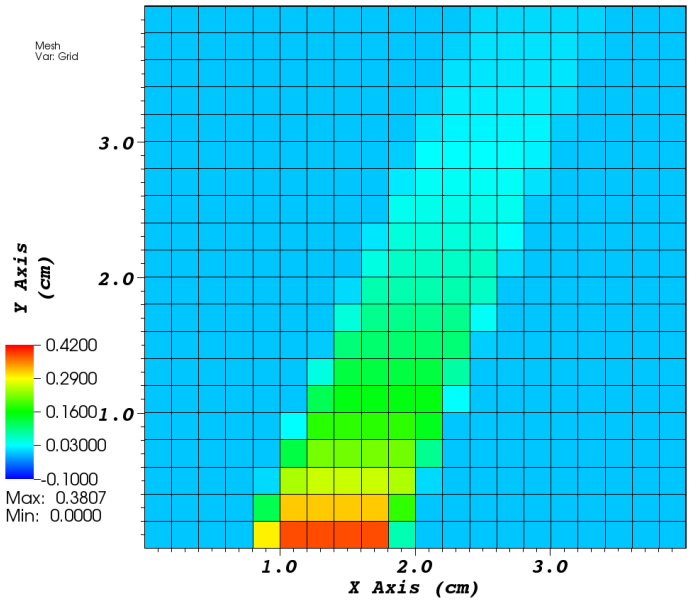
(b)

Figure 36. Cell-Average Scalar Flux for Incident Beam in a Vacuum (time-dependent) using PWL-LC (a) $vt \in (7.0, 8.0)cm$ **(b)** $vt \in (24.0, 25.0)cm$

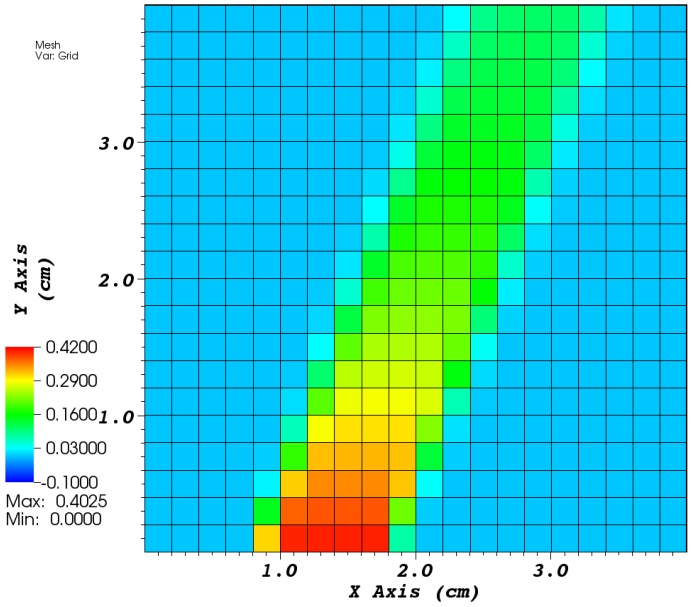
6.3.2 Streaming in a Pure Absorber

This test problem is a time-dependent version of the beam streaming problem presented in section 6.1.5. The speed of the neutrons in this problem is $v \approx 8.70x10^5$ cm/s (0.704 eV) with an initial condition of $\psi_{m,g}^{initial}\left(\vec{r}, t=0\right)=0$. Fully implicit time-differencing (Backward Euler) is used with the PWC-LC and PWL-LC methods as described in section 2.3, with angular-flux solutions projected onto constant or PWL bases at the end of each time step.

First, the solution from the fully-implicit PWC-LC method is shown for various time steps. This method approximates the time-*n* angular flux as a constant in each cell. Figure 37 shows the average scalar flux solution for various times with a large time step of Δt = 2.0 μs. The long-time steady solution given in Figure 37(b) is not the same as the steady-state PWC-LC solution given in Figure 24(a). This fully-implicit method "smears" the solution slightly because of the projection of the beginning-of-time-step solutions onto cell-wise constants and because of the implicit finite differencing in time. However, with a large time step this smearing is minimal and this time-dependent solution is close to the steady-state solution.



(a)



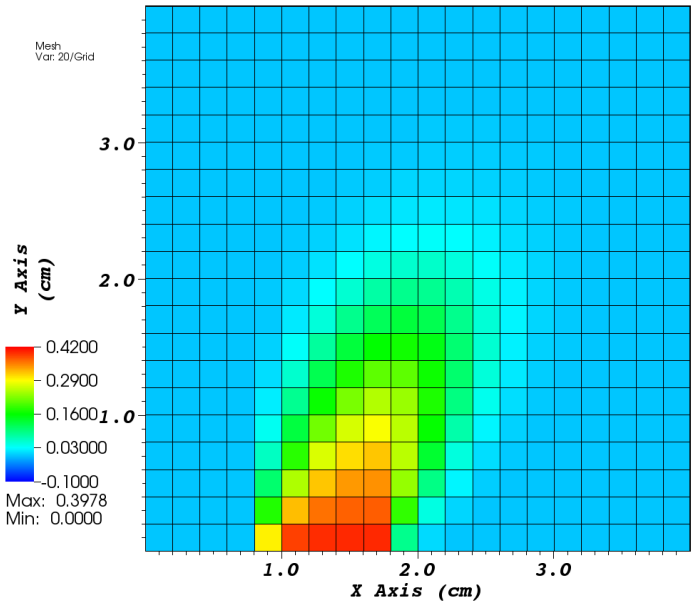
(b)

Figure 37. Fully-Implicit PWC-LC Cell-Average Scalar Flux for Beam in Pure Absorber with Δt = 2.0 μs (a) $t \in (0.0, 2.0)\,\mu s$ **(b)** $t \in (18.0, 20.0)\,\mu s$

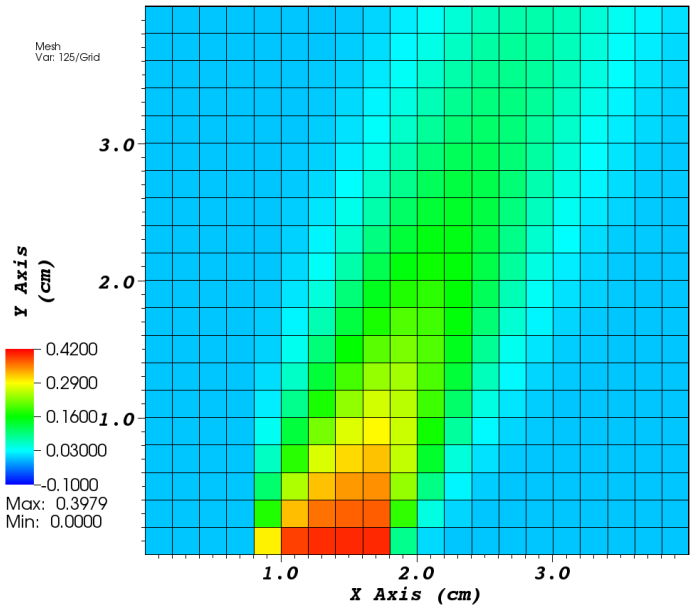
Figure 38 shows the cell-average flux solution using the PWC-LC method at various times with a small time step of $\Delta t = 0.1\ \mu s$. It can be clearly seen that the long-time steady solution in Figure 38(b) is very different from the steady-state PWC-LC solution given in Figure 24(a). Figure 38 underscores the observation from analysis that the steady-state solution will approach the (highly diffusive) PWC-DFEM solution as the time step becomes small. It has been verified that the steady-state PWC-DFEM solution is very close to the solution shown in Figure 38(b). In other words, the "extra" collision and source terms that appear in the time-differenced equation do not cancel each other in steady-state because the source term is a cell-wise constant. That is, for this PWC-LC method, the following terms are not zero.

$$\frac{\bar{\Psi}_k^{n+1/2} - \bar{\psi}_{cell}^n}{v\theta\Delta t^n} \neq 0 \tag{6.13}$$

Therefore, the steady-state solution reached by a time-dependent run of this PWC-LC method is dependent on the size of the time step for the given finite-difference time discretization.



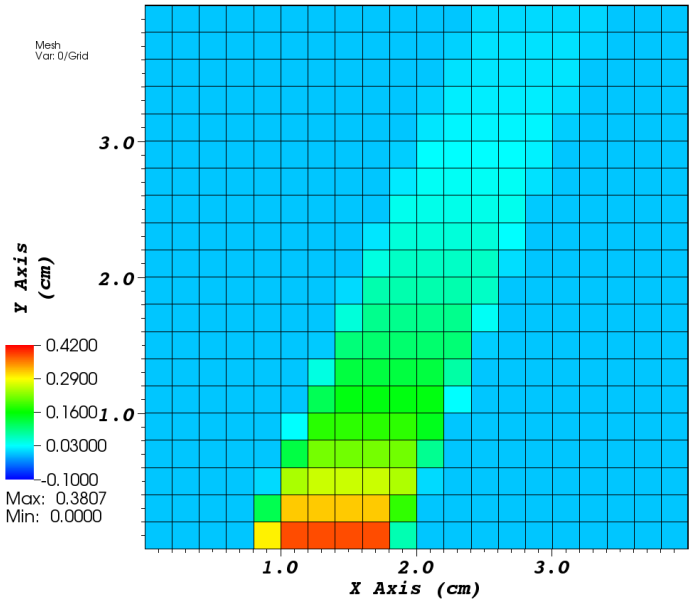
(a)



(b)

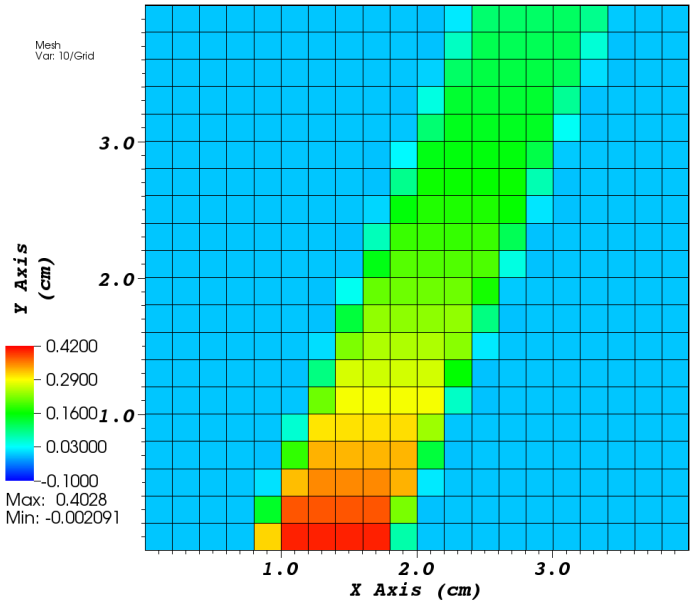
Figure 38. Fully-Implicit PWC-LC Cell-Average Scalar Flux for Beam in Pure Absorber with Δt = 0.1 μs (a) $t \in (1.9, 2.0)\,\mu s$ **(b)** $t \in (19.9, 20.0)\,\mu s$

Next, the solution from the fully-implicit PWL-LC method is shown for the same two time steps. Figure 39 shows the cell-average scalar flux solution from the PWL-LC method with a time step of $\Delta t = 2.0\ \mu s$. The long-time steady solution in Figure 39(b) is very close to the PWL-LC steady-state solution given in Figure 24(a). Note again the difference in solution is due to the dependence on the time-step and minimal smearing that occurs since the beginning-of-time-step solution is approximated as PWL in a cell.



(a)

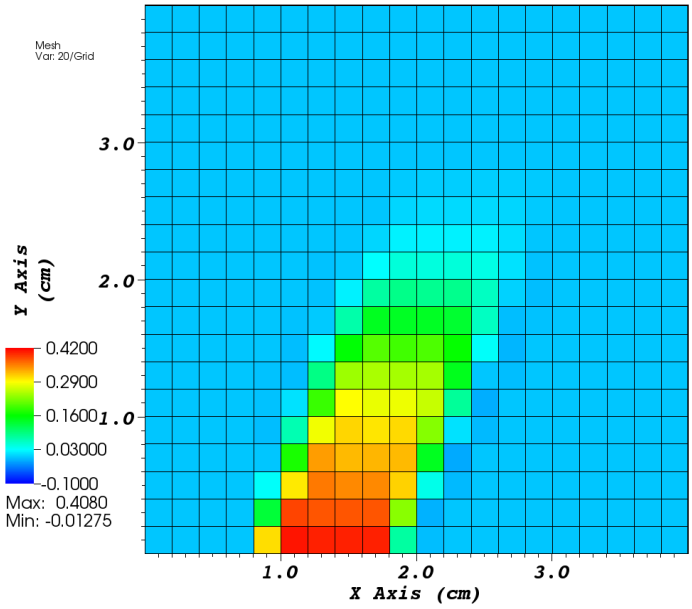
Figure 39. Fully-Implicit PWL-LC Cell-Average Scalar Flux for Beam in Pure Absorber with $\Delta t = 2.0\ \mu s$ (a) $t \in (0.0, 2.0)\ \mu s$ (b) $t \in (18.0, 20.0)\ \mu s$



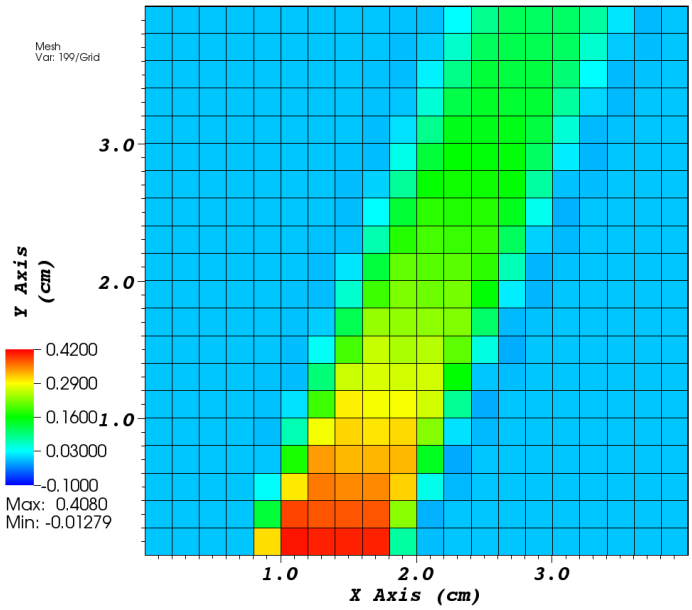
(b)

Figure 39. Continued

Figure 40 shows the cell-average flux solution using from the PWL-LC method with a small time step of $\Delta t = 0.1\ \mu s$. As expected from the analysis of section 2.3, the long-time steady solution shown in Figure 40(b) is very similar to the steady-state PWL-DFEM solution in Figure 24(b).



(a)



(b)

Figure 40. Fully-Implicit PWL-LC Cell-Average Scalar Flux for Beam in Pure Absorber with Δt = 0.1 μs (a) $t \in (1.9, 2.0)\,\mu s$ **(b)** $t \in (19.9, 20.0)\,\mu s$

The results from the finite-differenced PWL-LC method applied to this beam streaming problem are a strong motivating factor for developing the STLC method.

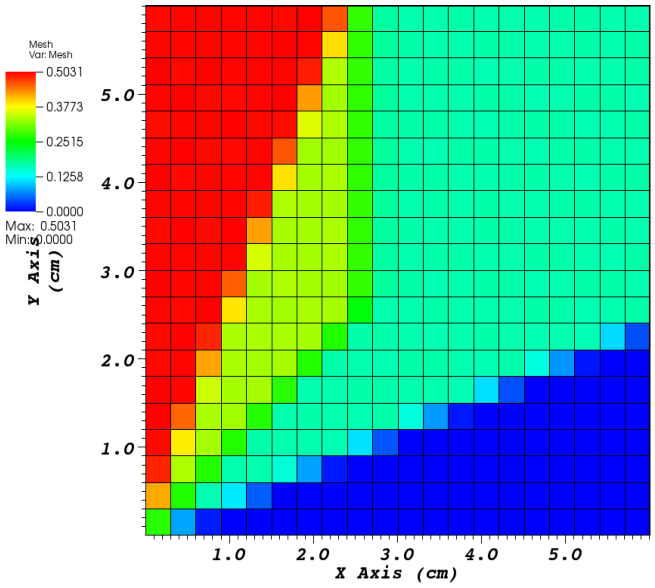
6.4 STLC Test Problems in X-Y-vT

This subsection presents time-dependent test problems in *x-y-vt* space that employ the STLC method with various source approximations.

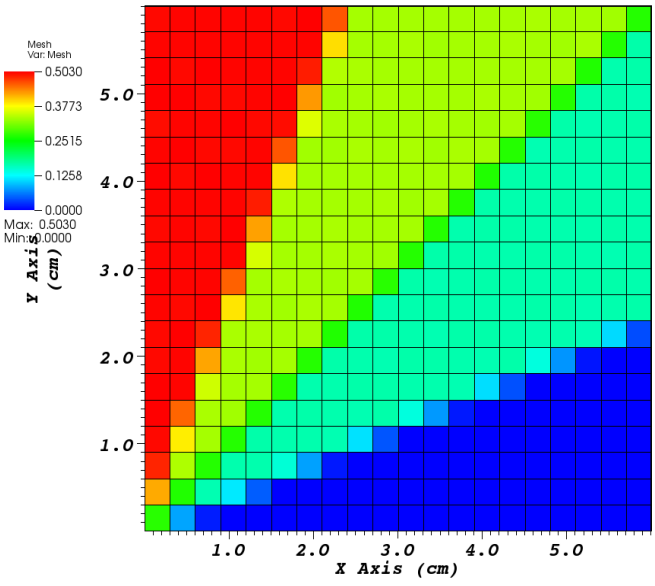
6.4.1 Streaming in a Vacuum

Consider the test problem given in section 6.3.1. This problem starts with an initial condition $\psi_{m,g}^{initial}\left(\vec{r},t=0\right)=0$ and a fixed time step with $v\Delta t$ = 1.0 cm. The problem is run out to a final time of vT = 25 cm. A track spacing of $\Delta\omega = \Delta u$ = 0.05 cm is used for the STLC method.

Figure 41 shows the cell-average scalar flux results using the PPWL-STLC method at an early time step and once steady-state has been reached. As expected, neither the early time step solution nor the steady-state solution in Figure 41 show spreading of the solution with this PPWL-STLC method. Also, this method does not produce negative cell-average scalar flux values, unlike the finite-differenced PWL-DFEM and PWL-LC method solutions. Furthermore, the steady-state solution given in Figure 41(b) is very similar to the steady-state solution from the PWL-LC method of Figure 13(b). The small difference between these solutions comes from the slight oscillation around a steady solution that can occur with the PPWL-STLC method without very fine track spacing, due to the different track areas on different time boundaries.



(a)



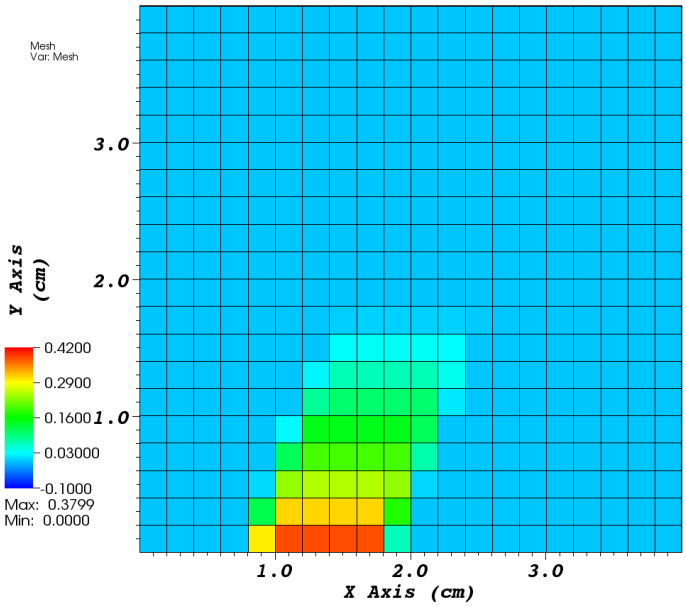
(b)

Figure 41. Cell-Average Scalar Flux for Incident Beam in a Vacuum (time-dependent) using PPWL-STLC (a) $vt \in (7.0, 8.0)cm$ **(b)** $vt \in (24.0, 25.0)cm$

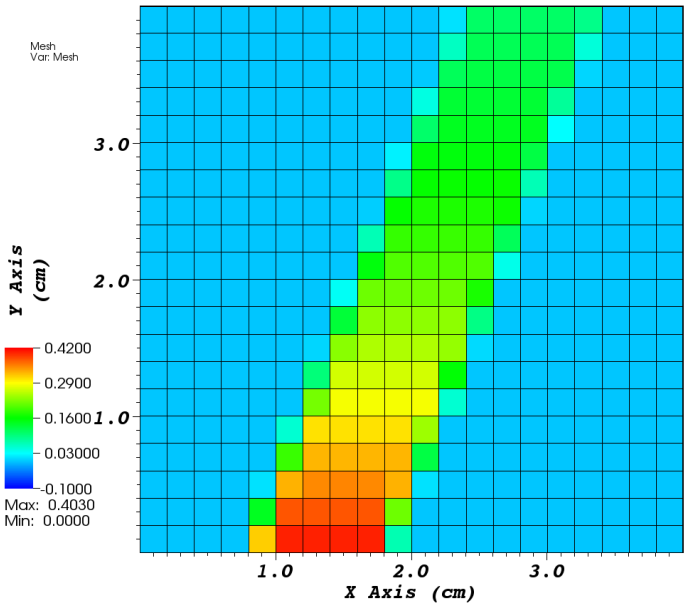
This test problem has shown that this PPWL-STLC method does not spread the solution or find negative cell-average flux solutions for problems involving streaming in a vacuum and also produces a solution much more consistent with steady-state than the fully-implicit PWL-LC method solution.

6.4.2 Streaming in a Pure Absorber

This next test problem revisits the beam streaming in a pure absorber that was presented in sections 6.1.5 and 6.3.1. The same properties and zero initial condition as defined in these previous sections apply. A fine track spacing is used and is equal to $\Delta\omega = \Delta u = 0.01$ cm. Figure 42 shows the cell-average scalar flux solution using the PWC-STLC method with a large time step of $\Delta t = 2.0$ μs. The slight smearing that was associated with the PWC-LC solution is not present with this PWC-STLC solution. Also the long-time steady solution shown in Figure 42(b) agrees with the PWC-LC steady-state solution given in Figure 24(a), as expected for this absorption streaming problem.



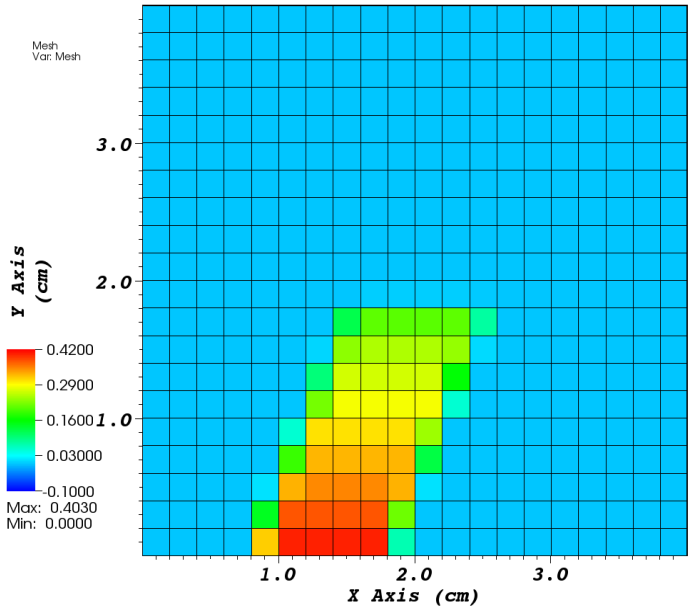
(a)



(b)

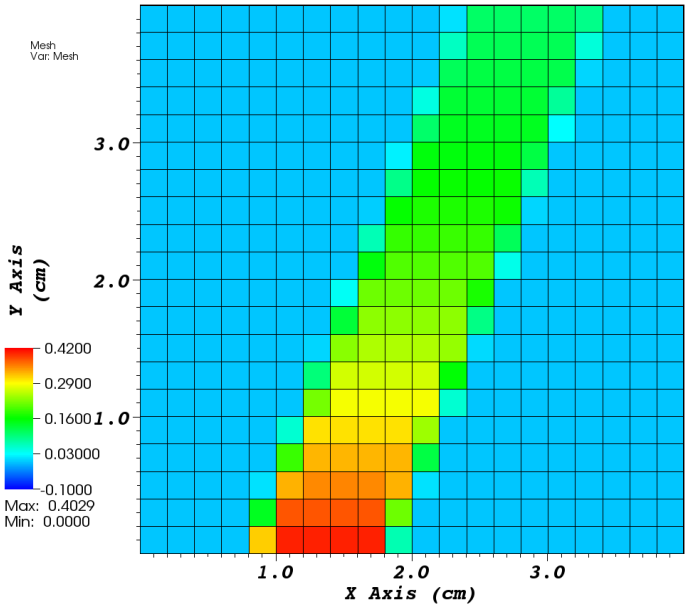
Figure 42. PWC-STLC Cell-Average Scalar Flux for Beam in Pure Absorber with $\Delta t = 2.0\ \mu s$ (a) $t \in (0.0, 2.0)\ \mu s$ (b) $t \in (18.0, 20.0)\ \mu s$

Figure 43 shows the cell-average PWC-STLC scalar flux solution with a small time step of $\Delta t = 0.1\ \mu s$. Again, the long-time solution given in Figure 43(b) agrees with the steady-state PWC-LC solution given in Figure 24(a), as expected for this small time step. Note that this solution is not dependent on the time step, superior to the finite-differenced PWC-LC method.



(a)

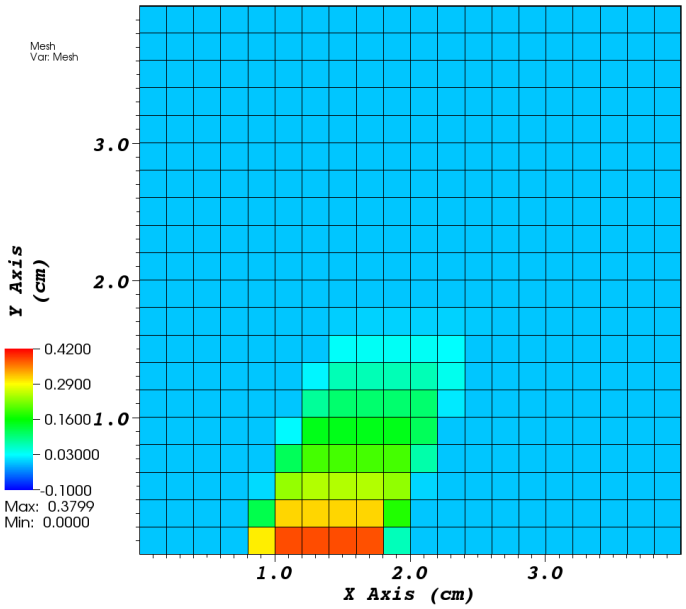
Figure 43. PWC-STLC Cell-Average Scalar Flux for Beam in Pure Absorber with $\Delta t = 0.1\ \mu s$ (a) $t \in (1.9, 2.0)\ \mu s$ (b) $t \in (19.9, 20.0)\ \mu s$



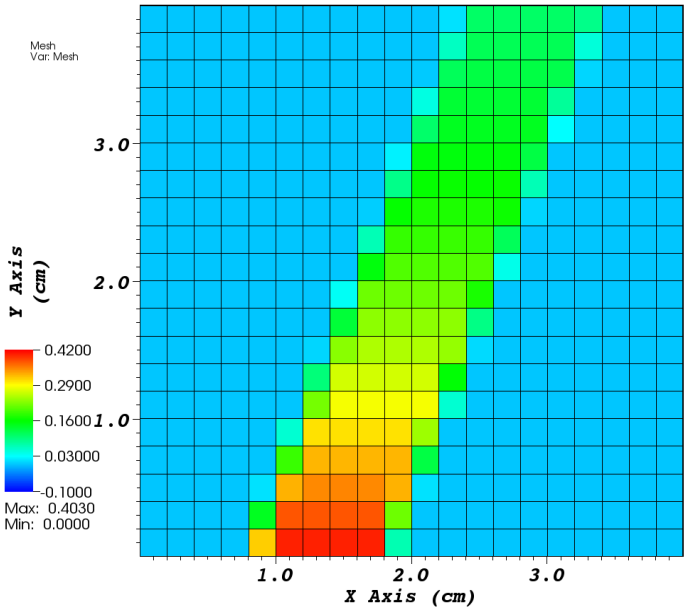
(b)

Figure 43. Continued

Next, the solution from the PPWL-STLC method is compared to the steady-state LC solution for various time steps. Figure 44 shows the cell-average PPWL-STLC flux solution with a large time step of $\Delta t = 2.0\ \mu s$. The long-time steady solution shown in Figure 44(b) agrees with the LC steady-state solution given in Figure 24(a) as well as with the previous PWC-STLC steady solutions for both time steps.



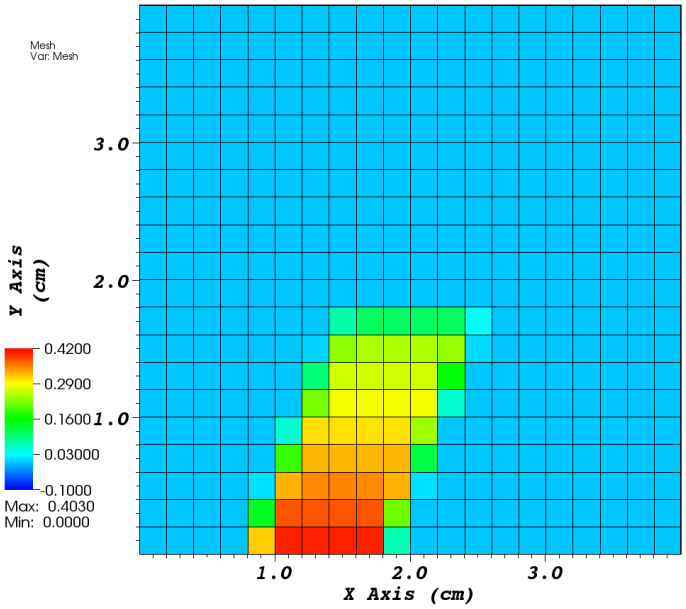
(a)



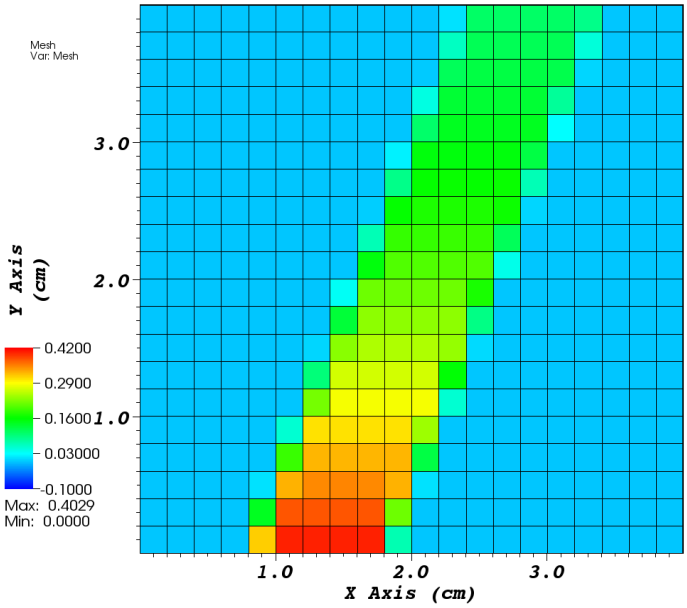
(b)

Figure 44. PPWL-STLC Cell-Average Scalar Flux for Beam in Pure Absorber with $\Delta t = 2.0\ \mu s$ (a) $t \in (0.0, 2.0)\ \mu s$ (b) $t \in (18.0, 20.0)\ \mu s$

Figure 45 shows the PPWL-STLC cell-average flux solution for a small time step of $\Delta t = 0.1\ \mu s$. The long-time steady solution given in Figure 45(b) agrees with the steady solution of Figure 44(b) and the LC steady-state solution given in Figure 24(a). Note that this solution does not approach a corresponding DFEM solution as the finite-differenced PWL-LC solution did for small time steps; it reproduces the steady-state PWL-LC solution. Also, it should be noted that the small differences that occur between the STLC solutions and the steady-state LC solution can be attributed to the track spacing. The track area associated with the tracks that enter a cell on a given time-step boundary is usually different from the track area that enters this same cell on the time-step boundary of the next time step. This difference in tracks entering each time-step can lead to a long-time steady-state solution that slightly oscillates around a steady-state solution. This is due to the track spacing and can be mitigated by using very fine track spacing. However, even with fine track spacing, the steady-state solution for some time-dependent problems may still be "bumpy" and this issue needs further investigation.



(a)



(b)

Figure 45. PPWL-STLC Cell-Average Scalar Flux for Beam in Pure Absorber with $\Delta t = 0.1\ \mu s$ (a) $t \in (1.9, 2.0)\ \mu s$ (b) $t \in (19.9, 20.0)\ \mu s$

6.4.3 Linear Manufactured Solution

This next test problem is similar to the manufactured linear solution test problem from section 6.1.6. It will demonstrate that the PPWL-STLC method finds the exact solution for a problem with a linearly anisotropic solution that is also linear in *x*, *y*, and *t*. The same process of the method of manufactured solutions is used to define the boundary, initial, and fixed source conditions for this test problem. The manufactured linear angular flux solution for this problem is given in Eq. (6.14).

$$\Psi_{mg}\left(x,y,t\right)=\psi_0+\left(x-\frac{\mu_m}{\sigma_{t,g}}\right)\psi_x+\left(y-\frac{\eta_m}{\sigma_{t,g}}\right)\psi_y+t\psi_t \tag{6.14}$$

With this angular flux solution, the scalar flux solution can be found as given in Eq. (6.15), assuming a symmetric quadrature set that sums to 4π.

$$\phi_g\left(x,y,t\right)=\sum_{m=1}^{M}w_m\Psi_{mg}\left(x,y,t\right)=4\pi\psi_0+4\pi x\psi_x+4\pi y\psi_y+4\pi t\psi_t \tag{6.15}$$

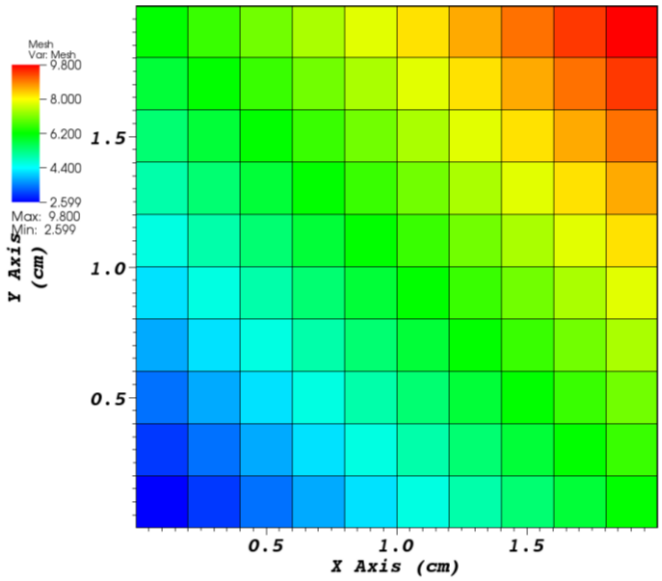
The source condition needed to produce the linear angular flux solution above is given in Eq. (6.16).

$$q_{g,\,fixed}\left(x,y,t\right)=\sigma_{a,g}\psi_0+\sigma_{a,g}x\psi_x+\sigma_{a,g}y\psi_y+\left(\sigma_{a,g}t+\frac{1}{v}\right)\psi_t \tag{6.16}$$

The same layout and setup of the test problem is used here as was given in section 6.1.6. A fixed time step of $v\Delta t$ = 0.2 cm is used and the problem is run up to a maximum time of vT = 2.0 cm. The boundary and initial conditions are given based upon Eq. (6.14) with a fixed source in each cell and time step given by Eq. (6.16) using the following coefficients.

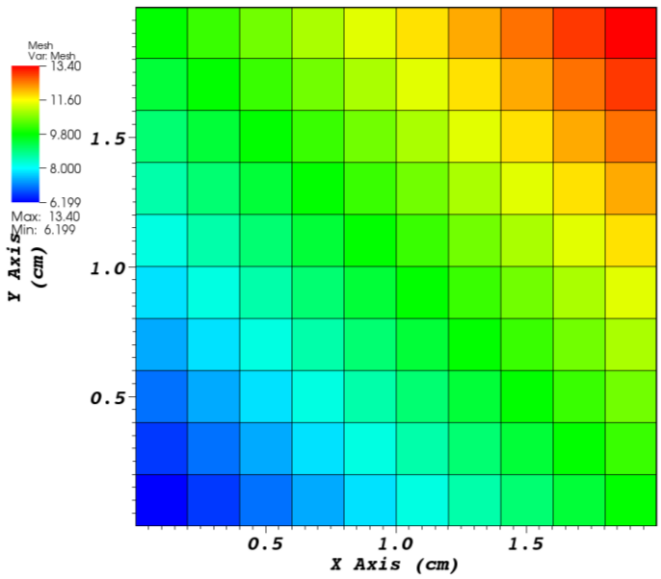
$$\psi_0 = 2.0 \quad \frac{n}{cm^2 - s - str}$$
$$\psi_x = \psi_y = 2.0 \quad \frac{n}{cm^3 - s - str} \tag{6.17}$$
$$\psi_t = 2.0 \quad \frac{n}{cm^2 - s^2 - str}$$

Figure 46 shows the cell-average scalar flux solution using the PPWL-STLC method for the full problem at two different time steps. Figure 47 shows the pointwise scalar flux solution (plotted at each cell element) at the same two time steps. These figures show that this PPWL-STLC method exactly reproduces the linear average scalar flux solution of Eq. (6.15) (up to a given tolerance on the residual of the iterative algorithm and the pointwise solution) as expected.



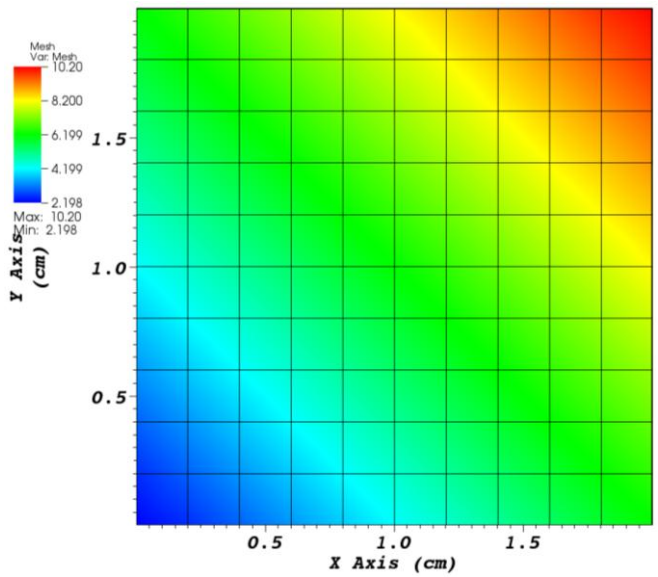
(a)

Figure 46. Time-Dependent Cell-Average Linear PPWL-STLC Scalar Flux Solution (a) *vt* = 0.1 cm (b) *vt* = 1.9 cm



(b)

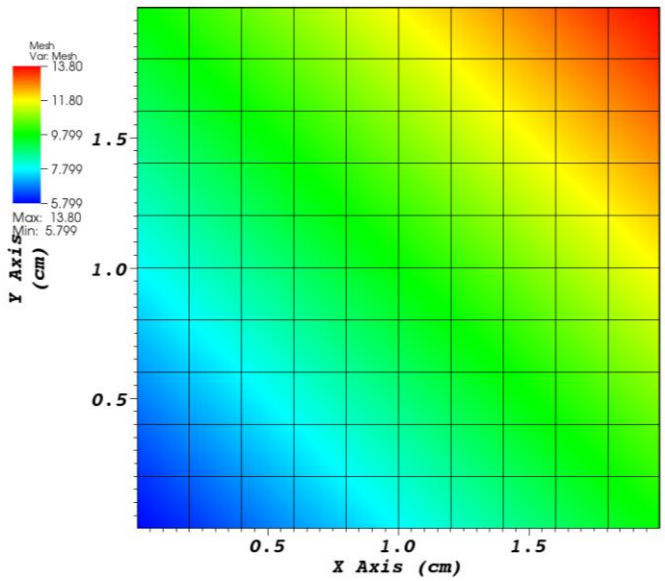
Figure 46. Continued



(a)

Figure 47. Time-Dependent Pointwise Linear PPWL-STLC Scalar Flux Solution

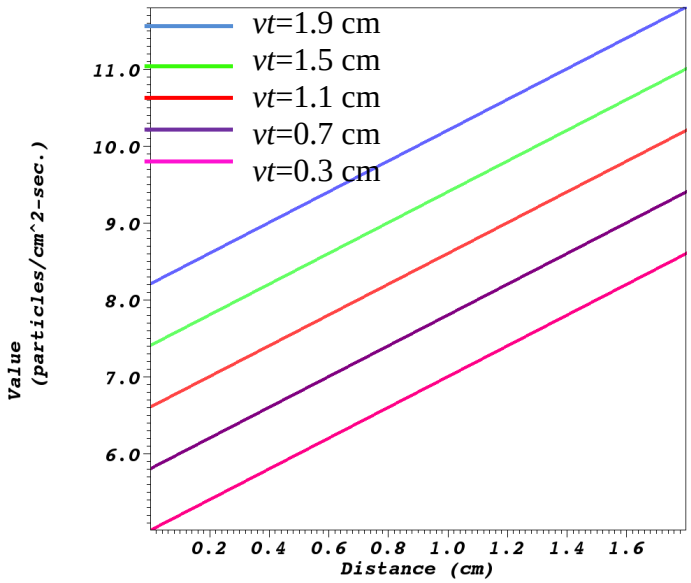
(a) *vt* = 0.1 cm (b) *vt* = 1.9 cm



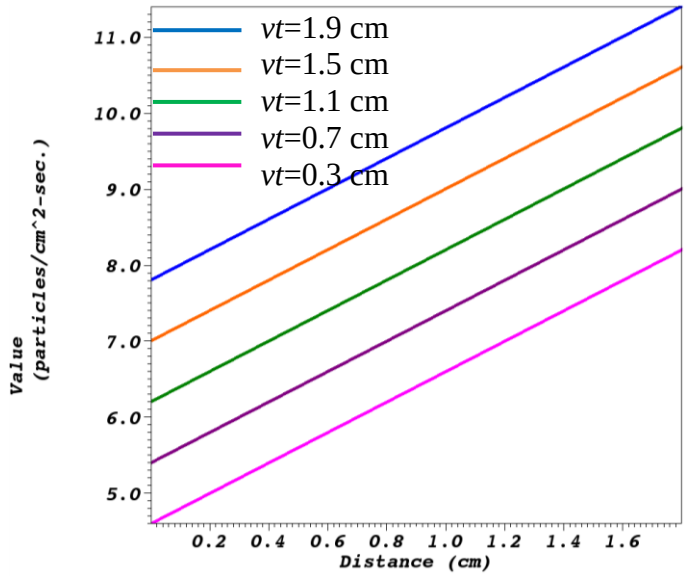
(b)

Figure 47. Continued

To show the linear variation in time, Figure 48 shows lineout plots of the average and pointwise scalar flux solutions at a given *y* position for various time values. Both of these plots show how this method captures the linear dependence of the solution in space and time. These results are expected because the PPWL-STLC method assumes a spatial PWL dependence and linear time dependence of the scattering source in each space-time cell.



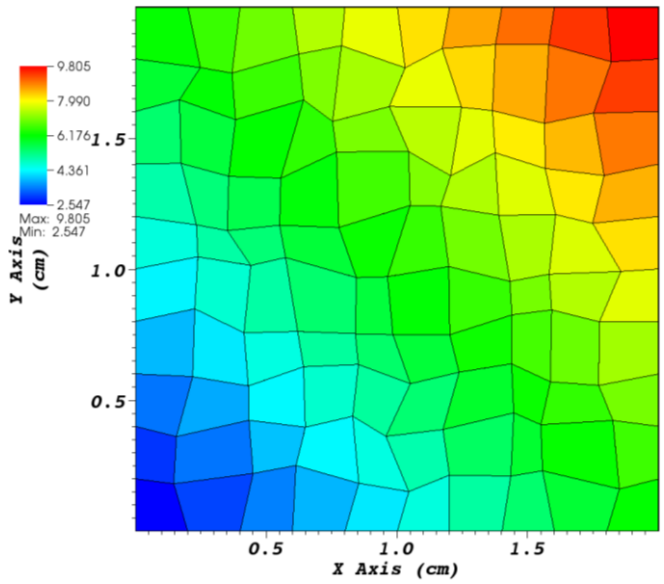
(a)



(b)

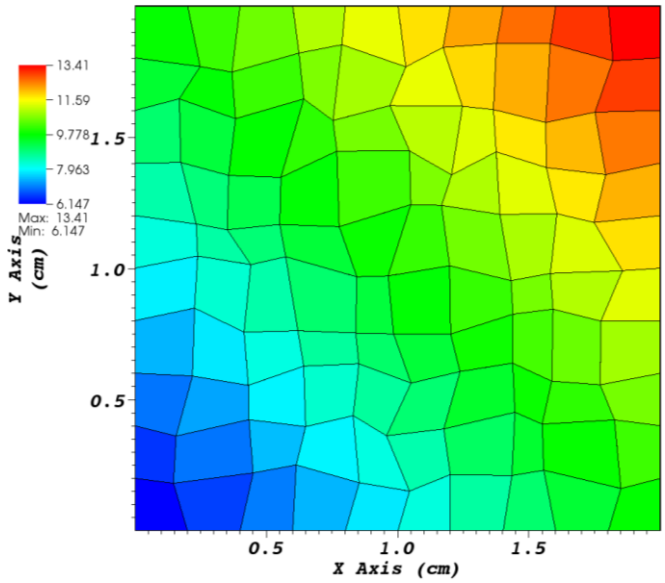
Figure 48. Time-Dependent PPWL-STLC Cross-Section Scalar Flux Solution (a) Cell-Average at y = 1.1 cm (b) Pointwise at y = 1.0 cm

This same test problem was also run using a non-orthogonal grid in x and y. Figure 49 shows the cell-average scalar flux solution using the PPWL-STLC method on this non-orthogonal grid at the same two time steps from Figure 46. Figure 50 shows the pointwise scalar flux solution at the same two time steps on this non-orthogonal grid. As expected this PPWL-STLC method exactly reproduces the linear average scalar flux solution of Eq. (6.15) (up to a given tolerance on the residual of the iterative algorithm and the pointwise solution) on this non-orthogonal grid as seen from the solutions of Figure 49 and Figure 50.



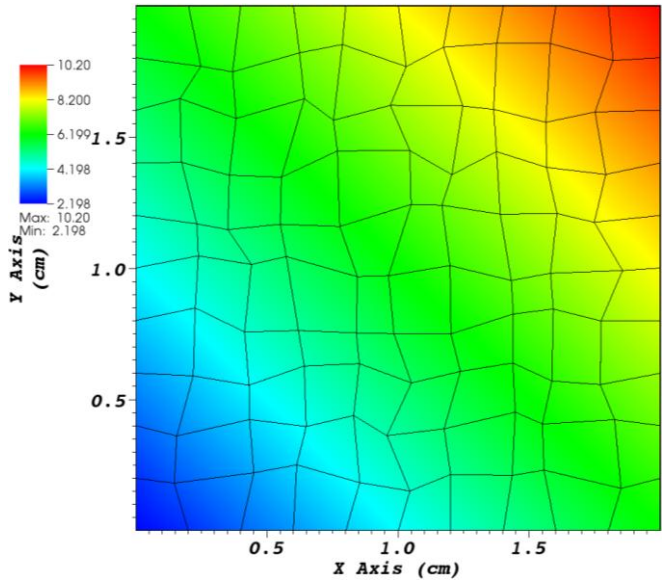
(a)

Figure 49. Time-Dependent Cell-Average Linear PPWL-STLC Scalar Flux Solution on Non-Orthogonal Grid (a) vt = 0.1 cm (b) vt = 1.9 cm



(b)

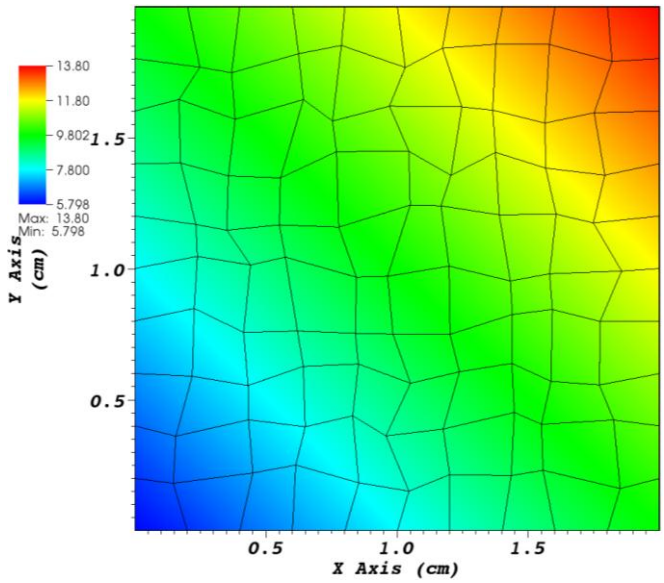
Figure 49. Continued



(a)

Figure 50. Time-Dependent Pointwise Linear PPWL-STLC Scalar Flux Solution

(a) *vt* = 0.1 cm (b) *vt* = 1.9 cm



(b)

Figure 50. Continued

This test problem has shown that the PPWL-STLC method can find an exact linear solution in space, time, and angle on both orthogonal and non-orthogonal grids.

6.4.4 The Crooked Pipe (TopHat) Problem

This radiative transfer test problem is a Cartesian geometry (x, y) version of the crooked pipe, or top-hat, problem originally developed by Graziani and LeBlanc[47]. The problem consists of an optically thick and optically thin region as shown in Figure 51. This figure also shows the orthogonal spatial grid used to solve this radiative transfer problem which is composed of 28x16 uniform cells of size $\Delta x = \Delta y = 0.25$ cm. Table 6 gives the physical properties of each material region of the top-hat problem. All boundaries are vacuum except for an incoming Planckian source incident on the

optically thin region on the left at a temperature of $T_{incident}$ = 3.48135x10^6 K, or $kT_{incident}$ = 300 eV.

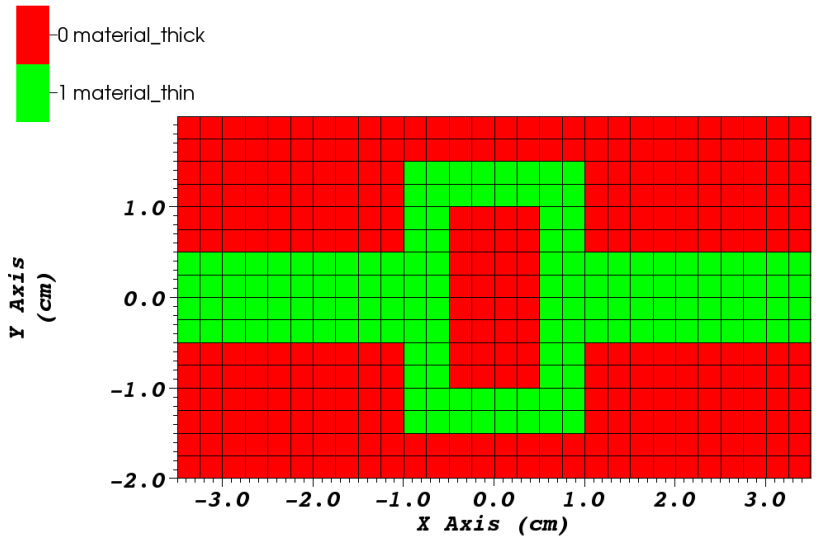


Figure 51. Material and Geometry Layout for Top-Hat Problem

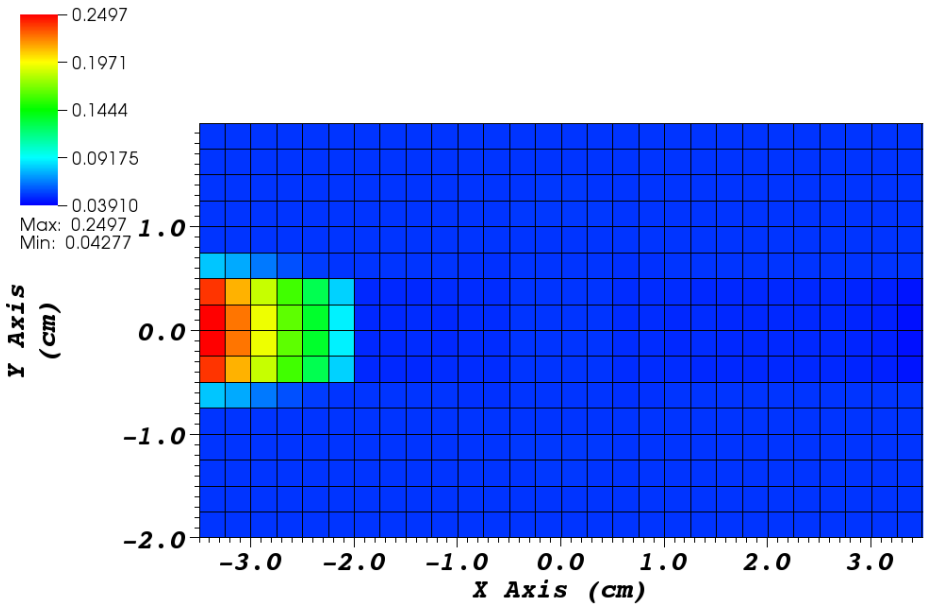
A product Gauss Legendre Chebyshev quadrature set is used to solve this problem with 2 polar angles and 20 azimuthal angles per quadrant. GMRES iteration is used to converge the absorption rate density (ARD) to a tolerance of $r_{2,rel}$ < 1E-06. The ARD is not converged pointwise (only 1 pointwise ARD iteration per temperature iteration), but the temperature is converged pointwise to a tolerance of $d_{2,rel}$ < 1E-03. This test problem was run on 56 processors using the Cab system at Lawrence Livermore National Laboratory. This system has an Intel Xeon architecture with 1,296 nodes housing 20,736 cores.

Table 6. Material Properties of Top-Hat Problem

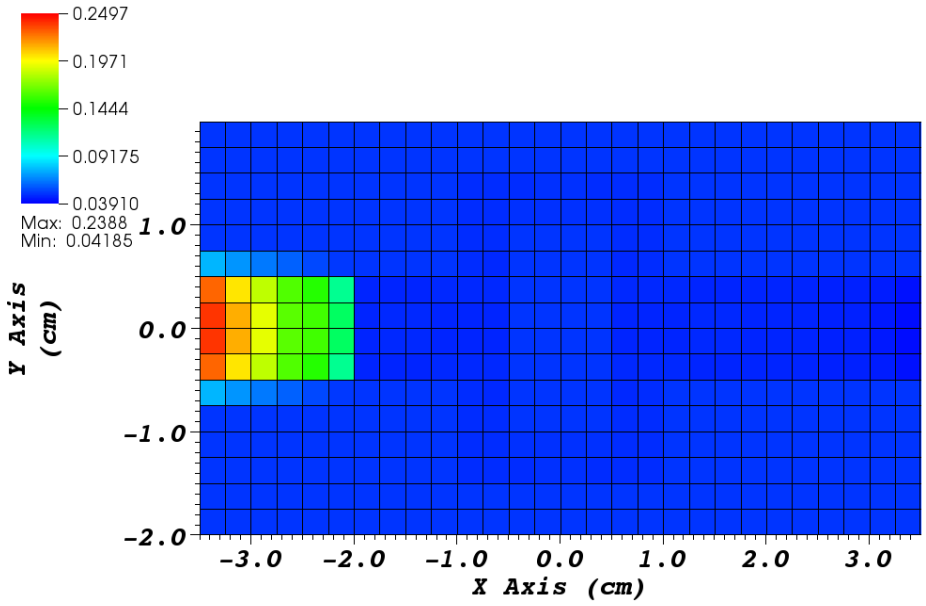
Property	Material_Thin	Material_Thick
Density [g/cm^3]	0.01	10.0
Opacity [cm^2/g]	20.0	20.0
$\mathbf{T_{initial}}$ [K]	5.80225E05	5.80225E05
Specific Heat [eV/g-K]	2.68926E19	2.68926E19

The track spacing used for the PPWL-STLC method on this top-hat problem is $\Delta\omega = \Delta u = 0.04$ cm. It should be noted that, with this fixed track spacing, as the time step becomes large, the number of tracks incoming and exiting on the spatial boundaries of the (x, y, vt) cell also becomes large. This large number of tracks makes this STLC method computationally expensive. All of these tracks in one step may or may not be needed depending on the problem. See recommendations for future work, section 8, for a further discussion on handling the increase in the number of tracks as time step increases.

The following set of figures shows temperature solutions at an early time in the problem using various fixed time step sizes. The purpose of these figures is to show that the PPWL-STLC method performs better than other methods in the streaming region of a radiative transfer problem. Figure 52 shows the average radiation temperature in each cell at an early time of T = 0.05 ns using the PPWL-STLC method with a time step size of $\Delta t = 0.05$ ns and $\Delta t = 0.01$ ns.



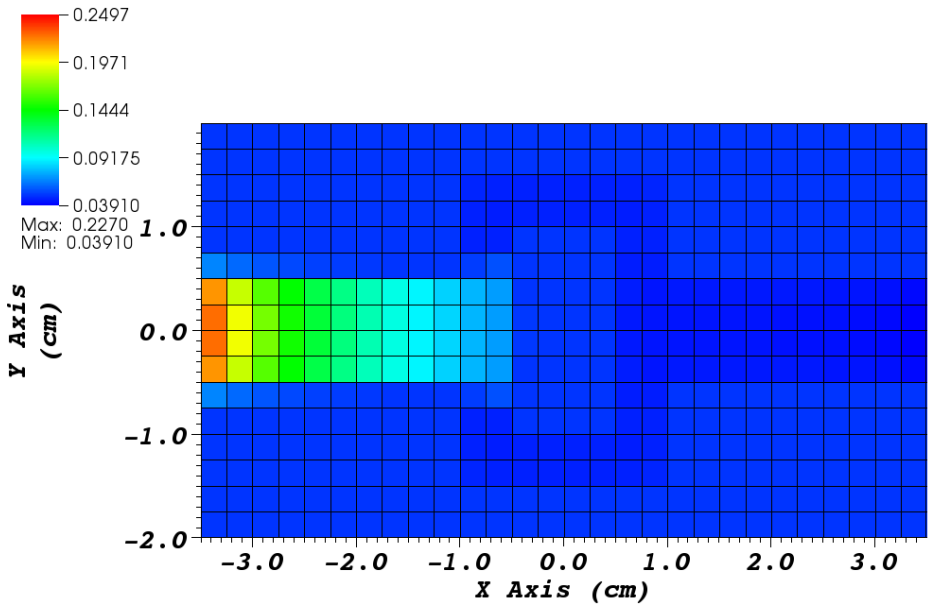
(a)



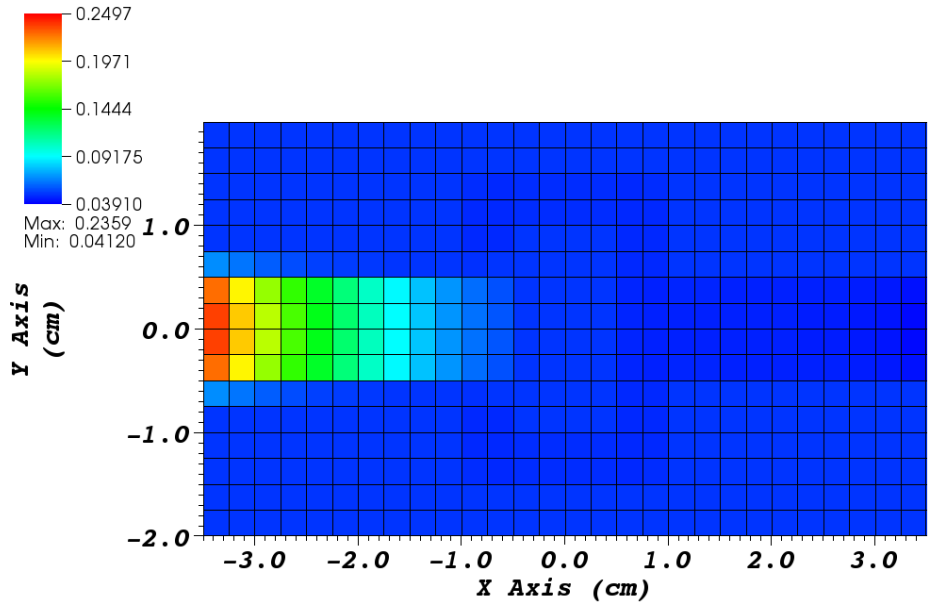
(b)

Figure 52. PPWL-STLC Cell-Average Radiation Temperature at T = 0.05 ns (a) Δt = 0.05 ns (b) Δt = 0.01 ns

These figures can be compared to the average radiation temperature solutions using the fully-implicit PWLD method and PWLD with TBDF2 time-differencing shown in Figure 53 and Figure 54 at the same time with the same time-step sizes. Note that the PWLD methods spread/smear the solution more than the PPWL-STLC method, as expected. The TBDF2 solution is closer to the PPWL-STLC solution, because it is more accurate than fully-implicit, but the PWLD method still smears this solution spatially and TBDF2 smears it in time. Due to the spreading of the solution, the PWLD methods must take a smaller time step to approach the correct solution, which the PPWL-STLC method obtains with a larger time step. For example, the PPWL-STLC solution at a time step of Δt = 0.05 ns, shown in Figure 52(a), is still more accurate than the PWLD solutions at a smaller time step of Δt = 0.01 ns shown in Figure 53(b) and Figure 54(b).

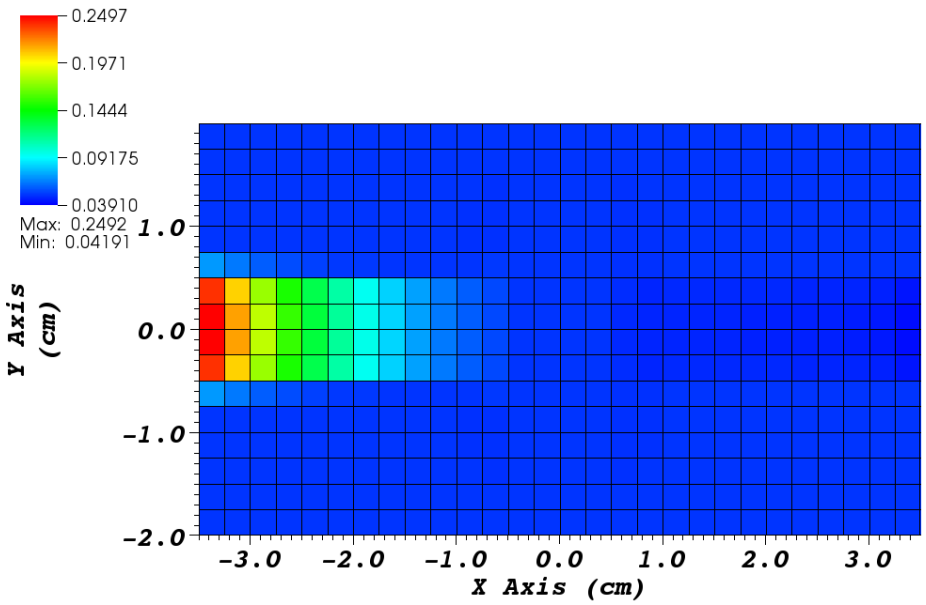


(a)

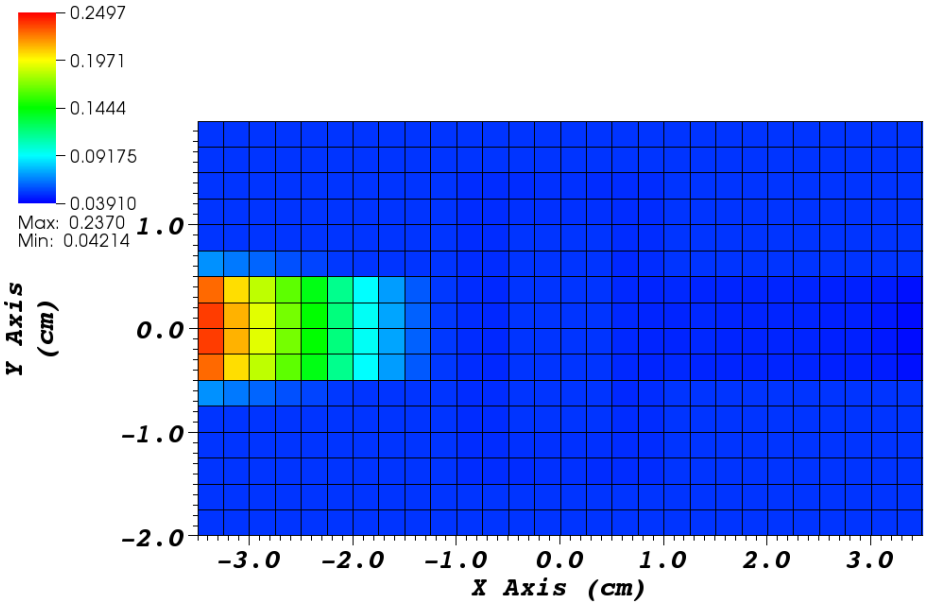


(b)

Figure 53. Fully-Implicit PWLD Cell-Average Radiation Temperature at T = 0.05 ns (a) Δt = 0.05 ns (b) Δt = 0.01 ns



(a)



(b)

Figure 54. TBDF2 PWLD Cell-Average Radiation Temperature at T = 0.05 ns (a) $\Delta t = 0.05$ ns (b) $\Delta t = 0.01$ ns

Figure 55 shows the fully-implicit PWLD cell-average radiation temperature solution with a smaller time step of Δt = 0.002 ns. This solution is closer to the more accurate PPWL-STLC solution, but some smearing of the solution is still evident even with Δt = 0.002 ns. Overall, the early-time temperature solution of this top-hat problem has shown that the PPWL-STLC method achieves more accurate solutions for a streaming radiative transfer problem than various time-differenced PWLD methods.

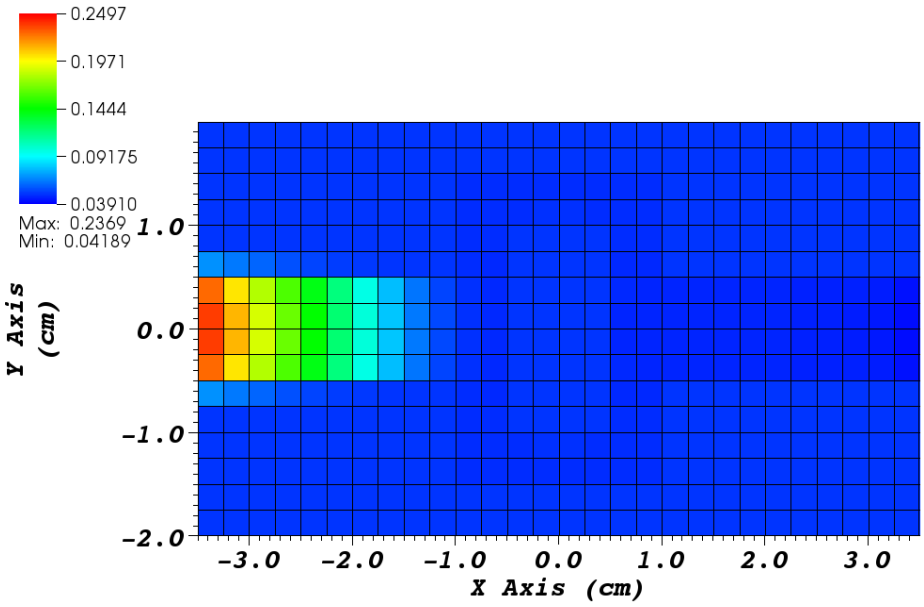


Figure 55. Fully-Implicit PWLD Cell-Average Radiation Temperature at T = 0.05 ns with Δt = 0.002 ns

Next, the solution to this top-hat problem at a long time is shown using the fully-implicit PWLD method on a refined grid of 112x64 cells. Figure 56 shows the PWLD material temperature solution at a time of t = 0.8 μs. This solution was obtained with an adaptive time-stepping procedure given a minimum time step of Δt = 0.01 ns. The

temperature solution shown in this figure gives an idea of how the radiation is affecting the material temperature after a relatively long time in the optically thick region of the problem.

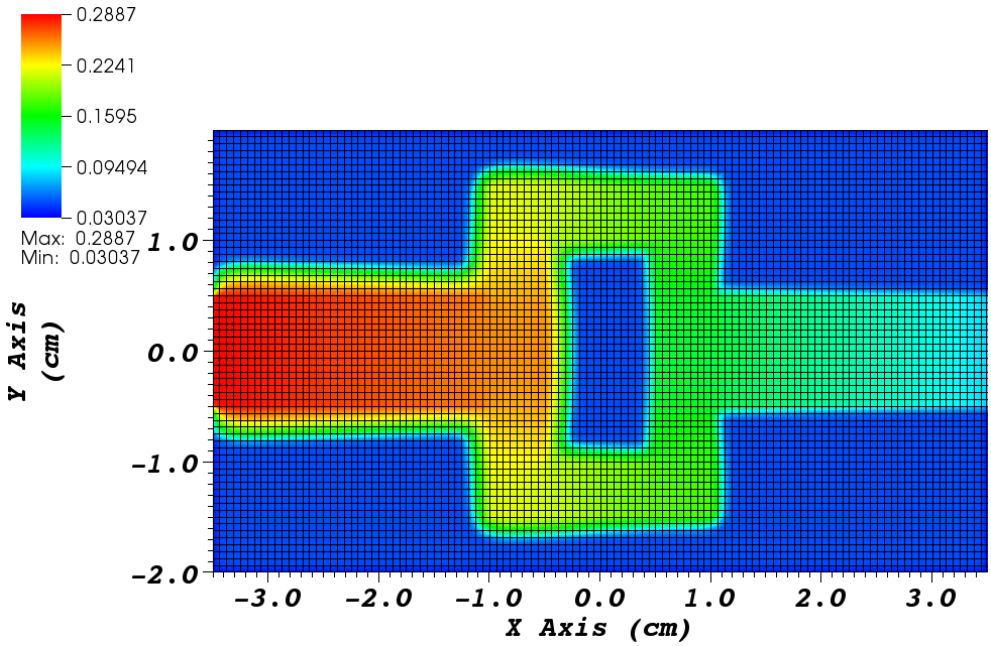


Figure 56. PWLD Material Temperature for the Top-Hat Problem at t = 0.8 μs

Figure 57 shows the PPWL-STLC material solution on the same refined grid using an adaptive time step at a time of t = 0.0257 μs. This solution is not far enough to show the spreading of the solution into the walls of the thick region but demonstrates the difficulty of solving this problem. In order to show the behavior of this PPWL-STLC method in the thick diffusion limit, a simpler problem is used as developed in the next section.

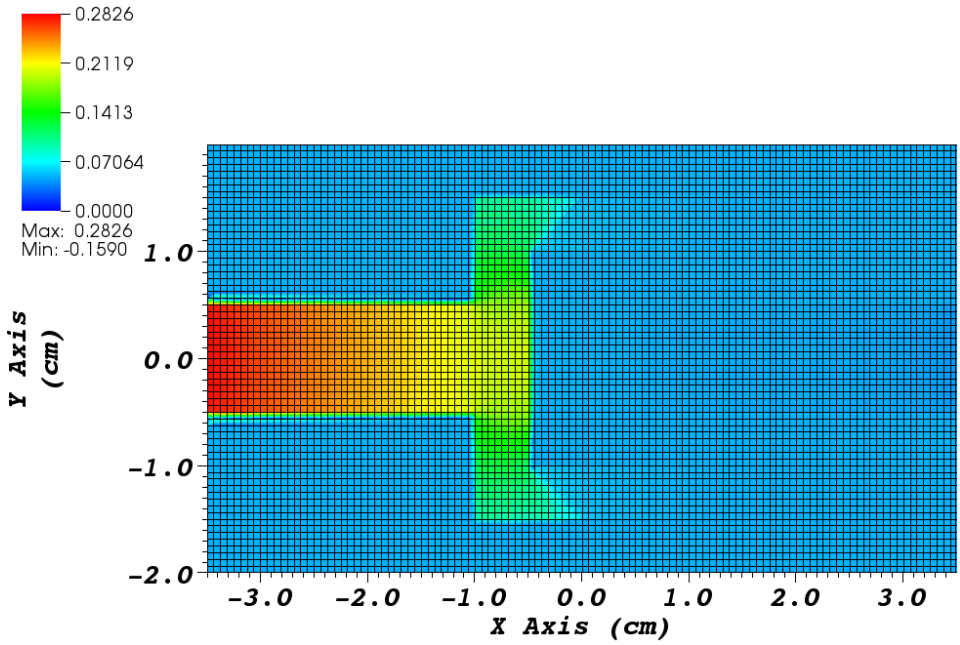


Figure 57. PPWL-STLC Material Temperature for the Top-Hat Problem at t = 0.0257 μs

6.4.5 Simple Thick Diffusion Limit Problem

This test problem seeks to show the behavior of the PWC-STLC and PPWL-STLC methods in the thick diffusion limit. It is a time-dependent problem consisting of a 16x16 non-orthogonal grid in (*x*, *y*) made up of thin material with a highly scattering (thick) block as shown in Figure 58. Table 7 gives the physical properties of the materials in this problem. A beam is incident on the left boundary of the problem for the direction $\vec{\Omega}_m$ such that μ_m is maximum and $\eta_m > 0$ for all t > 0 with a value of $\psi^{inc}_{m,left} = 1.0\,\text{n/cm}^2\text{-s-str}$. All other boundaries are vacuum and the initial condition is $\psi^{initial}_{m,g}\left(\vec{r}, t = 0\right) = 0$ n/cm^2-s-str. A Gauss-Legendre-Chebyshev product quadrature set is

used to solve this problem with 1 polar angle and 8 azimuthal angles per quadrant. Neutrons are travelling at a speed of v = 1.0 cm/μs in the problem and the STLC track spacing is $\Delta\omega = \Delta u$ = 0.015625 cm.

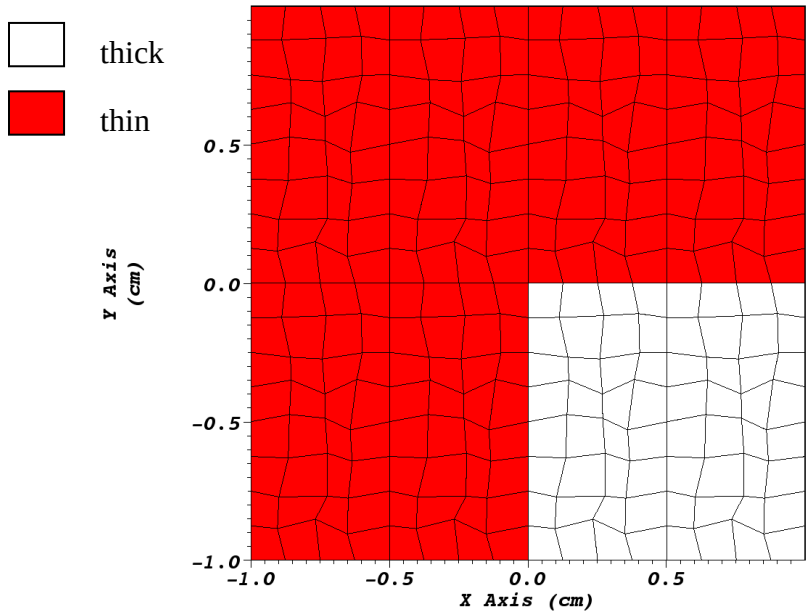


Figure 58. Thick Diffusion Limit Test Problem Materials and Grid

Table 7. Thick Diffusion Limit Test Problem Properties

Property	Thin	Thick
Σ_t [cm^{-1}]	1.0	1000
$c = \Sigma_s / \Sigma_t$	0	0.999999

GMRES is used to converge this problem to a tolerance of $r_{2,rel}$ < 1E-08 and a pointwise tolerance of $d_{2,rel}$ < 1E-03. The PWC-STLC and PWL-STLC methods solved this problem using a time step of Δt = 10 μs. These solutions are compared to the fully-

implicit PWLD solution and the PWLD method with TBDF2 time-differencing using the same time step. First, the solution of this problem in steady-state using the PWLD method is given in Figure 59 to show how this solution should behave. This figure shows the smooth solution that is attained in the thin region of the problem near the left boundary. Another interesting feature of this solution is the smooth behavior of the solution in the boundary layer on the left boundary of the thick region.

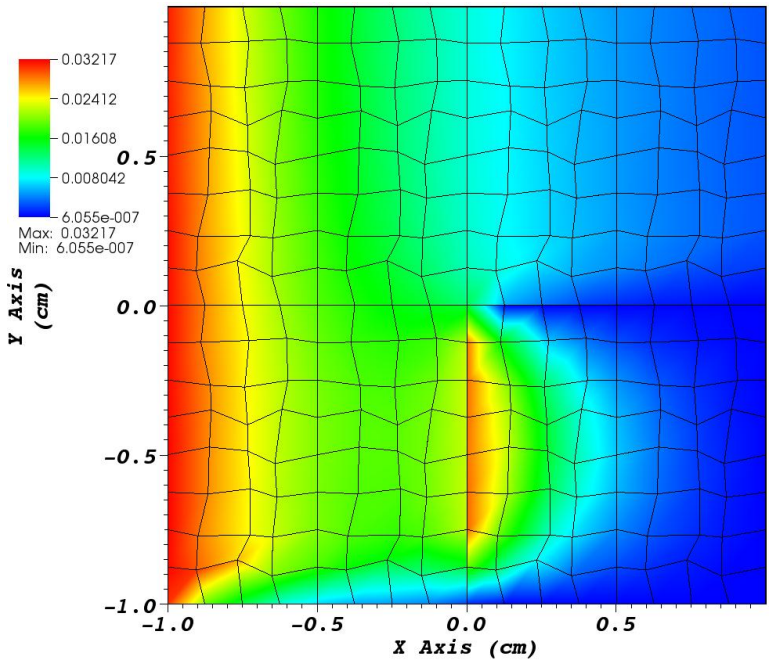


Figure 59. PWLD Steady-State Pointwise Scalar Flux for Thick Diffusion Limit Test Problem

Next, we show how the PWC-STLC method fails in the thick diffusive region in this problem. Figure 60 shows a cross section of the cell-average scalar flux solution from the PWC-STLC method at a position of y = -0.43 cm in the thick diffusive region

of the problem at various times along with the steady-state cell-average PWC-LC solution. From this figure it is clear that the time-dependent solutions after the second time step are almost the same as the steady-state solution. This is in accord with the results of the PWC-STLC analysis performed in section 4.5.1. The leading-order solution in this thick region does not know anything about the time-dependence of the problem because the solution is determined only by the incident flux on the thick diffusive region, which reaches steady-state in this problem before the end of the first time step. As a result, the solution from this PWC-STLC method is completely unphysical.

Figure 61 shows a similar plot of the lineout of the cell-average scalar flux from the PWC-STLC method compared to the cell-average scalar flux from the PPWL-STLC at a later time step and the steady-state PWL-LC solution. These plots show that the PPWL solution at a time of $t \in (50, 60)$ μs is much lower than the PWC-STLC solution is predicting, illustrating that this PWC-STLC solution is unphysical in this limit. The steady-state solutions from the PWC-LC and PWL-LC methods are fairly close only because the diffusion operator in this region is almost a Laplacian, and with the chosen grid, the leading-order PWC-LC solution is not far from the solution of a Laplacian. (See section 4 for more details.)

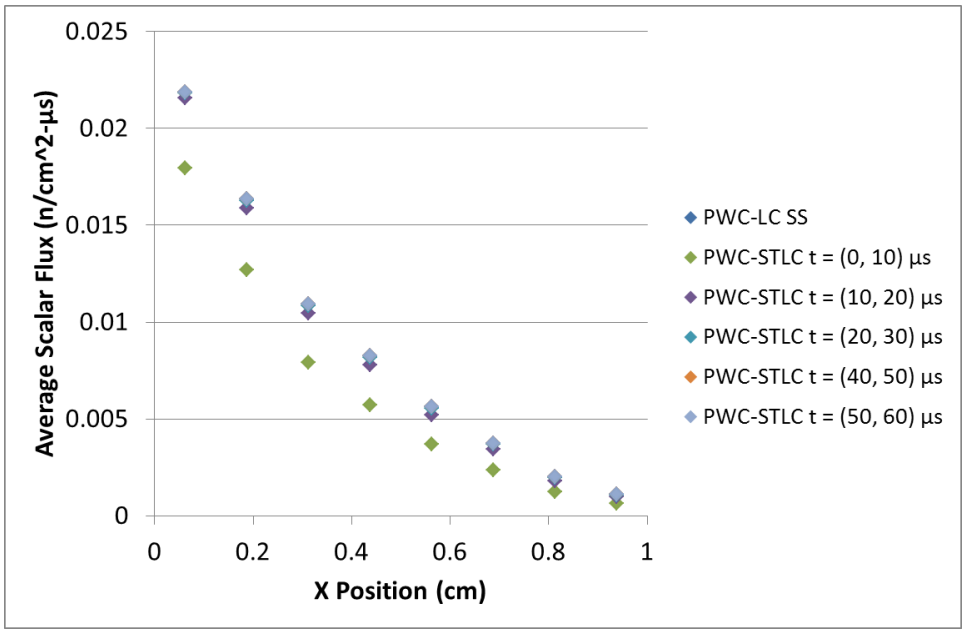


Figure 60. PWC-STLC Cell-Average Scalar Flux at y = -0.63 cm for Various Times in Thick Diffusive Region

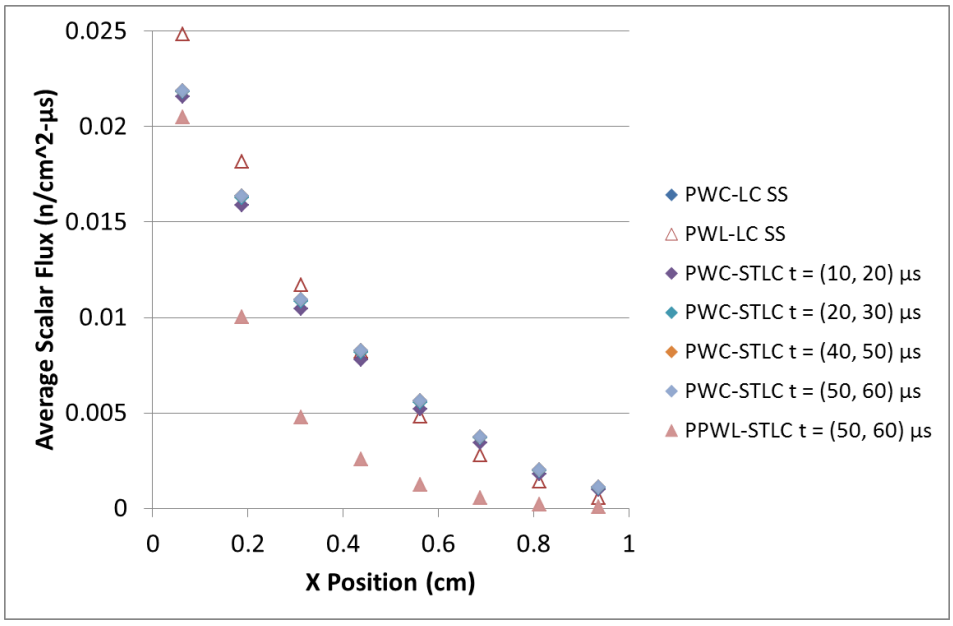


Figure 61. PWC-STLC Cell-Average Scalar Flux at y = -0.63 cm Compared to PPWL-STLC Cell-Average Scalar Flux in Thick Diffusive Region

Next, the accuracy of the PPWL-STLC method in the thick diffusion limit is examined. Figure 62 shows the pointwise scalar flux solution at a position of y = -0.43 cm from the PPWL-STLC method at various time steps along with a comparison to the steady-state PWL-LC solution along this line. It can be seen that this PPWL-STLC solution is approaching the steady-state PWL-LC solution in the thick region as expected from the analysis described in section 4.5.4.

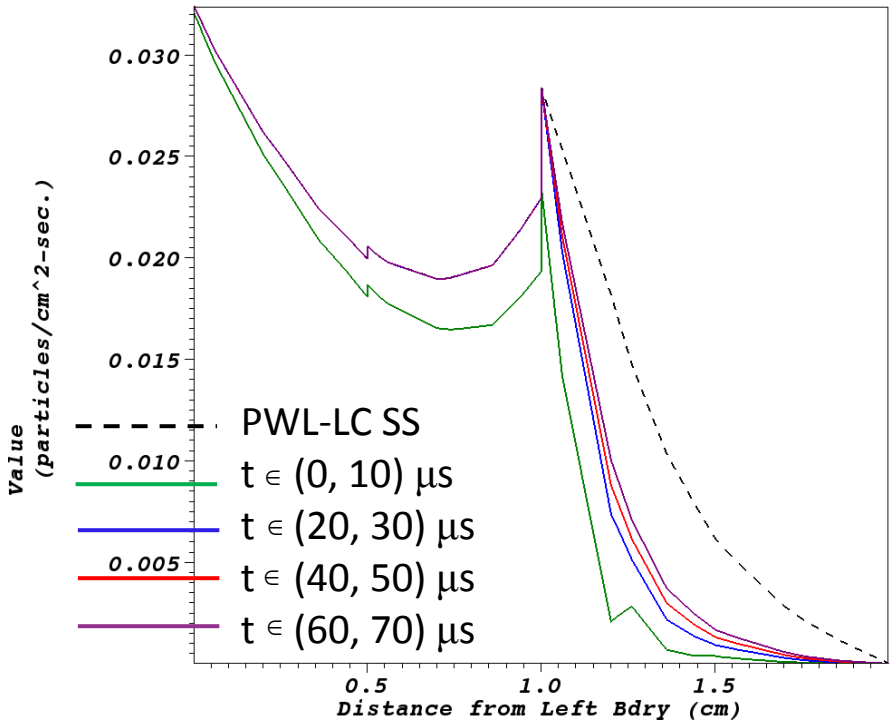


Figure 62. PPWL-STLC Pointwise Scalar Flux Solution at Various Times for Thick Diffusive Problem

The bumps seen in this solution will be addressed shortly. The behavior of this PPWL-STLC method can be compared to the PWLD method with various time discretizations by looking at the solutions plotted in Figure 63.

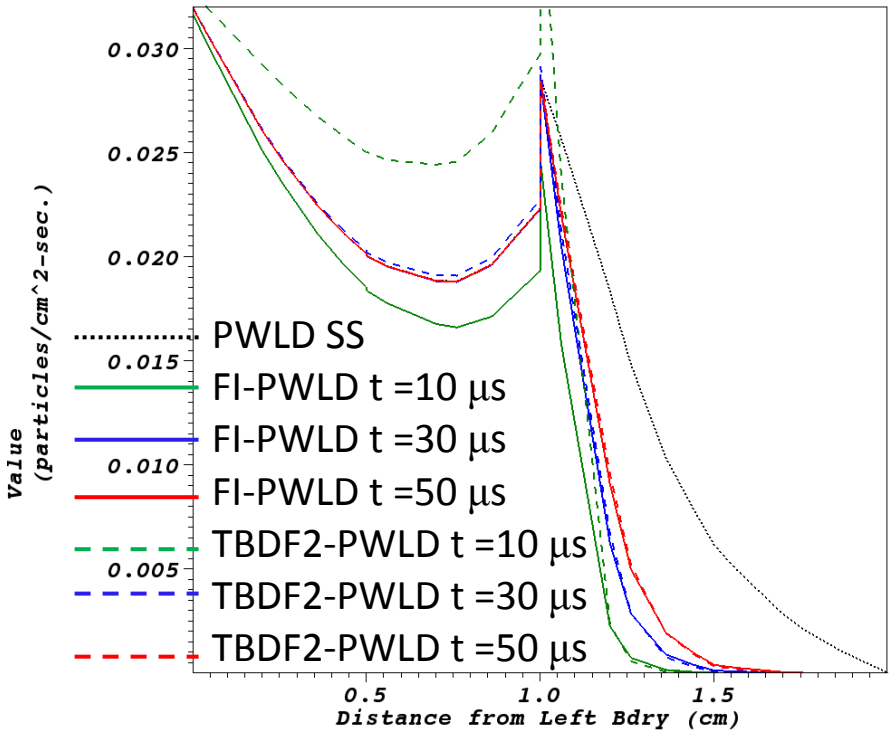


Figure 63. FI-PWLD and TBDF2-PWLD Pointwise Scalar Flux Solutions at Various Times for Thick Diffusive Problem

This figure shows the pointwise scalar flux solution at y = -0.43 cm from various PWLD solutions. First, the fully-implicit PWLD (FI-PWLD) solutions shown at various times are indeed approaching the steady-state PWLD solution as time increases. Second, the solutions from the PWLD method using TBDF2 time-discretization

(TBDF2-PWLD) becomes the same as the FI-PWLD after the first few time steps. Finally, from Figure 62 and Figure 63, it can be seen that the behavior for these PWLD methods is similar to the behavior of the PPWL-STLC method; they approach the same steady-state solution in this thick diffusive region at the same rate. The PWLD method is known to be accurate for this kind of problem. Therefore, we infer that the PPWL-STLC method is also accurately predicting the physical behavior of this system.

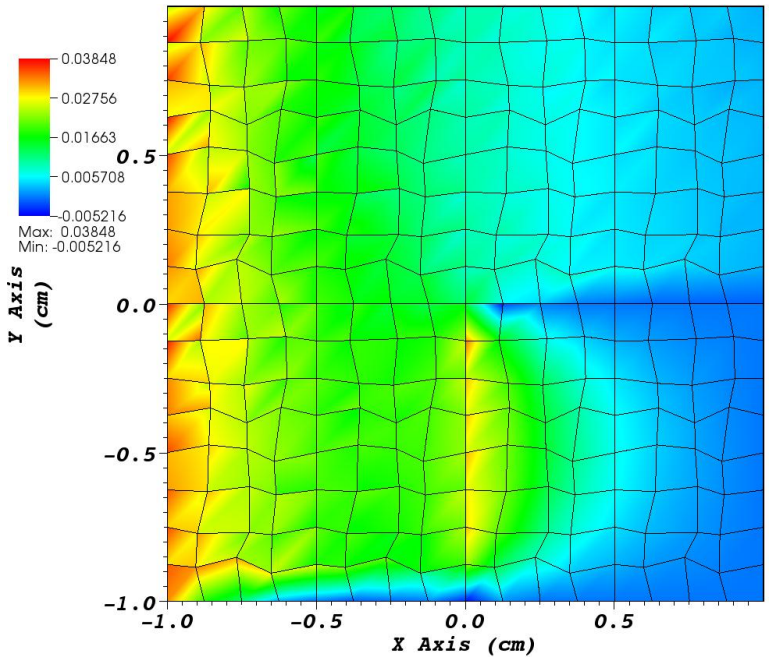
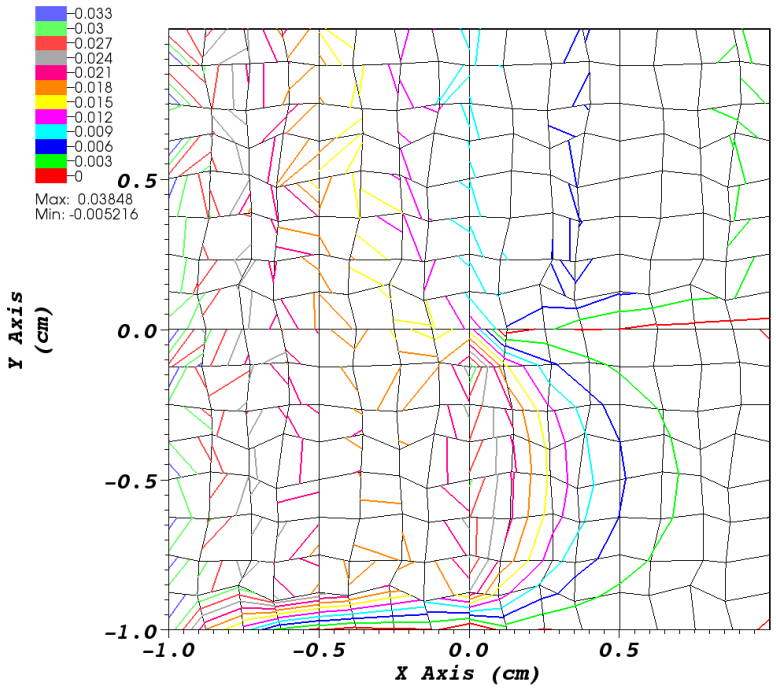


Figure 64. PWL-LC Steady-State Scalar Flux for Thick Diffusion Limit Test Problem

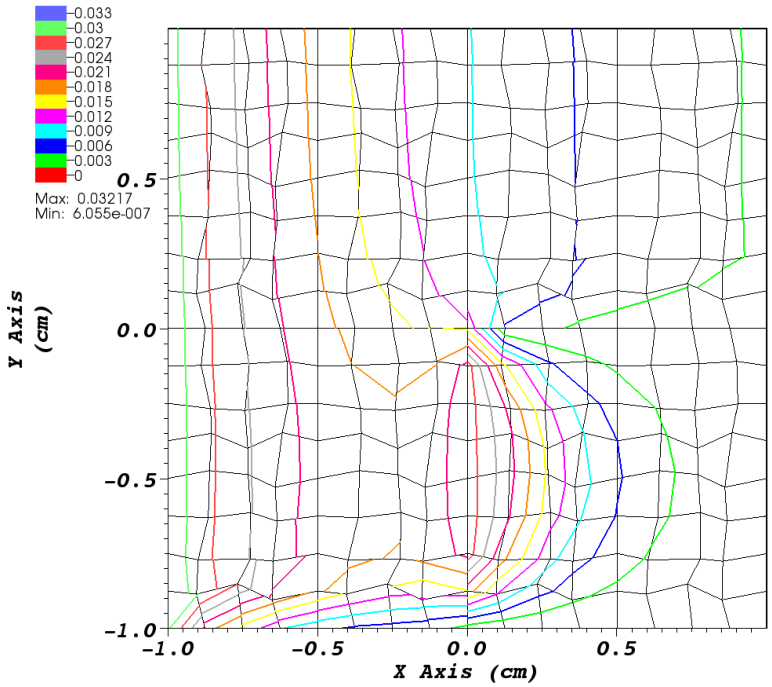
The bumps seen in Figure 62 are evidence of a flaw in the LC and STLC least-squares construction of the collision source. Figure 64 shows the steady-state pointwise scalar flux solution from the PWL-LC method using a track spacing of $\Delta\omega = 0.015625$ cm.

This solution should be smooth, similar to the PWLD solution in Figure 59 (note that these figures are not on the same scale). Contour plots of the PWL-LC and PWLD steady-state solutions are shown in Figure 65 (these figures are on the same scale). This figure clearly shows the non-smoothness of the solution that occurs with the PWL-LC method as opposed to the PWLD method in steady-state.



(a)

Figure 65. Steady-State Scalar Flux Contour Plot (a) PWL-LC (b) PWLD



(b)

Figure 65. Continued

Some of these "bumps" can be attributed to inverting the matrix in our least-squares procedure after all angles have been accumulated. For example, in this test problem, the solution on the left boundary only has a non-zero contribution for the one angle at which the incident beam is entering. However the matrix on the left hand side, $[M_{LC}]$, of our least-squares procedure, given in Eq. (5.7), includes contributions from all tracks for all angles, which means our least-squares matrix is being weighted by the track-volume for all angles in the problem. However, the right-hand side vector, $\vec{T}_{LC}$, in the matrix equation includes the track volume for only one angle (the only non-zero contribution). This mismatch causes an inaccurate fit to the solution.

The equations outlined in section 5 for this steady-state PWL-LC method are reiterated here for discussion. Similar matrices and vectors to Eqs. (5.7) and (5.8) can be defined for one angle as given in Eq. (6.18) and Eq. (6.19).

$$\left[M_{mLC}\right] \equiv \frac{1}{4\pi} \sum_{k=1}^{\#\text{ tracks}} A_{k,m} \int_{0}^{\Delta s_{k,m}} ds \vec{b}\vec{b}^{T} \tag{6.18}$$

$$\vec{T}_{mLC} \equiv \sum_{k=1}^{\#\text{ tracks}} A_{k,m} \int_{0}^{\Delta s_{k,m}} ds \vec{b} \Psi_{m,g,k}\left(s\right) \tag{6.19}$$

Using these definitions, the procedure followed in PDT to find the least-squares flux for this PWL-LC method is given in Eq. (6.20), which is the same as Eq. (5.9).

$$\vec{\phi}_{g,cell}^{SS} = \left(\sum_{m=1}^{M} w_m \left[M_{mLC}\right] \right)^{-1} \left(\sum_{m=1}^{M} w_m \vec{T}_{mLC} \right) \tag{6.20}$$

Inverting the system after all angles have been accumulated causes the matrix to be weighted by the track-volume and basis contribution for each angle. Therefore, like in this test problem for example, if only one or two angles have an angular contribution that is non-zero, this least-squares solution is influenced by the basis-weighted track-volume from all angles. This solution procedure is not quite correct because there is contamination from directions that do not contribute to the vector $\vec{T}_{mLC}$.

To remove this flaw, a different least-squares procedure can be employed in which the $[M_{mLC}]$ matrix is inverted for each angle and the contribution to the least-squares flux is accumulated for each angle. This procedure is given by Eq. (6.21).

$$\vec{\phi}_{g,cell}^{SS} = \sum_{m=1}^{M} w_m \left(\left[M_{mLC} \right]^{-1} \vec{T}_{mLC} \right) \tag{6.21}$$

This "new" PWL-LC method was implemented in PDT (along with the corresponding PPWL-STLC method) to show that it improves the solution for this test problem. A contour plot of the steady-state scalar flux using this "new" PWL-LC method is given in Figure 66. Figure 66 shows that the solution is much smoother in the thin (left) region of the problem compared to the original least-squares method shown in Figure 65(a). These new results are much closer to the PWLD results in Figure 65(b).

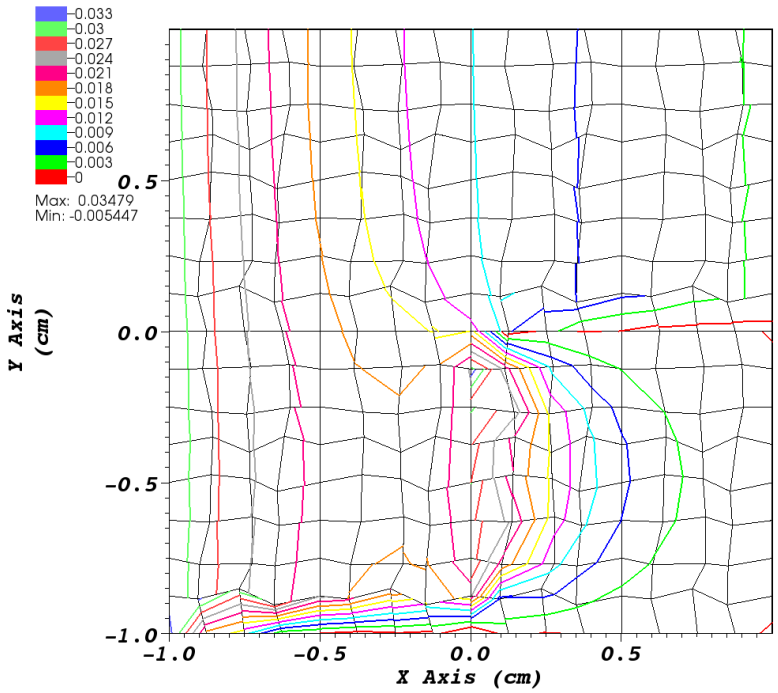


Figure 66. PWL-LC "New" Method Steady-State Discontinuous Scalar Flux Contour Plot

Both this "new" PWL-LC method and the original PWL-LC method still exhibit non-smoothness in the thick diffusive region. This non-smoothness can also be attributed to the fact that the track-based least-squares projections can be different when different directions have significantly different track-based areas on a surface. From the analysis of the usual PWL-LC method in the thick diffusion limit, it was found that the leading-order solution for a cell on the boundary of a thick diffusive region depends on the track-area basis-weighted angular solution for the incoming and exiting angles on each edge of a cell. In this test problem, the incoming angular flux on the left boundary of the thick diffusive region is non-zero only for one incident angle. The outgoing flux values on this boundary edge are non-zero due to scattering in this thick region. Since the directions in this problem have significantly different track-based areas on cell surfaces, the contribution to the least-squares scalar flux is being weighted by these different areas; therefore the true solution is being "smeared" by this different weighting. Although an analysis of the "new" PWL-LC method has not been performed in the thick diffusive limit, it is expected that the leading-order solution will satisfy the same boundary relation. This track-based volume-and-area flaw is an issue that has not been resolved in this research, as is discussed in section 8.

An estimate of the number of angular flux unknowns for this test problem using each method is given in Table 8. Note that the number of angular flux unknowns for the PPWL-STLC method is dependent on the chosen track spacing.

Table 8. Approximate Total Angular Flux Unknowns for One Time Step of the Time-Dependent Thick Diffusive Problem

Unknowns for One Time Step	FI-PWLD	TBDF2-PWLD	PPWL-STLC
Total Angular Flux Unknowns	3.277×10^4	6.554×10^4	1.206×10^9

This thick diffusive test problem has shown that the PWC-STLC method fails in the thick diffusive limit, as predicted by analysis. Also, the PPWL-STLC method exhibits the expected good performance in this thick diffusive limit, as predicted by analysis. This test problem also exposes a flaw in the least-squares procedure used for the LC and STLC methods developed in this dissertation, which led to the testing of a "new" least-squares method. The issue of how to best handle track-based least-squares projections when different directions have significantly different track-based volumes and areas is left as an unresolved issue in this dissertation.

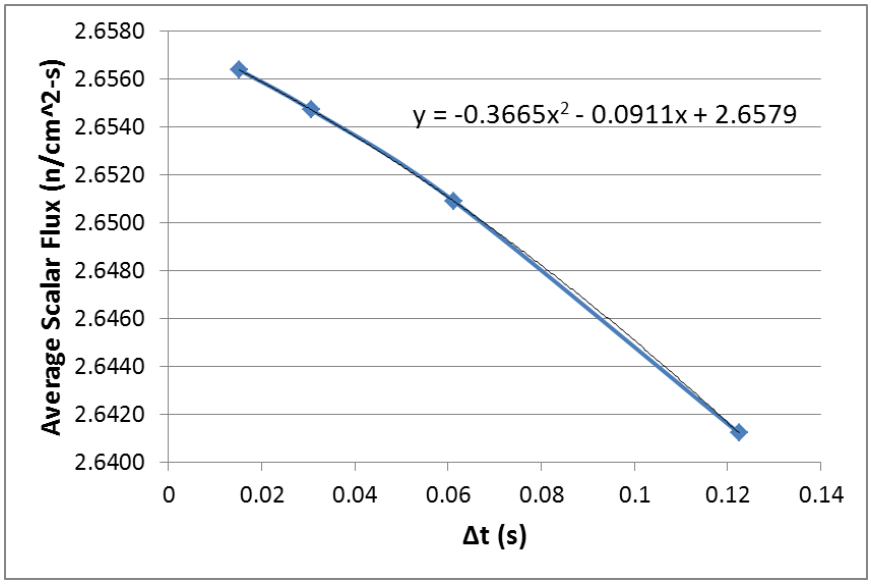
6.4.6 STLC Order of Convergence in Time

The next test problem seeks to show how PWC-STLC and PPWL-STLC methods converge as the time step is refined. It consists of a 1 cm by 1 cm problem divided into 10x10 cells of uniform size, $\Delta x = \Delta y = 0.1$ cm, with a track spacing of $\Delta\omega = \Delta u = 0.00625$ cm. A Gauss-Legendre-Chebyshev product quadrature set with 1 polar angle and 20 azimuthal angles per quadrant is used to solve this problem. The total cross section of the material in the problem is $\Sigma_t = 10.0\,\text{cm}^{-1}$ with a scattering ratio of

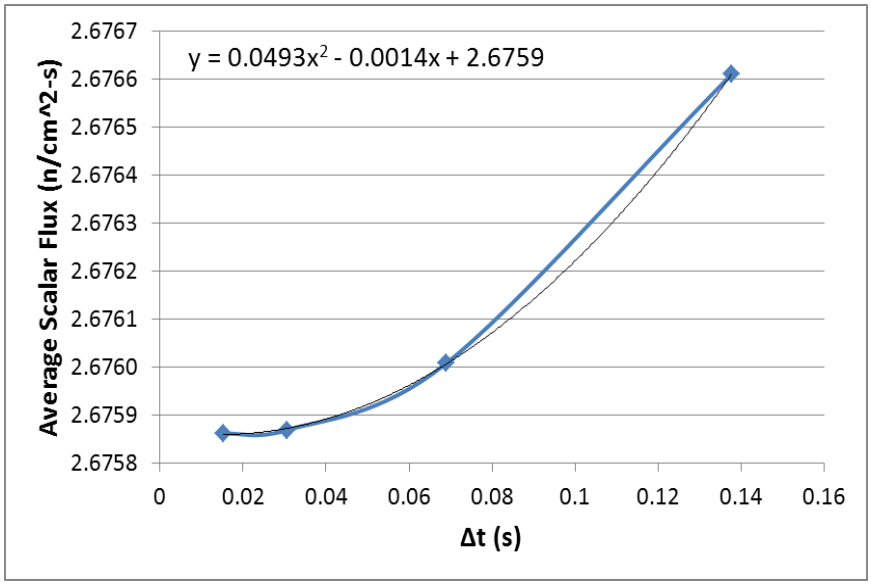
$c = \Sigma_s / \Sigma_t = 0.9$. The speed of neutrons in this time-dependent problem is v = 1.0 cm/μs. An isotropic beam is incident on the left side of the problem for t > 0 with a value of $\psi^{inc}_{m,\text{left}} = 5\,\text{n/cm}^2\text{-}\mu\text{s-str}$ with all other boundary conditions set to vacuum. The initial flux is zero. As with previous test problems, GMRES is used to converge the solution with a tolerance of $r_{2,\text{rel}}$ < 1E-11 and a relative pointwise solution tolerance of $d_{2,\text{rel}}$ < 1E-7.

To investigate the convergence of the STLC methods in time, the track spacing is kept constant as the time-step size is refined. Figure 67 shows the cell-average scalar flux from the PWC-STLC and PPWL-STLC methods plotted at a time of t = 1.1023 μs for a cell on the left boundary of the problem at a *y*-position this is a half-cell width above the midway point (referred to as the midpoint of cell 5). Figure 68 shows the total average scalar flux in the problem at a time of t = 1.1023 μs from the PWC-STLC and PPWL-STLC methods. It should be noted that that these methods will converge to different solutions, because they use different spatial treatments.

From the PPWL-STLC solutions shown in Figure 67(b) and Figure 68(b) there is evidence that this method is second order in time. It is also evident from these figures that the PWC-STLC method converges significantly more slowly than the PPWL-STLC method. For example, note that the scale on the PPWL-STLC total average flux in Figure 68(b) spans a range of 0.14 but the scale from the PWC-STLC total average flux in Figure 68(a) spans a range of 4. Since the PPWL-STLC method converges faster than the PWC-STLC method a larger time step can be used with the PPWL-STLC method compared to the PWC-STLC method to find a similarly accurate solution.

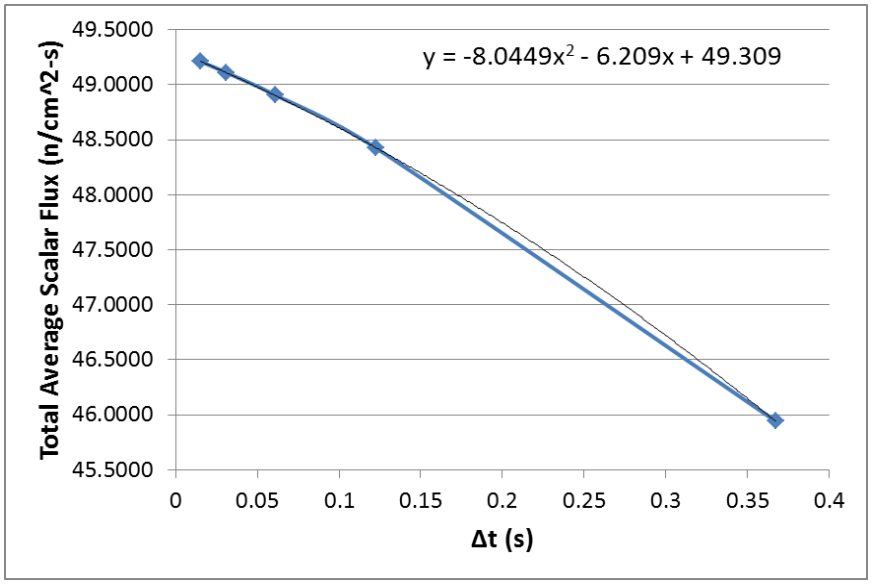


(a)

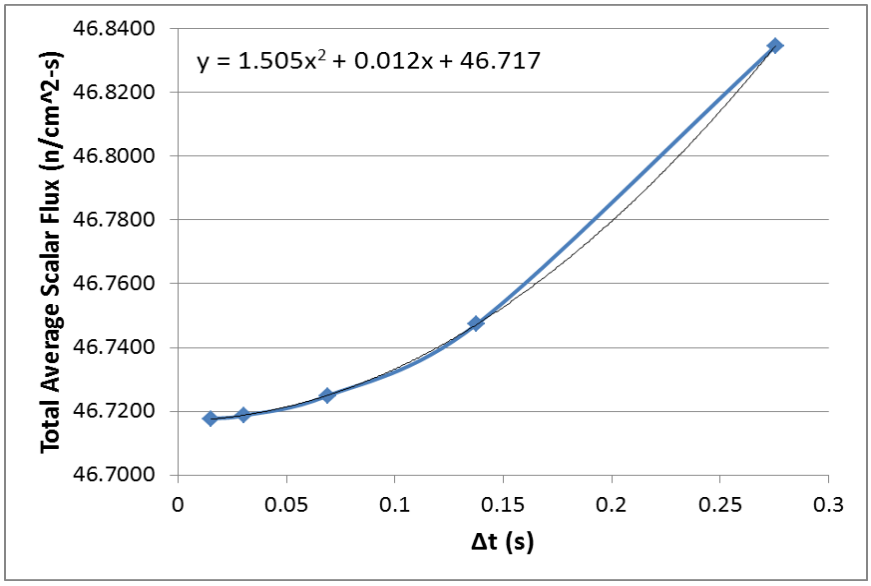


(b)

Figure 67. Cell-Average Scalar Flux at t = 1.1023 μs for Midpoint of Cell 5 (a) PWC-STLC (b) PPWL-STLC



(a)



(b)

Figure 68. Problem-Integrated Scalar Flux at t = 1.1023 μs (a) PWC-STLC (b) PPWL-STLC

This test problem presents evidence that the PPWL-STLC method is second-order in time. It has also shown that the PWC-STLC method converges more slowly than the PPWL-STLC method, but we cannot be certain with this test problem whether the PWC-STLC method is first-order or between first and second-order in time. STLC methods lack smoothness of solutions in time due to the different track layouts that are present at different time boundaries with a finite track spacing. This issue makes order-of-convergence studies in time problematic for STLC methods.

6.4.7 Track-Based Conservation

This test problem shows how the PWC-LC method satisfies our definition of track-based conservation. Recall the definition of track-based conservation given in Eq. (2.7). This conservation equation (referred to as TBC) is given in Eq. (6.22).

$$TBC \equiv \sum_{m=1}^{M} w_m \sqrt{\mu_m^2 + \eta_m^2} \sum_{k=1}^{\#\text{ tracks}} \Delta\omega_{k,m} \left\{ \begin{array}{l} \Psi_{m,k}^{out} - \Psi_{m,k}^{in} + \dfrac{\sigma_a \Delta s_{k,m}}{4\pi} \bar{\phi}_k^{SS} \\ -\dfrac{\Delta s_{k,m}}{4\pi} \bar{Q}_{k,m} \end{array} \right\} \qquad (6.22)$$

The PWC-LC method should satisfy this track-based conservation statement. A simple one-cell 2D (x, y) test problem is used to show this property. It consists of a scattering material with a total cross section of $\sigma_t = 0.3\,\text{cm}^{-1}$ and a scattering ratio of c = 0.5. An isotropic flux is incident on all boundaries, $\psi_m^{inc} = 2.0\,\text{n/cm}^2\text{-s-str}$, with a fixed isotropic source, q_{fixed} = 0.6 n/cm^3-s-str. The cell dimensions are 2 cm x 2 cm with a track spacing of $\Delta\omega$ = 0.1 cm. An S_4 level symmetric quadrature set is used and the pointwise solution tolerance is set to $d_{2,rel} < 1E\text{-}10$.

The PWC-LC method, with $\Delta\omega$ = 0.1 cm, applied to this test problem finds TBC = -1.776E-11, which is within the iteration error associated with this conservation statement with the given pointwise tolerance on the solution. The "new" PWC-LC method outlined in section 6.4.5 does not satisfy this conservation statement on this test problem, as evidenced by its TBC = -1.064E-04. With a much coarser track spacing of $\Delta\omega$ = 0.9 cm, the PWC-LC method also satisfies conservation with TBC = -9.280E-11 but the "new" PWC-LC method does not with TBC = -5.513E-03.

This test problem shows that the PWC-LC method satisfies this track-based conservation statement because the definition of the least-squares flux comes directly from this statement. Since the PWL-LC and STLC methods follow the same least-squares procedure, they also satisfy this track-based conservation statement. The "new" method developed in section 6.4.5 does not satisfy the track-based conservation statement given in Eq. (6.22) for this problem because a different least-squares method is used to find the scalar flux.

7. RECOMMENDATIONS

Here we outline several aspects of LC and STLC research that we recommend for further investigation.

One area of promising research with LC and STLC methods is track-based sweeping algorithms. Currently, PDT uses cell-based sweeps to solve transport problems in parallel[26]. The automated domain decomposition option in PDT uses algorithms developed recently to determine the optimal domain decomposition and sweep scheduling for a given problem using cell-based transport sweeps. With cell-based sweeps, the sweep problem is divided into tasks, each of which involves solving the problem for a set of cells. Track-based sweeps would execute tasks that involve solving the sweep problem for a set of tracks, which in general would not be all of the tracks within any set of cells.

Track-based sweeps have the potential to produce a more scalable parallel algorithm than cell-based transport sweeps, for two reasons. Figure 69, which shows a sample track layout on a problem domain for one angular direction using KBA partitioning, illustrated one reason: with track-based sweeping, each processor can begin doing work with no pipe-fill time because each processor owns tracks that start on the boundary. The second main reason is that long characteristics dramatically simplify the dependencies of sweep uknowns on their upstream counterparts, because the unknowns along a ray depend on only upstream unknowns on that ray. Reduction of dependencies can translate into increased parallel efficiency, and track-based sweeps have the potential to exploit this.

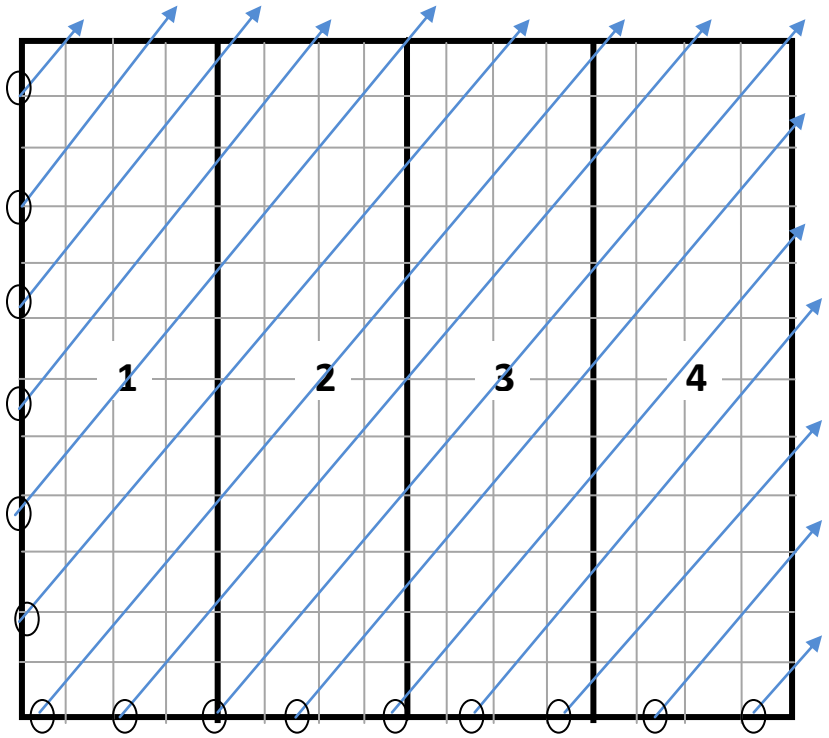


Figure 69. Track Layout for One Angular Direction with KBA Partitioning

Computational efficiency arose as a major concern with the LC and STLC methods developed in this research, at least as they were implemented in PDT. The CASMO code demonstrated that LC codes can be highly efficient[11], at least in a 2D serial code with a PWC source approximation. The achievable efficiency remains an open question in the setting of PWL approximations, massively parallel codes with domain decomposition, and 3D. Further work should explore what efficiency gains are possible in this setting. One possibility is to exploit the fact that for prismatic cells, many tracks are "stacked" and have the same track geometry in the plane of the top and

bottom of the prism. LC methods on prism-shaped cells (including STLC methods for space-time cells, which are effectively prism cells) can use the same track geometry for tracks with the same path across the *x*-*y* plane. This can be exploited to significantly improve computational efficiency. The "stacks" of tracks that occur for prism geometry allow for the geometric and exponential calculation to be performed once for a given set of stacked tracks.

A separate efficiency issue involves the track spacing that is necessary to produce an accurate solution without artificial extrema (bumpiness). As one diffusion-limit test problem illustrated, the methods developed in this research require fine enough track spacing such that the track-based volume is close to the true cell volume for essentially every direction in the quadrature set. In some problems, this apparently extends to track-based surface areas being close to true surface areas, which is challenging for directions that are nearly parallel to a surface. If a method can be found to generate accurate, smooth solutions with only the number of tracks needed to resolve angular flux variations, solution quality and computational efficiency will both increase substantially for the LC methods developed in this work.

The method and algorithms developed for the PPWL-STLC method in (x, y, t) can be easily applied to prism cells in (x, y, z). These prism cells are common in nuclear reactor models. Taking advantage of these prism cells allows for easier computations that treating them as general polyhedral and using standard 3D PWL basis functions. The PPWL basis functions could also be used by a DFEM on prismatic cells.

Another aspect of the LC and STLC methods that should be explored is the use of adaptive track spacing. Without this, if fine spacing is required to resolve the solution in one part of the domain, these rays stretch across the entire problem domain. This can increase total track count significantly. With adaptive spacing, most of the problem could use relatively coarse track spacing, with finely spaced tracks introduced only where needed.

A related issue that arose during this research concerns cells with high aspect ratios. An important example is space-time cells for which $v\Delta t >>$ spatial cell diameter. If tracks for a given direction are spaced finely enough that there are multiple tracks crossing the "top" surface of a prismatic cell (the spatial-domain surface at the end of the time step), then there are a great many parallel tracks that cross the spatial cell during the time step, at finely spaced time intervals. The solution usually does not vary much across these small time intervals; otherwise the large time step probably would not be used. However, removal of some tracks in the middle of the time step in a given cell may not be advisable, because those tracks may be needed to resolve the end-of-step flux in a different cell downstream. This issue needs to be addressed if STLC methods are to be computationally competitive. One way to address this is to make "stacked" tracks much less costly, as suggested above. Another possibility is adaptive track spacing, also suggested above. An interesting consequence of track spacing in high-aspect cells is that with the STLC methods as currently implemented, long time steps do not necessarily mean shorter run times; a time step twice as long may simply have twice as many tracks and thus cost just as much to solve as two time steps of half the length.

For the LC and STLC methods developed in this work, it was found through testing that for track-based equations involving a summation over directions, the different track-based volumes and areas associated with different quadrature directions can cause problems when the solution varies a lot from direction to direction. An alternative least-squares formulation was explored that involves inverting a matrix for each direction but the properties of this new method remain to be studied. Even with this matrix inversion for each angle, oscillations still persisted in the boundary cells of the thick diffusion region of one test problem, which are caused by this issue of different track-based surface areas for different directions. A solution to this issue should be sought in future research.

8. SUMMARY AND CONCLUSIONS

8.1 Summary

The research outlined in this work has covered the development and analysis of several new long characteristic methods that discretize the spatial and temporal variables of the general linear Boltzmann transport equation. The motivation for this research was the accuracy and potential parallel efficiency of LC methods. Past research has shown that LC methods are more accurate in streaming-dominated regions than other spatial discretization methods. A limit of interest in radiative transfer problems is the thick diffusion limit, therefore an LC method that can perform well in this limit was sought. Furthermore, LC methods offer the potential for significantly improved scaling to massively parallel computing architectures using track-based transport sweeps.

The new contributions in this research included the following developments and analyses:

- Developed a long characteristic method in space-time (STLC),
- Implemented and tested this STLC method and other LC methods in PDT,
- Performed new analyses in the thick diffusion limit,
    - First diffusion limit analysis for an LC method,
    - First diffusion limit analysis for and STLC method (several variations),
    - First diffusion limit analysis for a characteristic method whose basis-function gradients do not live in the basis function space,
    - First diffusion limit analysis for a time-dependent PWL-LC method with FD in time,

- Analyzed the PWL-LC-FD-P method in the short-time-step limit (steady-state limit),
- Developed a PPWL-LC and PPWL-STLC method for 3D cells that are polygonal prisms.

This research has extended the previously developed STLC method in (x, t) to (x, y, t) using a least-squares finite element approximation for the collision source. Along with this development, a PWL-LC method was developed in (x, y) and (x, y, z) that uses a spatially piece-wise linear collision source approximation. Both the LC and STLC methods analytically invert the streaming-plus-collision operator of the transport equation; therefore, the only approximation made in these methods is in the representation of the collision source.

The PWL-LC method in (x, y) and (x, y, z) for steady-state problems uses a least-squares spatially piece-wise linear finite element source representation and finds the analytic angular flux solution along each characteristic ray using this source representation. This method can be conservative in a track-based or cell-based manner.

If a certain finite-differencing method is used in time in which end-of-time step angular fluxes are projected onto PWL bases in each cell, then the resulting PWL-LC-FD-P method has several drawbacks. A steady-state solution will depend on the size of the time step, which is an undesirable and unphysical property for a method for solving time-dependent problems. In the limit of large time steps, the solution approaches the steady-state PWL-LC solution. However, in the limit of small time steps, the solution

approaches a steady-state PWL-DFEM solution, which is not the solution one seeks when using an LC method.

All of the new STLC methods developed in this work use a least-squares procedure to find the cell-based collision source and analytically solve for the angular flux along each characteristic ray in space-time. These new methods include PWC-STLC, PWL-C-STLC, PWL-L-STLC, PPWL-STLC, and PWL-STLC. The PWC-STLC method approximates the collisional source as a constant in each cell over each time step. The other methods approximate the source as PWL in the spatial variable in each cell but differ in their approximations of the time dependence of the source. The PWL-C-STLC method approximates the source as constant in time within in each time step. The PWL-L-STLC method adds a single linear function in time for each space-time cell. The PWL-STLC method uses standard 3D PWL functions on each (x, y, t) cell, which means linear functions on each tetrahedral subcell. The PPWL-STLC method uses PWL functions developed in this research for use on prismatic cells. In this method the (x, y, t) cell is divided into triangular prisms, with the coefficient of each spatial PWL function allowed to vary linearly in time.

Asymptotic analyses of the PWL-LC and STLC methods were performed in the thick diffusion limit in the context of the steady-state and time-dependent radiative transfer equations. This is the first analysis of a multi-dimensional LC method in this limit and the first analysis of an MOC method in which the gradients of the basis functions do not live in the basis-function space. Asymptotic analysis of the PWL-LC method in steady-state in the thick diffusion limit showed the leading-order solution

satisfied a discretized equilibrium diffusion equation similar to a steady-state FEM diffusion equation. The analysis of the PWL-LC equations with a certain finite-differencing method in time showed that the leading-order solution satisfies a reasonably discretized equilibrium diffusion equation analogous to a PWL-DFEM discretization in space with fully-implicit time differencing. This is a positive result. The PWC-LC method was shown to fail in the thick diffusion limit, as expected based upon previous analyses of methods that used PWC representation for the collisional source.

Results from the asymptotic analysis of the PWC-STLC method showed that this method fails in the thick diffusion limit: its leading-order solution depends only on boundary conditions, not on sources, cross sections, or initial conditions for a given time step, as expected from the results of the steady-state PWC-LC analysis. The PWL-C-STLC leading-order solution satisfies a discretized equilibrium diffusion equation similar to a fully-implicit PWLD discretized diffusion equation. The PWL-L-STLC leading-order solution satisfies a diffusion equation similar to a fully-implicit FEM discretized balance equation but with an added time-slope term. The addition of the time-slope terms, which is expected from the results of previous research in (x, t), can cause the solution to be less accurate than other methods since this term is determined only by the time slopes of boundary conditions and is not influenced by sources or cross sections. Finally, the PPWL-STLC solution satisfies a FEM diffusion equation in which the spatial variation is PWL and the time variation of each spatial unknown is linear within each time step and discontinuous at time-step boundaries. This new diffusion discretization appears to be more accurate than PWL discretizations with finite-

differencing in time, so this is a strong result for PPWL-STLC. It is interesting that the PPWL basis functions could be used in a DFEM, with a similarly strong result likely being the outcome in the thick diffusion limit.

One of the motivations for developing LC methods is their unsurpassed accuracy in streaming problems. Several test problems were used to demonstrate this for the LC and STLC methods developed in this research, and the results were contrasted with those from DFEM methods, which tend to be subject to unphysical smoothing of wavefronts and unphysical oscillations near discontinuities. Another motivation was to develop the first LC methods that can perform well on meshes with arbitrary polygonal and polyhedral cells, even in thick diffusive regions. This was the primary reason for choosing PWL basis functions. Several test problems demonstrated that the PWL-LC methods (including the STLC variants) obtain solutions on irregular grids that are of comparable accuracy to those on regular grids. One benefit of using PWL basis functions is that the resulting methods can produce the exact solution when that solution is linear. This was demonstrated with PWL-LC and with the various STLC methods on several test problems.

Analysis from a simple time-dependent test problem showed that the PPWL-STLC method converges rapidly as the time step is refined. This test problem's analytic solution was not smooth in time, and thus it was difficult to assess the method's truncation order. However, for integral quantities and others that were relatively smooth there was evidence of second-order behavior. This same test problem showed that the

PWC-STLC method converges much more slowly, indicating that for such problems the benefits for the linear-in-time source representation are well worth the costs.

8.2 Conclusions

This research has shown that LC methods, including STLC methods, can produce solutions that are more accurate than today's state-of-the-art DFEM's in streaming-dominated problems. These LC and STLC methods are also as accurate as DFEM's in diffusive problems. There are several unresolved issues with these LC methods concerning the apparent need for fine track spacing and the associated computational cost for these tracks. These issues need further study, which afterward, may reveal whether the PWL-LC and PPWL-STLC methods are more attractive than similar DFEM's for radiative transfer problems and other particle-transport problems of practical interest.

REFERENCES

1. T. A. WAREING, J. E. MOREL, J. M. MCGHEE, "A Diffusion Synthetic Acceleration Method for the Sn Equations with Discontinuous Finite Element Space and Time Differencing," *Proc. of Int. Conf. on Mathematics and Computation, Reactor Physics and Environmental Analyses in Nuclear Applications*, Madrid, Spain (1999).
2. T. M. PANDYA and M. L. ADAMS, "Method of Long Characteristics Applied in Space and Time," *Proc. Int. Conf. on Mathematics, Computational Methods & Reactor Physics*, Saratoga Springs, New York, May 3-7, 2009, American Nuclear Society (2009) (CD-ROM).
3. T. M. PANDYA. "Long Characteristic Method in Space and Time for Transport Problems," Master Thesis, Texas A&M University, College Station (2009).
4. J. B. TAYLOR and A. J. BARATTA, "A Time-Dependent Method Of Characteristics For 3d Nuclear Reactor Kinetics Applications," *Proc. Int. Conf. on Mathematics, Computational Methods & Reactor Physics*, Saratoga Springs, New York, May 3-7, 2009, American Nuclear Society (2009) (CD-ROM).
5. J. R. ASKEW, "A Characteristics Formulation of the Neutron Transport Equation in Complicated Geometries," AAEW-M 1108, United Kingdom Atomic Energy (1972).
6. M. J. HALSALL, "A Characteristic Solution to the Neutron Transport Equation in Complicated Geometries," AEEW-R 1291, UKAEA Atomic Energy Establishment, Winfrith (1980).
7. S. G. HONG and N. Z. CHO, "CRX: A Code for Rectangular and Hexagonal Lattices Based on the Method of Characteristics," *Ann. Nucl. Energy*, **25**, 547 (1998).
8. M. R. ZIKA and M. L. ADAMS, "Transport Synthetic Acceleration for Long-Characteristics Assembly-Level Transport Problems," *Nucl. Sci. Eng.*, **134**, 135 (2000).

9. S. LOUBIÉRE, R. SANCHEZ, M. COSTE, A. HÉBERT, Z. STANKOVSKI, C. VAN DER GUTH, I. ZMIJAREVIC, "APOLLO2 Twelve Years Later," *Proc. Int. Conf. Mathematical and Computational, Reactor Physics and Environmental Analysis in Nuclear Applications*, Madrid, Spain (1999).
10. K. S. SMITH and J. D. RHODES, "CASMO-4 Characteristics Method for Two-Dimensional PWR and BWR Core Calculations," *Trans. Am. Nucl. Soc*., **83**, 294 (2000).
11. K.S. SMITH and J. D. RHODES, "Full-Core, 2-D, LWR Core Calculations with CASMO-4E," *Proc. Int. Conf. on the New Frontiers of Nuclear Technology: Reactor Physics, Safety and High-Performance Computing*, Seoul, Korea, October 7-10 (2002).
12. M. DAHMANI and R. ROY, "Parallel Solver Based on the Three-Dimensional Characteristics Method: Design and Performance Analysis," *Nucl. Sci. Eng.*, **150**, 155 (2005).
13. M. A. SMITH, D. KAUSHIK, A. WOLLABER, W. S. YANG, B. SMITH, C. RABITI, G. PALMIOTTI, "Recent Research Progress on UNIC at Argonne National Laboratory," *Proc. Int. Conf. on Mathematics, Computational Methods & Reactor Physics*, Saratoga Springs, New York, May 3-7, 2009, American Nuclear Society (2009) (CD-ROM).
14. M. A. SMITH, A. MARIN-LAFLECHE, W. S. YANG, D. KAUSHIK, A. SIEGEL, "Method of Characteristics Development Targeting the High Performance BlueGene/P Computer at Argonne National Laboratory," *Proc. Int. Conf. on Mathematics and Computational Methods Applied to Nucl. Sci. and Engineering*, Rio de Janeiro, Brazil, May 8-12, 2011, American Nuclear Society (2011) (CD-ROM).
15. E. E. LEWIS and W. F. MILLER, Computational Methods of Neutron Transport, American Nuclear Society, La Grange Park, IL (1993).

16. C. RABITI, G. PALMIOTTI, W. S. YANG, M. A. SMITH, D. KAUSHIK, A. B. WOLLABER, "Quasi Linear Representation Of The Isotropic Scattering Source For The Method Of Characteristics," *Proc. Int. Conf. on Mathematics, Computational Methods & Reactor Physics*, Saratoga Springs, New York, May 3-7, 2009, American Nuclear Society (2009) (CD-ROM).
17. E. MASIELLO, R. CLEMENTE, S. SANTANDREA, "High-Order Method Of Characteristics For 2-D Unstructured Meshes," *Proc. Int. Conf. on Mathematics, Computational Methods & Reactor Physics*, Saratoga Springs, New York, May 3-7, 2009, American Nuclear Society (2009) (CD-ROM).
18. R. SANCHEZ and S. SANTANDREA, "Convergence Analysis For The Method Of Characteristics In Unstructured Meshes," *Proc. Int. Conf. on Mathematics, Computational Methods & Reactor Physics*, Saratoga Springs, New York, May 3-7, 2009, American Nuclear Society (2009) (CD-ROM).
19. S. SANTANDREA, R. SANCHEZ and P. MOSCA, "A Linear Surface Characteristics Approximation for Neutron Transport in Unstructured Meshes," *Nucl. Sci. Eng.*, **160**, 23 (2008).
20. X. M. CHAI and K. WANG, "The Linear Source Approximation In Three Dimension Characteristics Method," *Proc. Int. Conf. on Mathematics, Computational Methods & Reactor Physics*, Saratoga Springs, New York, May 3-7, 2009, American Nuclear Society (2009) (CD-ROM).
21. M. L. ADAMS, T. A. WAREING, W. F. WALTERS, "Characteristic Methods in Thick Diffusive Problems," *Nucl. Sci. Eng.*, **130**, 18 (1998).

22. T. S. BAILEY, M. L. ADAMS, B. YANG, and M. R. ZIKA, "A Piecewise Linear Finite Element Discretization of the Diffusion Equation for Arbitrary Polyhedral Grids," *J. Comp. Physics*, **227**, 3738 (2008).

23. T. S. BAILEY, "The Piecewise Linear Discontinuous Finite Element Method Applied to the RZ and XYZ Transport Equations," PhD Thesis, Texas A&M University, College Station (2008).

24. T. S. BAILEY, J. S. WARSA, J. H. CHANG, M. L. ADAMS, "A Piecewise Bi-Linear Discontinuous Finite Element Spatial Discretization of the Sn Transport Equation," *Proc. Int. Conf. on Mathematics and Computational Methods Applied to Nucl. Sci. and Engineering*, Rio de Janeiro, Brazil, May 8-12, 2011, American Nuclear Society (2011) (CD-ROM).

25. R. S. BAKER and K. R. KOCH, "An S_n Algorithm for the Massively Parallel CM-200 Computer," *Nucl. Sci. Eng.* **128**, 312 (1998).

26. W. D. HAWKINS, T. SMITH, M. P. ADAMS, L. RAUCHWERGER, N. AMATO, M. L. ADAMS, "Efficient Massively Parallel Transport Sweeps," *Trans. Amer. Nucl. Soc*., To appear in **107** (2012).

27. M. ADAMS, N. AMATO, P. NELSON, and L. RAUCHWERGER, "Efficient Massively-Parallel Implementation of Modern Deterministic Transport Calculations," Report to the Department of Energy, Texas A&M University, College Station (2002).

28. M. MATHIS, N. AMATO, M. L. ADAMS, "A General Performance Model for Parallel Sweeps on Orthogonal Grids for Particle Transport Calculations", *Proc. Int. Conf. on Supercomputing*, Santa Fe, New Mexico (2000).

29. S. PAUTZ, M. ADAMS, T. PANDYA, "Scalable Parallel Prefix Solvers for Discrete Ordinates Transport," *Proc. Int. Conf. on Mathematics, Computational Methods & Reactor*

Physics, Saratoga Springs, New York, May 3-7, 2009, American Nuclear Society (2009) (CD-ROM).

30. S. PAUTZ, M. ADAMS, T. PANDYA, "Scalable Parallel Prefix Solvers for Discrete Ordinates Transport in Multidimensions," *Nucl. Sci. Eng.* **169**, 245 (2011).
31. T. S. BAILEY and R. D. FALGOUT, "Analysis of Massively-Parallel Discrete-Ordinates Transport Sweep Algorithms with Collisions," *Proc. Int. Conf. on Mathematics, Computational Methods & Reactor Physics*, Saratoga Springs, New York, May 3-7, 2009, American Nuclear Society (2009) (CD-ROM).
32. H. G. STONE and M. L. ADAMS, "New Spatial Discretization Methods for Transport on Unstructured Grids," *Proc. Conf. Mathematics and Computation, Supercomputing, Reactor Physics and Nuclear and Biological Applications*, Avignon, France, September 11-15, 2005, American Nuclear Society (2005) (CD-ROM).
33. G. G. DAVIDSON and T. S. PALMER, "Finite Element Transport Using Wachspress Rational Basis Functions on Quadrilaterals in Diffusive Regions," *Nucl. Sci. Eng.* **159**, 242 (2008).
34. J. S. WARSA, "A Continuous Finite Element-Based, Discontinuous Finite Element Method for S_N Transport," *Nucl. Sci. Eng.* **160**, 385 (2008).
35. H. G. STONE and M. L. ADAMS, "A Piecewise Linear Finite Element Basis with Application to Particle Transport," *Proc. Int. Conf. on Nuclear Mathematical and Computational Sciences*, Gatlinburg, Tennessee, April 6-10, 2003, American Nuclear Society (2003) (CD-ROM).
36. E. W. LARSEN, G. C. POMRANING, and V. C. BADHAM, "Asymptotic analysis of radiative transfer problems," *J. Quant. Spectrosc. Radiat. Transfer* **29**, 285 (1983).

37. M. L. ADAMS and P. F. NOWAK, "Asymptotic Analysis of Computational Method for Time- and Frequency-Dependent Radiative Transfer," *J. of Comp. Phys.* **146**, 366 (1998).
38. G. C. POMRANING, "A Variational Treatment of the Mixed Boundary Condition for the Equilibrium Diffusion Equation," *J. Quant. Spectrosc. Radiat. Transfer* **36**, 471 (1986).
39. J. E. MOREL, T. A. WAREING, K. SMITH, "A Linear-Discontinuous Spatial Differencing Scheme for *Sn* Radiative Transfer Calculations," *J. of Comp. Phys*. **128**, 445 (1996).
40. M. L. ADAMS, "Discontinuous Finite Element Transport Solutions in Thick Diffusive Problems," *Nucl. Sci. Eng.*, **137**, 298 (2001).
41. H. A. VAN DER VORST, Iterative Krylov Methods for Large Linear Systems, Cambridge University Press, Cambridge, UK (2003).
42. J. H. CHANG, "Efficient Algorithms for Discrete-Ordinates Transport Iterations in Massively Parallel Computers," PhD Thesis, Texas A&M University, College Station (2004).
43. M. L. ADAMS and E. W. LARSEN, "Fast Iterative Methods for Discrete-Ordinates Particle Transport Calculations," *Prog. Nucl. Energy*, **40**, 3 (2002).
44. E. W. LARSEN, J. E. MOREL, W. F. MILLER JR., "Asymptotic Solutions of Numerical Transport Problems in Optically Thick, Diffusive Regimes," *J. Comp. Phys*., **69**, 238 (1987).
45. E. W. LARSEN, J. E. MOREL, "Asymptotic Solutions of Numerical Transport Problems in Optically Thick, Diffusive Regimes II," *J. Comp. Phys*., **83**, 212 (1989).
46. E. E. LEWIS, M. A. SMITH, N. TSOULFANIDIS, G. PALMIOTTI, T. A. TAIWO, R. N. BLOMQUIST, "Benchmark Specification for Deterministic 2-D/3-D MOX Fuel Assembly Transport Calculations without Spatial Homogenization (C5G7 MOX Benchmark)," NEA/NSC/DOC(2001)4, (2001).

47. F. GRAZIANI and J. LEBLANC, “The Crooked Pipe Test Problem,” Lawrence Livermore National Laboratory Report UCRL-MI-143393 (2000).

48. T. M. PANDYA, M. L. ADAMS, and W. D. HAWKINS, “Long Characteristics with Piecewise Linear Sources Designed for Unstructured Grids,” *Proc. Int. Conf. on Mathematics and Computational Methods Applied to Nuclear Science and Engineering*, Rio de Janeiro, Brazil, May 8-12, 2011, American Nuclear Society (2011) (CD-ROM).

APPENDIX A

This appendix gives the details of the asymptotic analysis of the PWL-LC and STLC methods in the thick diffusion limit. Refer to section 4 of this dissertation for an overview of the main results from the detailed analyses given below.

A.1 PWL-LC Steady-State Asymptotic Analysis in the Diffusion Limit

The analysis of the PWL-LC method in steady-state shown here is similar to the time-dependent analysis given in section 4.3. First, discrete-ordinates angular discretization and multigroup discretization of frequency are applied to the steady-state radiative transfer equations given in Eq. (A.1) and Eq. (A.2).

$$\vec{\Omega}\bullet\vec{\nabla}\psi\left(\vec{r},\vec{\Omega},\nu\right)+\sigma_a\left(\vec{r},\nu,T\left(\vec{r}\right)\right)\psi\left(\vec{r},\vec{\Omega},\nu\right)=\frac{1}{4\pi}\sigma_a\left(\vec{r},\nu,T\left(\vec{r}\right)\right)B\left(\nu,T\left(\vec{r}\right)\right) \tag{A.1}$$

$$\int_0^\infty d\nu\int_{4\pi}d\Omega\sigma_a\left(\vec{r},\nu,T\left(\vec{r}\right)\right)\left[\psi\left(\vec{r},\vec{\Omega},\nu\right)-\frac{1}{4\pi}B\left(\nu,T\left(\vec{r}\right)\right)\right]+Q\left(\vec{r}\right)=0 \tag{A.2}$$

This discretization is shown in Eqs. (A.3) and (A.4).

$$\vec{\Omega}_m\bullet\vec{\nabla}\psi_{m,g}\left(\vec{r}\right)+\sigma_{a,g}\left(\vec{r},T\left(\vec{r}\right)\right)\psi_{m,g}\left(\vec{r}\right)=\sigma_{a,g}\left(\vec{r},T\left(\vec{r}\right)\right)\frac{1}{4\pi}B_g\left(T\left(\vec{r}\right)\right) \tag{A.3}$$

$$\sum_{g=1}^{G}\sum_{m=1}^{M}w_m\sigma_{a,g}\left(\vec{r},T\left(\vec{r}\right)\right)\left[\psi_{m,g}\left(\vec{r}\right)-\frac{1}{4\pi}B_g\left(T\left(\vec{r}\right)\right)\right]+Q\left(\vec{r}\right)=0 \tag{A.4}$$

Next, these radiative transfer equations are scaled by the same small parameter as in the previous section to give the diffusion limit as given in Eqs. (A.5) and (A.6).

$$\vec{\Omega}_m\bullet\vec{\nabla}\psi_{m,g}\left(\vec{r}\right)+\frac{\sigma_{a,g}\left(\vec{r},T\left(\vec{r}\right)\right)}{\varepsilon}\psi_{m,g}\left(\vec{r}\right)=\frac{\sigma_{a,g}\left(\vec{r},T\left(\vec{r}\right)\right)}{\varepsilon}\frac{1}{4\pi}B_g\left(T\left(\vec{r}\right)\right) \tag{A.5}$$

$$\sum_{g=1}^{G}\sum_{m=1}^{M} w_m \frac{\sigma_{a,g}\left(\vec{r},T\left(\vec{r}\right)\right)}{\varepsilon}\left[\psi_{m,g}\left(\vec{r}\right)-\frac{1}{4\pi}B_g\left(T\left(\vec{r}\right)\right)\right]+\varepsilon Q\left(\vec{r}\right)=0 \qquad \text{(A.6)}$$

Finally, the PWL-LC discretization is applied to these scaled equations. Equation (A.7) gives the PWL-LC discretized transport equation for one track, *k*. Similar to the time-dependent material energy equation of Eq. (4.37), the PWL-LC discretized and scaled material energy equation is given in Eq. (A.8).

$$\frac{\partial\psi_{m,g,k}}{\partial s}+\frac{\sigma_{a,g}\left(T\left(s\right)\right)}{\varepsilon}\psi_{m,g,k}=\frac{\sigma_{a,g}\left(T\left(s\right)\right)}{\varepsilon}\frac{1}{4\pi}B_{g,k}\left(s\right) \qquad \text{(A.7)}$$

$$\sum_{g=1}^{G}\frac{\sigma_{a,g}}{\varepsilon}\left[\phi_{cell,g,j}-B_{cell,g,j}\right]+\varepsilon Q_j=0 \qquad \text{(A.8)}$$

The temperature, Planckian, and scalar intensity for this PWL-LC method can be expressed using the PWL basis. The unknown coefficients in these equations are found by the least-squares fit in each cell as outlined previously. The coefficients of the scalar intensity, for example, are found by the set of equations given in Eq. (4.38). In order to do this analysis in general, so it can be applied in (*x*, *y*) and (*x*, *y*, *z*), the Jacobian that arose from different coordinate systems in section 2 is denoted by J_m and the area associated with a track, which was defined by track spacing ($\Delta\omega_{k,m}$ and $\Delta u_{k,m}$) in the previous sections, is denoted by $A_{k,m}$. Both of these terms are present in the least-squares fit for a cell.

The PWL-LC equation for a cell is found by summing over all tracks in a cell and all angles as in the least-squares procedure. This summation leads to the set of equations (for $i = 1 \ldots N$) given in Eq. (A.9).

$$\sum_{m=1}^{M} w_m J_m \sum_{k=1}^{\#tracks} A_{k,m} \int_0^{\Delta s_{k,m}} dsb_i(s) \left[\begin{array}{l} \dfrac{\partial \psi_{m,g,k}}{\partial s} + \dfrac{\sigma_{a,g}\left(T(s)\right)}{\varepsilon} \psi_{m,g.k} \\ -\dfrac{\sigma_{a,g}\left(T(s)\right)}{\varepsilon} \dfrac{1}{4\pi} B_{g,k}(s) \end{array} \right] = 0 \qquad \text{(A.9)}$$

A symmetric quadrature set is assumed for this analysis with the same properties assumed in section 4.3. Also the same notation is used as given in Eq. (4.42).

To begin this steady-state asymptotic analysis, the unknowns are postulated to have the expansions given in Eq. (4.43) and Eq. (4.44). Similar expansions are also applied to temperature-dependent quantities including $\sigma_{a,g}$ and $B_{g,k}$. The expansions are then substituted into the scaled and discretized radiative transfer equations. First, from the $O(1/\varepsilon)$ terms of Eq. (A.7) and Eq. (A.8), it is found that the leading-order angular intensity is isotropic and Planckian and that the least-squares PWL scalar intensity equals the PWL Planckian.

$$\psi^{(0)}_{m,g,k}(s) = \frac{1}{4\pi} B^{(0)}_{g,k}(s) \qquad \text{(A.10)}$$

$$\phi^{(0)}_{cell,g,j} = B^{(0)}_{cell,g,j} \qquad \text{(A.11)}$$

Next, summing Eq. (A.9) over all groups, the $O(\varepsilon)$ terms yield the following expression.

$$\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=1}^{\#tracks} A_{k,m} \int_0^{\Delta s_{k,m}} dsb_i(s) \left[\begin{array}{l} \dfrac{\partial \psi^{(1)}_{m,g,k}}{\partial s} + \\ \left[\sigma_{a,g} \left(\psi_{m,g,k} - \dfrac{1}{4\pi} B_{g,k} \right) \right]^{(2)} \end{array} \right] = 0 \qquad \text{(A.12)}$$

The $O(\varepsilon)$ terms from Eq. (A.8) are the following:

$$\sum_{g=1}^{G}\left[\sigma_{a,g}\left(\phi_{cell,g,j}-B_{cell,g,j}\right)\right]^{(2)}+Q_j=0\,. \tag{A.13}$$

The terms in Eq. (A.13) can be used to substitute for terms in Eq. (A.12), for the same reasoning given by Eq. (4.50). The first-order derivative left in Eq. (A.12) can be analyzed as was done in Eqs. (4.52)-(4.57). With this result, Eq. (A.12) becomes the following i^{th} equation for the cell.

$$\begin{aligned}&\sum_{g=1}^{G}\sum_{edge\in v_i}\int_{\partial A_e} dA_e b_{i,e} J_e^{(1)}-\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=1}^{\#tracks} A_{k,m}\int_0^{\Delta s_{k,m}} ds\bar{\nabla} b_i \bullet \bar{\Omega}_m \psi_{m,g,k}^{(1)}(s)-\\ &\sum_{g=1}^{G}\frac{1}{4\pi}\sum_{m=1}^{M} w_m J_m \sum_{k=1}^{\#tracks} A_{k,m}\int_0^{\Delta s_{k,m}} ds b_i(s) Q_{PWL}=0\end{aligned} \tag{A.14}$$

This equation is now summed for all cells that touch the vertex, v_i, at which b_i is centered. During this summation the edge net current will cancel due to the shared edges between these cells. Therefore this summation yields the following expression, which will be returned to shortly with a definition for the first-order current term that appears.

$$\sum_{cells\in v_i}\left[\begin{aligned}&-\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=1}^{\#tracks} A_{k,m}\int_0^{\Delta s_{k,m}} ds\bar{\nabla} b_i \bullet \bar{\Omega}_m \psi_{m,g,k}^{(1)}(s)-\\ &\sum_{g=1}^{G}\frac{1}{4\pi}\sum_{m=1}^{M} w_m J_m \sum_{k=1}^{\#tracks} A_{k,m}\int_0^{\Delta s_{k,m}} ds b_i(s) Q_{PWL}\end{aligned}\right]=0 \tag{A.15}$$

Next, the $O(1)$ terms from Eq. (A.9) are analyzed. Summing this equation over all groups, the $O(1)$ terms yield the following equation.

$$\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=1}^{\#tracks} A_{k,m} \int_0^{\Delta s_{k,m}} ds b_i(s) \left[\frac{\partial \psi_{m,g,k}^{(0)}}{\partial s} + \left[\sigma_{a,g} \left(\psi_{m,g,k} - \frac{1}{4\pi} B_{g,k} \right) \right]^{(1)} \right] = 0 \qquad \text{(A.16)}$$

Based upon the $O(1)$ terms from the material energy equation given in Eq. (A.8), the second term in Eq. (A.16) is zero leaving the following $O(1)$ terms:

$$\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=1}^{\#tracks} A_{k,m} \int_0^{\Delta s_{k,m}} ds b_i(s) \frac{\partial \psi_{m,g,k}^{(0)}}{\partial s} = 0 . \qquad \text{(A.17)}$$

Using relations from section 4, Eq. (A.17) becomes the following expression.

$$\begin{aligned} &\sum_{g=1}^{G}\sum_{m=1}^{M} w_m \sum_{k=1}^{\#tracks} b_i(s) \sum_{e \in cell} \partial A_{k,m,e} \left(-\vec{e}_n \bullet \vec{\Omega}_m \right) \psi_{m,g,k}^{(0)}(s) \Big|_0^{\Delta s_{k,m}} - \\ &\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=1}^{\#tracks} A_{k,m} \int_0^{\Delta s_{k,m}} ds \bar{\nabla} b_i \bullet \bar{\Omega}_m \psi_{m,g,k}^{(0)}(s) = 0 \end{aligned} \qquad \text{(A.18)}$$

The analysis of this term was done in the section 4 as well. It was found that assuming incident angular intensities are continuous spatially and fine track spacing, the unknowns at each vertex are continuous and can be defined at this vertex, and the incident unknowns on boundary cells are equal to a Marshak boundary condition given in Eq. (4.72). Without fine track spacing, continuity is lost, even with continuous incident partial currents.

Returning to finding a definition for the first-order current in Eq. (A.15), Eq. (A.7) is multiplied by $\frac{db_i}{ds}$ and a quadrature weight and summed over angles and groups to yield the following $O(1)$ terms.

$$\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=1}^{\#tracks} A_{k,m} \int_0^{\Delta s_{k,m}} ds\vec{\Omega}_m \bullet \vec{\nabla} b_i(s) \left[\begin{array}{l} \dfrac{\partial \psi_{m,g,k}^{(0)}(s)}{\partial s} + \\ \sigma_{a,g}^{(0)} \left(\psi_{m,g,k}^{(1)}(s) - \dfrac{1}{4\pi} B_{g,k}^{(1)}(s) \right) + \\ \sigma_{a,g}^{(1)} \left(\psi_{m,g,k}^{(0)}(s) - \dfrac{1}{4\pi} B_{g,k}^{(0)}(s) \right) \end{array} \right] = 0 \qquad \text{(A.19)}$$

This is the same expression that was analyzed in the previous section starting with Eq. (4.74) (here without the time-dependence). Therefore the result of this analysis given in Eq. (4.93) can be applied for the O(ε) current on interior cells. With this expression for the current, Eq. (A.15) can be written as the following expression.

$$\sum_{cells \in v_i} \left[-\frac{1}{3} \sum_{l=1}^{\#sides} \frac{ac}{\sigma_{R,l}^{(0)}} [K]_l \left(\vec{T}_{cell}^{(0)} \right)^4 - \sum_{g=1}^{G}\sum_{m=1}^{M} \frac{w_m}{4\pi} J_m \sum_{k=1}^{\#tracks} A_{k,m} \int_0^{\Delta s_{k,m}} ds b_i(s) Q_{PWL} \right] = 0 \quad \text{(A.20)}$$

The same mass matrix definition given in Eq. (4.96) can be applied to the second term in this equation.

$$\sum_{cells \in v_i} \left[-\frac{1}{3} \sum_{l=1}^{\#sides} \frac{ac}{\sigma_{R,l}^{(0)}} [K]_l \left(\vec{T}_{cell}^{(0)} \right)^4 - \sum_{g=1}^{G} [A] \vec{Q}_{cell} \right] = 0 \qquad \text{(A.21)}$$

This balance equation must be analyzed for cells on the problem boundary. This analysis will return a different expression for the first-order current on the boundary. The analysis of this first-order current term on the boundary has already been done in section 4.3 beginning with Eq. (4.99). From the result of this analysis, it can be seen that Eq. (A.21) holds for all cells with an added term for boundary cells that brings in the incident flux on sides of the cell that touch the problem boundary. The leading-order

PWL-LC discrete solution satisfies this reasonably discretized equilibrium diffusion equation that is very similar to a steady-state continuous FEM spatially discretized diffusion equation.

A.2. PWC-STLC Asymptotic Analysis in the Diffusion Limit

This section performs an asymptotic analysis of the radiative transfer equations using the PWC-STLC method with approximates the source in a cell as constant in time and space. Please refer to section 4 for any needed definitions. The STLC equation for a cell is found by applying the least-squares procedure (summing over all tracks in the cell for all angles), which is the approximate weighted integral over a cell. This summation produces the following equation.

$$\sum_{m=1}^{M} w_m J_m \sum_{k=1}^{\#\text{ tracks}} A_{k,m} \int_0^{\Delta s_{k,m}} ds \left[\frac{\partial \psi_{m,g,k}}{\partial s} + \frac{\sigma_{a,g}}{\varepsilon} \psi_{m,g,k} - \frac{\sigma_{a,g}}{\varepsilon} \frac{1}{4\pi} B_{g,k} \right] = 0 \qquad \text{(A.22)}$$

The PWC-STLC discretized material temperature equation is given in Eq. (A.23) where the temperature is approximated as constant over each time step and the equation hold independently for each spatial cell.

$$\varepsilon C_v \frac{T^{n+1} - T^n}{\Delta t^n} = \sum_{g=1}^{G} \frac{\sigma_{a,g}}{\varepsilon} \left[\phi_{cell,g}^{n+1/2} - B_{cell,g}^{n+1/2} \right] + \varepsilon Q^{n+1/2} \qquad \text{(A.23)}$$

First, the solution on the interior of the problem is analyzed. Looking at the $O(1/\varepsilon)$ terms from the discretized equation for a track, the isotropic and Planckian nature of the leading-order angular intensity is found as given in Eq. (A.24).

$$\psi_{m,g,k}^{(0)}(s) = \frac{1}{4\pi} B_{g,k}^{(0)}(s) \qquad \text{(A.24)}$$

Next, the $O(1)$ terms from the discretized equation for a cell are shown in Eq. (A.25) when summed over all groups.

$$\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=1}^{\#\text{ tracks}} A_{k,m} \int_0^{\Delta s_{k,m}} ds \left[\frac{\partial \psi_{m,g,k}^{(0)}}{\partial s} + \left[\sigma_{a,g} \left(\psi_{m,g,k} - \frac{1}{4\pi} B_{g,k} \right) \right]^{(1)} \right] = 0 \qquad \text{(A.25)}$$

The $O(1)$ terms from the material energy equation are given in Eq. (A.26).

$$\sum_{g=1}^{G} \left[\sigma_{a,g} \left(\phi_{cell,g}^{n+1/2} - B_{cell,g}^{n+1/2} \right) \right]^{(1)} = 0 \qquad \text{(A.26)}$$

The terms in Eq. (A.26) can be substituted into Eq. (A.25), yielding the following expression.

$$\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=1}^{\#\text{ tracks}} A_{k,m} \int_0^{\Delta s_{k,m}} ds \left[\frac{\partial \psi_{m,g,k}^{(0)}}{\partial s} \right] = 0 \qquad \text{(A.27)}$$

Only tracks of type 1 contribute to this term, because they are of $O(1)$ of the cell volume, which simplifies to the following expression.

$$\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k \in \text{type1}} A_{k,m} \left[\psi_{m,g,k}^{(0)} \left(\Delta s_{k,m} \right) - \psi_{m,g,k}^{(0)} \left(0 \right) \right] = 0 \qquad \text{(A.28)}$$

These leading-order values represent the angular intensity at the exiting and entering position of each type 1 track in the cell. Therefore, for these tracks on a spatial edge of the cell, the expression given in Eq. (4.55) can be used. With this substitution, Eq. (A.28) turns into an expression for the leading-order net current on each spatial edge of the cell as given in Eq. (A.29).

$$\sum_{g=1}^{G}\sum_{m=1}^{M} w_m \sum_{e \in \text{cell}} \sum_{k \in \text{type1}} \partial A_{k,m,e} \left(\vec{e}_n \bullet \vec{\Omega}_m \right) \psi_{m,g,k}^{(0)} = 0 \qquad \text{(A.29)}$$

This relation means that the leading-order cell-edge net currents are zero, at least when summed over a given cell's edges. These edge terms will bring in unknowns from neighboring cells due to these edge net currents. This equation can be split over incoming and outgoing angles and the isotropic leading-order intensity expression can be substituted yielding the following equation, where this equation holds for cells not on the boundary. From this equation, it can be seen that the equation for a cell will include the Planckian of this cell and all of its neighboring cells.

$$\sum_{g=1}^{G}\sum_{e\in\text{cell}}\left[\begin{array}{l}\displaystyle\sum_{\vec{e}_n\bullet\vec{\Omega}_m>0} w_m \sum_{k\in\text{type1}} \partial A_{k,m,e}\left(\vec{e}_n\bullet\vec{\Omega}_m\right)\frac{B^{(0)}_{g,cell}}{4\pi} \\ \displaystyle -\sum_{\vec{e}_n\bullet\vec{\Omega}_m<0} w_m \sum_{k\in\text{type1}} \partial A_{k,m,e}\left|\vec{e}_n\bullet\vec{\Omega}_m\right|\frac{B^{(0)}_{g,neighbor}}{4\pi}\end{array}\right]=0 \qquad \text{(A.30)}$$

Equation (A.30) holds for boundary cells if $B^{(0)}_{g,neighbor}$ is replaced by $\psi_{m,g,inc,bdry}$ on a boundary edge. A Marshak boundary condition on a boundary edge can be defined as given in Eq (A.31).

$$\begin{aligned}&\sum_{g=1}^{G}\sum_{\vec{e}_n\bullet\vec{\Omega}_m<0} w_m \sum_{k\in\text{type1}} \partial A_{k,m,e}\left(\vec{e}_n\bullet\vec{\Omega}_m\right)\frac{B^{(0)}_{g,M,bdry}}{4\pi} \equiv \\ &\sum_{g=1}^{G}\sum_{\vec{e}_n\bullet\vec{\Omega}_m<0} w_m \sum_{k\in\text{type1}} \partial A_{k,m,e}\left|\vec{e}_n\bullet\vec{\Omega}_m\right|\psi_{m,g,inc}\end{aligned} \qquad \text{(A.31)}$$

With this definition for $B^{(0)}_{g,neighbor}$ in (A.30) for boundary edges, this equation holds for all cells in the problem. Therefore, for all *J* cells, there is a set of *J* equations with *J* unknowns, where each of these equations is linearly independent. These equations are an approximate discretization of the Laplacian operator, which is invertible. Therefore, the leading-order Planckian for this PWC-STLC method is determined by the spatial

mesh and the boundary conditions of the problem and does not depend on any physical properties such as sources or opacities or the initial conditions for the time step. This result is expected based upon past research[20,26] and shows that this PWC-STLC method fails in this thick diffusion limit.

Although the steady-state PWC-LC thick diffusion limit analysis was not explicitly included in this dissertation, the results for this PWC-STLC would apply to this method as well. Therefore, the PWC-LC method would fail in the steady-state thick diffusion limit because the leading-order solution depends only on the mesh and the boundary conditions.

A.3. PWL-C-STLC Asymptotic Analysis in the Diffusion Limit

This section performs an asymptotic analysis of the radiative transfer equations using the PWL-C-STLC method. The source in a space-time cell is approximated as PWL in space and constant in time with this method. Using this basis, the STLC transport equations for a cell are again found by approximating the weighted integral over a cell as given in Eq. (A.32) for $i = 1,\ldots,N$, where N equals the number of vertices in the spatial cell.

$$\sum_{m=1}^{M} w_m J_m \sum_{k=1}^{\#\text{ tracks}} A_{k,m} \int_0^{\Delta s_{k,m}} b_i(s) \left[\frac{\partial \psi_{m,g,k}}{\partial s} + \frac{\sigma_{a,g,k}}{\varepsilon} \psi_{m,g,k} - \frac{\sigma_{a,g,k}}{\varepsilon} \frac{1}{4\pi} B_{g,k} \right] ds = 0 \quad \text{(A.32)}$$

The temperature for this approximation is considered to be constant in a time step, so the material energy equation is given in Eq. (A.23). This analysis will begin by looking at different order terms from these scaled and discretized radiative transfer equations on the interior of a problem.

First, the $O(1/\varepsilon)$ terms from the transport equation for a track given in Eq. (4.127) show the isotropic nature of the leading-order angular intensity as was given in Eq. (A.24). The $O(1)$ terms from the i^{th} transport equation given in Eq. (A.32) when summed over all groups are the following.

$$\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=1}^{\#\text{ tracks}} A_{k,m} \int_0^{\Delta s_{k,m}} b_i(s) \left[\frac{\partial \psi_{m,g,k}^{(0)}}{\partial s} + \left[\sigma_{a,g} \left(\psi_{m,g,k} - \frac{1}{4\pi} B_{g,k} \right) \right]^{(1)} \right] ds = 0 \qquad \text{(A.33)}$$

The $O(1)$ terms from the material energy equation are given in Eq. (A.26). The Planckian and scalar intensity are both represented spatially in terms of the PWL basis and are found by the least-squares procedure, which includes the same summation over tracks in the cell as appears in Eq. (A.33). Therefore, Eq. (A.26) can be substituted into Eq. (A.33), yielding the following expression.

$$\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=1}^{\#\text{ tracks}} A_{k,m} \int_0^{\Delta s_{k,m}} b_i(s) \frac{\partial \psi_{m,g,k}^{(0)}}{\partial s} ds = 0 \qquad \text{(A.34)}$$

This term must be analyzed for tracks of type 1 since other track types make an $O(\varepsilon)$ or smaller contribution to the equation. Performing the integration over s by parts gives the following expression.

$$\begin{aligned} &\sum_{g=1}^{G}\sum_{m=1}^{M} w_m \sum_{k=type1} b_i(s) \sum_{e\in cell} \partial A_{k,m,e} \left(\vec{e}_n \bullet \vec{\Omega}_m \right) \psi_{m,g,k}^{(0)} \Big|_0^{\Delta s_{k,m}} - \\ &\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=type1} A_{k,m} \int_0^{\Delta s_{k,m}} ds \vec{\nabla} b_i \bullet \vec{\Omega}_m \psi_{m,g,k}^{(0)}(s) = 0 \end{aligned} \qquad \text{(A.35)}$$

Using the result that the leading-order intensity is isotropic, and assuming a fine track spacing, the integral over s in the second term of Eq. (A.35) will cancel for opposing

angles. The remaining term in Eq. (A.35) has contributions from the spatial edges of the cell that touch the vertex at which the PWL basis function is centered.

$$\sum_{g=1}^{G}\sum_{m=1}^{M} w_m \sum_{k=type1} b_i(s) \sum_{e\in cell} \partial A_{k,m,e} \left(\vec{e}_n \bullet \vec{\Omega}_m\right) \psi^{(0)}_{m,g,k} \Big|_0^{\Delta s_{k,m}} = 0 \qquad \text{(A.36)}$$

Equation (A.36) means that the weighted integral over the spatial edges of a cell should yield a zero net current. Through these net edge currents, the unknowns from neighboring cells come into the equation for this cell.

This equation has already been analyzed in section 4.3 for the PWL-LC method starting with Eq. (4.65). The only difference in this analysis from the previous is the different edge area given by type 1 tracks as opposed to all of the tracks in the cell. It is found that if incident partial currents are continuous spatially and track spacing is fine, then the unknowns at a vertex are continuous and can be defined at this vertex. The unknowns on the spatial edges of boundary cells are then equal to a Marshak boundary condition as found for the PWL-LC method given in Eq. (4.72). Note that without fine track spacing, continuity of unknowns is lost even with continuous incident partial currents on the boundary, as shown in section 4.3.

Next, the $O(\varepsilon)$ terms are analyzed from the least-squares version of the transport equation for a cell. Summing these terms over all groups for the i^{th} PWL basis function yields the following equation.

$$\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \left[\begin{array}{l} \left[\sum_{k=1}^{\#\text{ tracks}} A_{k,m} \int_0^{\Delta s_{k,m}} b_i(s) \frac{\partial \psi_{m,g,k}}{\partial s} \right]^{(1)} + \\ \left[\sum_{k=1}^{\#\text{ tracks}} A_{k,m} \int_0^{\Delta s_{k,m}} b_i(s) \sigma_{a,g,k} \left(\psi_{m,g,k} - \frac{1}{4\pi} B_{g,k} \right) \right]^{(2)} \end{array} \right] ds = 0 \qquad \text{(A.37)}$$

The $O(\varepsilon)$ terms from the material energy equation are given in Eq. (A.38), where the temperature has been written at the midpoint of the time step.

$$C_v \frac{T_j^{(0),n+1/2} - T_j^{(0),n-1/2}}{\Delta t^n} = \sum_{g=1}^{G} \left[\sigma_{a,g} \left(\phi_{g,j}^{n+1/2} - B_{g,j}^{n+1/2} \right) \right]^{(2)} + Q_j^{n+1/2} \qquad \text{(A.38)}$$

These terms can be substituted into Eq. (A.37), because of the least-squares definition of $\phi_{g,j}^{n+1/2}$, to give the following equation.

$$\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \left[\begin{array}{l} \left[\sum_{k=1}^{\#\text{ tracks}} A_{k,m} \int_0^{\Delta s_{k,m}} ds b_i(s) \frac{\partial \psi_{m,g,k}}{\partial s} \right]^{(1)} + \\ \frac{1}{4\pi} \sum_{k=1}^{\#\text{ tracks}} A_{k,m} \int_0^{\Delta s_{k,m}} ds b_i(s) \left[C_v \frac{T_{PWL}^{(0),n+1/2} - T_{PWL}^{(0),n-1/2}}{\Delta t^n} - Q_{PWL} \right] \end{array} \right] = 0 \qquad \text{(A.39)}$$

Tracks of type 1, 3, and 4 must be considered in analyzing this expression. First, since it is assumed the source, *Q*, and leading-order temperature can be written in terms of the PWL basis, these terms from Eq. (A.39) become the following expression.

$$\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=\text{type1}}^{\text{type3,4}} A_{k,m} \int_0^{\Delta s_{k,m}} b_i(s) ds \left(\begin{array}{l} \frac{C_v}{4\pi} \frac{\sum_{j=1}^{N} T_{cell,j}^{(0),n+1/2} b_j(s) - \sum_{j=1}^{N} T_{cell,j}^{(0),n-1/2} b_j(s)}{\Delta t^n} \\ - \frac{1}{4\pi} \sum_{j=1}^{N} Q_{cell,j} b_j(s) \end{array} \right) \qquad \text{(A.40)}$$

From this expression, a track-based mass matrix term is defined as given in Eq. (A.41).

$$\left[A\right]_{ij} \equiv \sum_{l=1}^{\text{Nsides}} \frac{1}{4\pi} \sum_{m=1}^{M} w_m J_m \sum_{k=\text{type1}}^{\text{type3,4}} A_{kl,m} \int_0^{\Delta s_{kl,m}} ds b_i(s) b_j(s) \tag{A.41}$$

Using this definition, Eq. (A.39) becomes the following expression.

$$\begin{aligned} &\sum_{g=1}^{G} \sum_{m=1}^{M} w_m J_m \left[\sum_{k=1}^{\#\text{ tracks}} A_{k,m} \int_0^{\Delta s_{k,m}} b_i(s) \frac{\partial \psi_{m,g,k}}{\partial s} ds \right]^{(1)} + \\ &\sum_{g=1}^{G} \frac{C_v}{\Delta t^n} \left\{ \left[A\right] \left(\vec{T}_{cell}^{(0),n+1/2} - \vec{T}_{cell}^{(0),n-1/2} \right) \right\}_i - \sum_{g=1}^{G} \left\{ \left[A\right] \vec{Q}_{cell} \right\}_i = 0 \end{aligned} \tag{A.42}$$

Returning to the derivative term in this equation, the integral over *s* can be performed by parts giving the following expression.

$$\begin{aligned} &\sum_{g=1}^{G} \sum_{m=1}^{M} w_m J_m \left[\sum_{k=1}^{\#\text{ tracks}} A_{k,m} \int_0^{\Delta s_{k,m}} b_i(s) \frac{\partial \psi_{m,g,k}}{\partial s} ds \right]^{(1)} = \\ &\sum_{g=1}^{G} \sum_{m=1}^{M} w_m J_m \left[\sum_{k=1}^{\#\text{ tracks}} A_{k,m} \left(b_i(s) \psi_{m,g,k}(s) \Big|_0^{\Delta s_{k,m}} - \int_0^{\Delta s_{k,m}} ds \frac{\partial b_i}{\partial s} \psi_{m,g,k}(s) \right) \right]^{(1)} \end{aligned} \tag{A.43}$$

Using Eq. (4.53), the second term resembles a weighted integral of the current along a track. With this substitution, Eq. (A.43) is rewritten as given in Eq. (A.44).

$$\begin{aligned} &\sum_{g=1}^{G} \sum_{m=1}^{M} w_m J_m \left[\sum_{k=1}^{\#\text{ tracks}} A_{k,m} \int_0^{\Delta s_{k,m}} b_i(s) \frac{\partial \psi_{m,g,k}}{\partial s} ds \right]^{(1)} = \\ &\sum_{g=1}^{G} \sum_{m=1}^{M} w_m J_m \left[\sum_{k=1}^{\#\text{ tracks}} A_{k,m} \left(b_i(s) \psi_{m,g,k}(s) \Big|_0^{\Delta s_{k,m}} - \int_0^{\Delta s_{k,m}} ds \vec{\nabla} b_i \bullet \vec{\Omega}_m \psi_{m,g,k}(s) \right) \right]^{(1)} \end{aligned} \tag{A.44}$$

The first term in Eq. (A.44) can be analyzed for tracks of type 1, 3, and 4 along the edges of the cell at which this PWL basis function is nonzero. Considering tracks of type 1, the exiting position of type 3 tracks, and the incoming position of type 4 tracks, the first

term of Eq. (A.44) turns into a summation over the spatial edges of the cell at which this basis function is nonzero as given in expression (A.45).

$$\sum_{g=1}^{G}\sum_{m=1}^{M} w_m \sum_{e\in cell}\sum_{k=\text{type1,3,4}} \partial A_{k,m,e} b_{i,k} \left(\vec{e}_n \bullet \vec{\Omega}_m\right) \psi^{(1)}_{m,g,k} \tag{A.45}$$

Therefore, for these tracks, this term is a weighted net current for the edges of the cell that touch vertex v_i, at which b_i is centered and nonzero. This term will be defined as the following for ease of notation, where it has been noted that the sum of the track areas over a spatial edge approximately yields the area of this edge (as defined previously).

$$\sum_{g=1}^{G}\sum_{e\in cell}\int_{\partial A_e} dA_e b_{i,e} J^{(1)}_e \equiv \sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{e\in cell}\sum_{k=\text{type1,3,4}} \partial A_{k,m,e} b_{i,k} \left(\vec{e}_n \bullet \vec{\Omega}_m\right) \psi^{(1)}_{m,g,k} \tag{A.46}$$

This edge-integrated current is just a defined term that reminds us of the physical meaning of the more complicated expression it replaces, and we use it even if the track-based area on an edge is different than the real area. Now the first term of Eq. (A.44) must be analyzed for tracks of type 3 and 4 which occur on the time boundary of the cell. Since the volume of these tracks is of $O(\varepsilon)$ of the total volume of the cell, the intensity of the first term of Eq. (A.44) will be leading-order. The Jacobian multiplied by the area of each track is directly related to the area of the track on the time boundary of the cell. Also, the basis function has a different value for each spatial position on the time boundaries, but it is constant in time so the top and bottom time boundaries have the same basis values for each spatial position. With this in mind, the first term becomes the following for tracks of type 3 and type 4.

$$\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k\in \text{type3,4}} A_{k,m}\, b_i(s)\psi^{(0)}_{m,g,k}(s)\Big|_0^{\Delta s_{k,m}} = \sum_{g=1}^{G}\sum_{m=1}^{M} w_m \left[\sum_{k\in \text{type4}} \partial A^{n+1}_{k,m} b_{i,k}\psi^{(0),n+1}_{m,g,k} - \sum_{k\in \text{type3}} \partial A^{n}_{k,m} b_{i,k}\psi^{(0),n}_{m,g,k}\right] \tag{A.47}$$

The summations over angles and tracks in Eq. (A.47) should yield the weighted scalar intensity at the beginning and ending time boundaries of the cell, which can also be written in terms of the Planckian. Also, the temperature can be written in terms of the PWL basis functions. A definition for this track defined weighted temperature is given in Eq. (A.48) where the $1/(c\Delta t^n)$ term that is hidden in the summation over tracks has been taken out of the summation.

$$\frac{a}{\Delta t^n}\left[M\right]_i^t\left(\vec{T}^t_{cell}\right)^4 \equiv \sum_{g=1}^{G}\sum_{m=1}^{M} w_m \sum_k \partial A^t_{k,m} b^t_{i,k}\psi^{(0),t}_{m,g,k} = ac\sum_{m=1}^{M} w_m \sum_{k\in \text{type3or4}} \partial A^t_{k,m} b_{i,k}\frac{1}{4\pi}\sum_{j=1}^{N} b_{j,k}\left(T^{(0),t}_{cell,j}\right)^4 , \tag{A.48}$$

where,

$$\frac{1}{c\Delta t^n}\left[M\right]^t_{ij} \equiv \sum_{g=1}^{G}\sum_{m=1}^{M} w_m \sum_k \partial A^t_{k,m} b^t_{i,k} b^t_{j,k} . \tag{A.49}$$

Using this definition, Eq. (A.47) becomes the following expression.

$$\sum_{g=1}^{G}\sum_{m=1}^{M} w_m \left[\sum_{k\in \text{type4}} \partial A^{n+1}_{k,m} b_{i,k}\psi^{(0),n+1}_{m,g,k} - \sum_{k\in \text{type3}} \partial A^{n}_{k,m} b_{i,k}\psi^{(0),n}_{m,g,k}\right] \equiv \frac{a}{\Delta t^n}\left(\left[M\right]_i^{n+1}\left(\vec{T}^{(0),n+1/2}_{cell}\right)^4 - \left[M\right]_i^{n}\left(\vec{T}^{(0),n-1/2}_{cell}\right)^4\right) \tag{A.50}$$

Combining the results of Eq. (A.46) and (A.50), Eq. (A.44) becomes the following expression.

$$\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \left[\sum_{k=1}^{\#\text{ tracks}} A_{k,m} \left(b_i(s)\psi_{m,g,k}(s)\Big|_0^{\Delta s_{k,m}} - \int_0^{\Delta s_{k,m}} ds\vec{\nabla} b_i \bullet \vec{\Omega}_m \psi_{m,g,k}(s) \right) \right]^{(1)} =$$

$$\frac{a}{\Delta t^n}\left([M]_i^{n+1} \left(\vec{T}_{cell}^{(0),n+1/2}\right)^4 - [M]_i^n \left(\vec{T}_{cell}^{(0),n-1/2}\right)^4 \right) + \sum_{g=1}^{G}\sum_{e\in cell}\int_{\partial A_e} dA_e b_{i,e} J_e^{(1)} - \tag{A.51}$$

$$\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \left[\sum_{k=1}^{\#\text{ tracks}} A_{k,m} \int_0^{\Delta s_{k,m}} ds\vec{\nabla} b_i \bullet \vec{\Omega}_m \psi_{m,g,k}(s) \right]^{(1)}$$

At the moment, the last term in this expression will be left as it is. Putting Eq. (A.51) into Eq. (A.42) gives the following equation.

$$\frac{a}{\Delta t^n}\left([M]_i^{n+1} \left(\vec{T}_{cell}^{(0),n+1/2}\right)^4 - [M]_i^n \left(\vec{T}_{cell}^{(0),n-1/2}\right)^4 \right) + \sum_{g=1}^{G}\sum_{e\in cell}\int_{\partial A_e} dA_e b_{i,e} J_e^{(1)} -$$

$$\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \left[\sum_{k=1}^{\#\text{ tracks}} A_{k,m} \int_0^{\Delta s_{k,m}} ds\vec{\nabla} b_i \bullet \vec{\Omega}_m \psi_{m,g,k}(s) \right]^{(1)} + \tag{A.52}$$

$$\sum_{g=1}^{G} \frac{C_v}{\Delta t^n} \left\{ [A]\left(\vec{T}_{cell}^{(0),n+1/2} - \vec{T}_{cell}^{(0),n-1/2}\right) \right\}_i - \sum_{g=1}^{G} \left\{ [A]\vec{Q}_{cell} \right\}_i = 0$$

This equation can be summed over all cells in this time step that touch the spatial vertex, v_i, at which b_i is centered. In doing so, the weighted edge net current on each spatial edge will cancel for this interior vertex and the following equation is found.

$$\sum_{cell\in v_i}\left[\begin{array}{l}\dfrac{a}{\Delta t^n}\left(\left[M\right]_i^{n+1}\left(\vec{T}_{cell}^{(0),n+1/2}\right)^4-\left[M\right]_i^{n}\left(\vec{T}_{cell}^{(0),n-1/2}\right)^4\right)- \\ \displaystyle\sum_{g=1}^{G}\sum_{m=1}^{M}w_m J_m\left[\sum_{k=1}^{\#\text{ tracks}}A_{k,m}\int_0^{\Delta s_{k,m}}ds\vec{\nabla}b_i\bullet\vec{\Omega}_m\psi_{m,g,k}\left(s\right)\right]^{(1)}+ \\ \displaystyle\sum_{g=1}^{G}\frac{C_v}{\Delta t^n}\left\{\left[A\right]\left(\vec{T}_{cell}^{(0),n+1/2}-\vec{T}_{cell}^{(0),n-1/2}\right)\right\}_i-\sum_{g=1}^{G}\left\{\left[A\right]\vec{Q}_{cell}\right\}_i\end{array}\right]=0 \qquad \text{(A.53)}$$

Next, the weighted current in this equation (in the middle line) is analyzed. Again, tracks of type 1, 3, and 4 must be considered for the $O(\varepsilon)$ terms, therefore this term will be split into the following terms.

$$\begin{array}{l}\displaystyle\sum_{g=1}^{G}\sum_{m=1}^{M}w_m J_m\left[\sum_{k=1}^{\#\text{ tracks}}A_{k,m}\int_0^{\Delta s_{k,m}}ds\vec{\nabla}b_i\bullet\vec{\Omega}_m\psi_{m,g,k}\left(s\right)\right]^{(1)}= \\ \displaystyle\sum_{g=1}^{G}\sum_{m=1}^{M}w_m J_m\sum_{k=\text{type1}}A_{k,m}\int_0^{\Delta s_{k,m}}ds\vec{\nabla}b_i\bullet\vec{\Omega}_m\psi_{m,g,k}^{(1)}\left(s\right)+ \\ \displaystyle\sum_{g=1}^{G}\sum_{m=1}^{M}w_m J_m\sum_{k=\text{type3\&4}}A_{k,m}\int_0^{\Delta s_{k,m}}ds\vec{\nabla}b_i\bullet\vec{\Omega}_m\psi_{m,g,k}^{(0)}\left(s\right)\end{array} \qquad \text{(A.54)}$$

The leading-order intensity is isotropic and Planckian and performing this substitution leads to the following expression for type 3 and 4 tracks.

$$\begin{array}{l}\displaystyle\sum_{g=1}^{G}\sum_{m=1}^{M}w_m J_m\sum_{k=\text{type3\&4}}A_{k,m}\int_0^{\Delta s_{k,m}}ds\vec{\nabla}b_i\bullet\vec{\Omega}_m\psi_{m,g,k}^{(0)}\left(s\right)= \\ \displaystyle\sum_{g=1}^{G}\sum_{m=1}^{M}w_m J_m\sum_{k=\text{type3\&4}}A_{k,m}\int_0^{\Delta s_{k,m}}ds\vec{\nabla}b_i\left(s\right)\bullet\vec{\Omega}_m\frac{1}{4\pi}\sum_{j=1}^{N}B_{cell,g,j}^{(0)}b_j\left(s\right)\end{array} \qquad \text{(A.55)}$$

Rearranging this expression leads to the following:

$$\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=\text{type3\&4}} A_{k,m} \int_0^{\Delta s_{k,m}} ds \vec{\nabla} b_i \bullet \vec{\Omega}_m \psi_{m,g,k}^{(0)}(s) = \sum_{g=1}^{G} \frac{1}{4\pi} \sum_{j=1}^{N} B_{cell,g,j}^{(0)} \sum_{m=1}^{M} w_m J_m \vec{\Omega}_m \bullet \sum_{k=\text{type3\&4}} A_{k,m} \int_0^{\Delta s_{k,m}} ds \vec{\nabla} b_i(s) b_j(s) \quad \text{(A.56)}$$

For these tracks of type 3 and 4, given a symmetric quadrature set and fine track spacing, this summation over angle should cancel because the track length of tracks of opposing angles that exit and enter on a time boundary should be the same. With this cancellation, the remaining terms of Eq. (A.54) involve only tracks of type 1.

$$\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \left[\sum_{k=1}^{\#\text{ tracks}} A_{k,m} \int_0^{\Delta s_{k,m}} ds \vec{\nabla} b_i \bullet \vec{\Omega}_m \psi_{m,g,k}(s) \right]^{(1)} = \sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=\text{type1}} A_{k,m} \int_0^{\Delta s_{k,m}} ds \vec{\nabla} b_i \bullet \vec{\Omega}_m \psi_{m,g,k}^{(1)}(s) \quad \text{(A.57)}$$

In order to find an expression for this first-order weighted current, the $O(1)$ terms from the transport equation for a track are multiplied by $\frac{db_i}{ds}$ and then the least-squares summation is applied as well as a summation over groups, giving the terms in Eq. (A.58).

$$\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=\text{type1}}^{\#tracks} A_{k,m} \int_0^{\Delta s_{k,m}} ds\vec{\Omega}_m \bullet \vec{\nabla} b_i(s) \left[\begin{array}{l} \dfrac{\partial \psi_{m,g,k}^{(0)}(s)}{\partial s} + \\ \sigma_{a,g}^{(0)} \left(\psi_{m,g,k}^{(1)}(s) - \dfrac{1}{4\pi} B_{g,k}^{(1)}(s) \right) + \\ \sigma_{a,g}^{(1)} \left(\psi_{m,g,k}^{(0)}(s) - \dfrac{1}{4\pi} B_{g,k}^{(0)}(s) \right) \end{array} \right] = 0 \qquad \text{(A.58)}$$

Considering the last terms of Eq. (A.58):

$$\sum_{g=1}^{G} \sigma_{a,g}^{(1)} \sum_{m=1}^{M} w_m J_m \sum_{k=\text{type1}}^{\#tracks} A_{k,m} \int_0^{\Delta s_{k,m}} ds\vec{\Omega}_m \bullet \vec{\nabla} b_i(s) \left(\psi_{m,g,k}^{(0)}(s) - \frac{1}{4\pi} B_g^{(0)}(s) \right), \qquad \text{(A.59)}$$

the number of tracks of type 1 is growing like $1/\varepsilon$ in the space-time cell as ε becomes small, so the error in this integral over s and summation over all angles of these weighted leading-order terms in Eq. (A.58) will be of $O(\varepsilon)$. Therefore, these leading-order terms in Eq. (A.58) are higher order than $O(1)$ and will not be considered. Returning to Eq. (A.58), assuming the opacity is constant in the cell it can be divided out and taken outside the integral as shown in Eq. (A.60).

$$\begin{array}{l} \displaystyle\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=\text{type1}}^{\#tracks} A_{k,m} \frac{1}{\sigma_{a,g}^{(0)}} \int_0^{\Delta s_{k,m}} ds\vec{\Omega}_m \bullet \vec{\nabla} b_i(s) \frac{\partial \psi_{m,g,k}^{(0)}(s)}{\partial s} + \\ \displaystyle\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=\text{type1}}^{\#tracks} A_{k,m} \int_0^{\Delta s_{k,m}} ds\vec{\Omega}_m \bullet \vec{\nabla} b_i(s) \left(\psi_{m,g,k}^{(1)}(s) - \frac{1}{4\pi} B_g^{(1)}(s) \right) = 0 \end{array} \qquad \text{(A.60)}$$

The Planckian can be written in terms of the PWL basis and this equation can be arranged as given in Eq. (A.61).

$$\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=\text{type1}}^{\#tracks} A_{k,m} \int_0^{\Delta s_{k,m}} ds\vec{\Omega}_m \bullet \vec{\nabla} b_i \psi_{m,g,k}^{(1)} =$$
$$-\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=\text{type1}}^{\#tracks} A_{k,m} \frac{1}{\sigma_{a,g}^{(0)}} \int_0^{\Delta s_{k,m}} ds\vec{\Omega}_m \bullet \vec{\nabla} b_i(s) \frac{\partial \psi_{m,g,k}^{(0)}(s)}{\partial s} + \qquad \text{(A.61)}$$
$$+\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=\text{type1}}^{\#tracks} A_{k,m} \frac{1}{4\pi} \sum_{j=1}^{N} B_{g,cell,j}^{(1)} \int_0^{\Delta s_{k,m}} ds\vec{\Omega}_m \bullet \vec{\nabla} b_i(s) b_j(s)$$

As from previous analyses, if opposing directions use the same tracks with the symmetric quadrature set, the integral over *s* in the last line of Eq. (A.61) will become zero when summed over all angles for tracks of type 1.

In the remaining terms of Eq. (A.61) the expression for the derivative of the leading-order intensity can be substituted and written in terms of the PWL basis.

$$\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=\text{type1}}^{\#tracks} A_{k,m} \int_0^{\Delta s_{k,m}} ds\vec{\Omega}_m \bullet \vec{\nabla} b_i \psi_{m,g,k}^{(1)} =$$
$$-\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=\text{type1}}^{\#tracks} A_{k,m} \frac{1}{4\pi} \sum_{j=1}^{N} B_{g,cell,j}^{(0)} \frac{1}{\sigma_{a,g}^{(0)}} \int_0^{\Delta s_{k,m}} ds\vec{\nabla} b_i \bullet \vec{\Omega}_m \vec{\Omega}_m \bullet \vec{\nabla} b_j \qquad \text{(A.62)}$$

The summation over type 1 tracks can be split into a summation over tracks on sides in the cell.

$$\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=\text{type1}}^{\#tracks} A_{k,m} \int_0^{\Delta s_{k,m}} ds\vec{\Omega}_m \bullet \vec{\nabla} b_i \psi_{m,g,k}^{(1)} =$$
$$-\sum_{g=1}^{G} \frac{1}{4\pi} \frac{1}{\sigma_{a,g}^{(0)}} \sum_{l=1}^{N_{sides}} \sum_{j=1}^{N} B_{g,cell,j}^{(0)} \sum_{m=1}^{M} w_m J_m \sum_{k=\text{type1}}^{\#\text{tracks on } l} A_{kl,m} \int_0^{\Delta s_{kl,m}} ds\vec{\nabla} b_{i,l} \bullet \vec{\Omega}_m \vec{\Omega}_m \bullet \vec{\nabla} b_{j,l} \qquad \text{(A.63)}$$

This integral over *s* evaluates to the length of the track on the side. Therefore, a track volume can be defined for this side for a given angle as given in Eq. (A.64).

$$V_{l,m} \equiv J_m \sum_{k=\text{type1}}^{\#\text{tracks on } l} A_{kl,m} \int_0^{\Delta s_{kl,m}} ds \tag{A.64}$$

Substituting this definition into Eq. (A.63) gives the following equation.

$$\begin{aligned} &\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=\text{type1}}^{\#tracks} A_{k,m} \int_0^{\Delta s_{k,m}} ds \vec{\Omega}_m \bullet \vec{\nabla} b_i \psi_{m,g,k}^{(1)} = \\ &-\sum_{g=1}^{G} \frac{1}{4\pi} \frac{1}{\sigma_{a,g}^{(0)}} \sum_{l=1}^{N_{sides}} \sum_{j=1}^{N} B_{g,cell,j}^{(0)} \vec{\nabla} b_{i,l} \bullet \sum_{m=1}^{M} w_m V_{l,m} \vec{\Omega}_m \vec{\Omega}_m \bullet \vec{\nabla} b_{j,l} \end{aligned} \tag{A.65}$$

The Rosseland mean opacity defined in section 4.3 can be applied to this equation. This definition is found in Eq. (4.85) and Eq. (4.86). Applying this definition to Eq. (A.65) leads to the following equation.

$$\begin{aligned} &\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=\text{type1}}^{\#tracks} A_{k,m} \int_0^{\Delta s_{k,m}} ds \vec{\Omega}_m \bullet \vec{\nabla} b_i \psi_{m,g,k}^{(1)} = \\ &-\frac{1}{4\pi} \frac{ac}{\sigma_{R,l}^{(0)}} \sum_{l=1}^{N_{sides}} \sum_{j=1}^{N} \left(T_{cell,j}^{(0),n+1/2}\right)^4 \vec{\nabla} b_{i,l} \bullet \sum_{m=1}^{M} w_m V_{l,m} \vec{\Omega}_m \vec{\Omega}_m \bullet \vec{\nabla} b_{j,l} \end{aligned} \tag{A.66}$$

A variable can also be defined for the weighted volume sum over angles, which is the same as given in Eq. (4.88). Normally, a symmetric quadrature set would yield $4\pi/3$ multiplied by the identity tensor because the cross terms cancel in the quadrature summation when considering Eq. (4.88). However, for STLC and LC methods, the cross terms in this angular summation do not cancel with a symmetric quadrature set because track volumes on a side are different for each angle given finite track spacing. This tensor is therefore "contaminated" by the contribution from these cross terms and can be defined as was given in Eq. (4.89). Putting this definition into Eq. (A.66) yields the following expression.

$$\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=\text{type1}}^{\#tracks} A_{k,m} \int_0^{\Delta s_{k,m}} ds\vec{\Omega}_m \bullet \vec{\nabla} b_i \psi_{m,g,k}^{(1)} =$$
$$-\frac{1}{4\pi}\frac{ac}{\sigma_{R,l}^{(0)}} \sum_{l=1}^{N_{sides}} \sum_{j=1}^{N} \left(T_{cell,j}^{(0),n+1/2}\right)^4 \frac{1}{4\pi} \vec{\nabla} b_{i,l} \bullet \sum_{m=1}^{M} w_m V_{l,m} \left[\frac{4\pi}{3}\bar{\bar{I}} + \delta\bar{\bar{\tau}}_l\right] \bullet \vec{\nabla} b_{j,l} \tag{A.67}$$

This expression has a contaminated stiffness matrix that was defined in Eq. (4.91). With this definition, Eq. (A.67) becomes the following equation.

$$\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=\text{type1}}^{\#tracks} A_{k,m} \int_0^{\Delta s_{k,m}} ds\vec{\Omega}_m \bullet \vec{\nabla} b_i \psi_{m,g,k}^{(1)} = -\frac{1}{3}\sum_{l=1}^{N_{sides}} \frac{ac}{\sigma_{R,l}^{(0)}} [K]_l \left(\vec{T}_{cell}^{(0),n+1/2}\right)^4 \tag{A.68}$$

Finally, the expression for this weighted first-order current can be put back into Eq. (A.53) to give the following balance equation.

$$\sum_{cell \in v_i} \left[\begin{array}{l} \frac{a}{\Delta t^n}\left([M]^{n+1}\left(\vec{T}_{cell}^{(0),n+1/2}\right)^4 - [M]^n\left(\vec{T}_{cell}^{(0),n-1/2}\right)^4\right) - \\ \frac{1}{3}\sum_{l=1}^{N_{sides}} \frac{ac}{\sigma_{R,l}^{(0)}} [K]_l \left(\vec{T}_{cell}^{(0),n+1/2}\right)^4 + \\ \sum_{g=1}^{G} \frac{C_v}{\Delta t^n} [A]\left(\vec{T}_{cell}^{(0),n+1/2} - \vec{T}_{cell}^{(0),n-1/2}\right) - \sum_{g=1}^{G} [A]\vec{Q}_{cell} \end{array} \right] = 0 \tag{A.69}$$

Next, this balance equation must be considered for a cell on the boundary of the problem for a time step after the initial time step. When considering this cell, the edge terms that previously cancelled from Eq. (A.52) must be considered. This analysis is very similar to the boundary analysis for the PWL-LC method that began with Eq. (4.99) so it will not be detailed here. The exiting position of type 3 tracks and entering position of type 4 tracks in the cell are lumped with the type 1 tracks in this boundary analysis since these all occur on spatial edges of the cell. The results from this boundary analysis show that the boundary condition seen by interior cells is different than the Marshak

boundary condition defined previously. With this boundary expression for the weighted first-order current (given in Eq. (4.121)), Eq. (A.69) is defined for all cells and for all time steps after the initial for $i = 1,\ldots, N$.

Finally, the leading-order solution for this PWL-C-STLC method must be defined for the initial time step. The analog of Eq. (A.69) for the first time step is given in Eq. (A.70). It is assumed the initial condition for a cell can be written in terms of the basis.

$$\sum_{cell\in v_i}\left[\begin{array}{l}\dfrac{a}{\Delta t^n}\left(\left[M\right]^{n+1}\left(\vec{T}_{cell}^{(0),n+1/2}\right)^4\right)- \\ \displaystyle\sum_{g=1}^{G}\sum_{m=1}^{M} w_m \sum_{k\in \text{type3}} \partial A_{k,m}^n b_{i,k}\psi_{m,g,k}^{(0),i} - \\ \dfrac{1}{3}\displaystyle\sum_{l=1}^{N_{sides}} \frac{ac}{\sigma_{R,l}^{(0)}}\left[K\right]_l\left(\vec{T}_{cell}^{(0),n+1/2}\right)^4 + \\ \displaystyle\sum_{g=1}^{G}\frac{C_v}{\Delta t^n}\left[A\right]\left(\vec{T}_{cell}^{(0),n+1/2}-\vec{T}_{cell}^{(0),i}\right)-\sum_{g=1}^{G}\left[A\right]\vec{Q}_{cell}\end{array}\right]=0 \qquad \text{(A.70)}$$

A general initial condition on each cell must be able to reproduce Eq. (A.69). If the initial intensity is Planckian at the initial temperature, Eq. (A.70) would revert back to Eq. (A.69). In order for a general initial condition to reproduce this equation, the initial condition must satisfy the following expression, where *n* denotes the beginning of the first time step and an *i* superscript denotes the initial conditions as defined for the original transport problem.

$$\left\{\frac{a}{\Delta t^n}\left[M\right]^n\left(\vec{T}_{cell}^{(0),n}\right)^4+\sum_{g=1}^{G}\frac{C_v}{\Delta t^n}\left[A\right]\left(\vec{T}_{cell}^{(0),n}\right)\right\}_{n=0}= \sum_{g=1}^{G}\sum_{m=1}^{M}w_m\sum_{k\in\text{type3}}\partial A_{k,m}^n b_{i,k}\psi_{m,g,k}^{(0),i}+\sum_{g=1}^{G}\frac{C_v}{\Delta t^n}\left[A\right]\left(\vec{T}_{cell}^{(0),i}\right) \tag{A.71}$$

A.4. PWL-L-STLC Asymptotic Analysis in the Diffusion Limit

This section performs a detailed analysis of the PWL-L-STLC method applied to the radiative transfer equations in the thick diffusion limit. The source in a space-time cell with this method is approximated as PWL in space with one added function to the basis that is linear in time, otherwise known as the time-slope. With this basis, the STLC transport equation for a cell is given as written in Eq. (A.72), defined for $i = 1,\ldots, N+1$.

$$\sum_{m=1}^{M}w_m J_m\sum_{k=1}^{\#\text{ tracks}}A_{k,m}\int_0^{\Delta s_{k,m}}b_i(s)\left[\frac{\partial\psi_{m,g,k}}{\partial s}+\frac{\sigma_{a,g}}{\varepsilon}\psi_{m,g,k}-\frac{\sigma_{a,g}}{\varepsilon}\frac{1}{4\pi}B_{g,k}\right]ds=0 \tag{A.72}$$

The material energy equation for this PWL-L-STLC method is given in Eq. (A.73), where this expression holds for all spatial points.

$$\varepsilon C_v\frac{\partial T_j}{\partial t}=\sum_{g=1}^{G}\frac{\sigma_{a,g}}{\varepsilon}\left[\phi_{g,j}-B_{g,j}\right]+\varepsilon Q_j \tag{A.73}$$

The temperature at a given spatial basis point is linear in time and can be written as the following:

$$T_j(t)=T_j^{n+1/2}+\left(T_z^{n+1}-T_z^{n+1/2}\right)\frac{2\left(t-t^{n+1/2}\right)}{\Delta t^n},\qquad j=1,\ldots,N, \tag{A.74}$$

where z refers to the spatial center of the space-time cell and $t^{n+1/2}$ refers to the midpoint of the time step. The T_z terms in Eq. (A.74) are defined using the track-based volume average over the space-time cell as given later in this analysis.

From the $O(1/\varepsilon)$ terms from the transport equation for one track, it is again found that the leading-order intensity is isotropic and Planckian as given in Eq. (A.75).

$$\psi_{m,g,k}^{(0)}(s) = \frac{1}{4\pi} B_{g,k}^{(0)}(s) \tag{A.75}$$

Next, the $O(1)$ terms from Eq. (A.72) are analyzed. Summing this equation over all groups yields the following $O(1)$ terms.

$$\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=1}^{\#\text{ tracks}} A_{k,m} \int_0^{\Delta s_{k,m}} b_i(s) \left[\frac{\partial \psi_{m,g,k}^{(0)}}{\partial s} + \left[\sigma_{a,g} \left(\psi_{m,g,k} - \frac{1}{4\pi} B_{g,k} \right) \right]^{(1)} \right] ds = 0 \tag{A.76}$$

The $O(1)$ terms from the material energy equation are given in Eq. (A.77).

$$\sum_{g=1}^{G} \left[\sigma_{a,g} \left(\phi_{g,j} - B_{g,j} \right) \right]^{(1)} = 0 \tag{A.77}$$

This equation holds for every spatial point and these terms can be substituted into Eq. (A.76), turning this equation into the following expression.

$$\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=1}^{\#\text{ tracks}} A_{k,m} \int_0^{\Delta s_{k,m}} b_i(s) \frac{\partial \psi_{m,g,k}^{(0)}}{\partial s} ds = 0 \tag{A.78}$$

This term must be analyzed for tracks of type 1 since they make up $O(1)$ of the cell volume. Performing integration by parts leads to the following expression.

$$\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=type1} A_{k,m} b_i(s) \psi^{(0)}_{m,g,k}(s)\Big|_0^{\Delta s_{k,m}} - \sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=type1} A_{k,m} \int_0^{\Delta s_{k,m}} ds \vec{\nabla} b_i(s) \bullet \vec{\Omega}_m \psi^{(0)}_{m,g,k}(s) = 0 \quad \text{(A.79)}$$

Looking at the second term in Eq. (A.79), this track volume summation is an approximation to the analytic integral of the weighted current in the space-time cell. The track volume summation can be thought of as a "quadrature" set, that becomes finer as $\varepsilon \to 0$, that is used to approximate the actual volume summation of the cell. Since the leading-order intensity is isotropic and can be written in terms of the basis, it is a smooth function that is at most quadratic in *s*. Therefore, the error in approximating the analytic integral of this function as given in Eq. (A.79) is at most $O(\varepsilon)$ for fine track spacing and tracks of type 1 and it can be ignored for these terms of $O(1)$. With this cancellation, the remaining term in Eq. (A.79) is the following.

$$\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=type1} A_{k,m} b_i(s) \psi^{(0)}_{m,g,k}(s)\Big|_0^{\Delta s_{k,m}} = 0 \quad \text{(A.80)}$$

This expression turns into a weighted net current on the spatial edges of the cell for tracks of type 1, as was shown in previous analyses. Equation (A.81) shows the weighted net current expression, which holds for $i = 1,\ldots,N+1$.

$$\sum_{g=1}^{G}\sum_{m=1}^{M} w_m \sum_{k=type1} b_i(s) \sum_{e \in cell} \partial A_{k,m,e} \left(\vec{e}_n \bullet \vec{\Omega}_m\right) \psi^{(0)}_{m,g,k} = 0 \quad \text{(A.81)}$$

For the $i = 1,\ldots, N$ basis functions that are PWL spatially, this equation turns into a weighted net current on each of the spatial edges of the cell at which b_i is nonzero, as was done in the previous analyses involving the spatial PWL basis. This expression

means the weighted net current on the spatial edges of the cell at which the PWL basis is nonzero is equal to zero, and this equation brings in the unknowns from neighboring cells across these edges. The analog of this equation can be written for an interior vertex of a boundary cell. Similar to the analysis performed in section 4.3 for these PWL basis functions, if the incident angular intensities are assumed to be continuous spatially, the unknowns at each spatial vertex are found to be continuous given fine track spacing. Also the unknowns on the boundary can be written as a Marshak condition as was done in the previous PWL analyses. Note again that without fine track spacing, continuity of unknowns is lost at vertices, even with continuous incident partial currents on the boundary.

Returning to Eq. (A.80), for the $i = N+1$ basis function, this weighted net current summation includes all of the spatial edges and brings in unknowns from all of the cell's neighbors. This equation can be written with this $N+1$ basis function in it as given in Eq. (A.82).

$$\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=type1} A_{k,m}\left[\frac{2\left(t_0 - t^{n+1/2}\right)}{\Delta t^n} + \frac{2\varepsilon s}{c\Delta t^n}\right]\psi_{m,g,k}^{(0)}(s)\Bigg|_0^{\Delta s_{k,m}} = 0 \qquad \text{(A.82)}$$

The $N+1$ basis function has been written in terms of the s variable as given in expression (A.83),

$$\frac{2\left(t - t^{n+1/2}\right)}{\Delta t^n} = \frac{2\left(t_0 - t^{n+1/2}\right)}{\Delta t^n} + \frac{2\varepsilon s}{c\Delta t^n}, \qquad \text{(A.83)}$$

where t_0 refers to the starting time of the track and the "speed" changes with epsilon as was done in the linear STLC analysis in (x, t)[12]. The second term in brackets in Eq.

(A.82) is of higher order so it is not included with the terms of $O(1)$. Therefore, this equation turns into the following summation over the spatial edges of the cell for tracks of type 1.

$$\sum_{g=1}^{G}\sum_{m=1}^{M} w_m \sum_{e\in cell} \sum_{k=type1} \partial A_{k,m,e}\left(\vec{e}_n \bullet \vec{\Omega}_m\right) \frac{2\left(t_0 - t^{n+1/2}\right)}{\Delta t^n} \psi^{(0)}_{m,g,k}(s)\Big|_0^{\Delta s_{k,m}} = 0 \qquad \text{(A.84)}$$

Assuming fine track spacing, this equation means the time moment of the net current at the spatial edges of the cell is zero. Next, the leading-order intensity is written as isotropic and in terms of the basis.

$$\sum_{g=1}^{G}\sum_{m=1}^{M} w_m \sum_{e\in cell} \sum_{k=type1} \partial A_{k,m,e}\left(\vec{e}_n \bullet \vec{\Omega}_m\right) \frac{2\left(t_0 - t^{n+1/2}\right)}{\Delta t^n} \frac{1}{4\pi} \sum_{j=1}^{N+1} B^{(0)}_{g,j} b_{j,k} = 0 \qquad \text{(A.85)}$$

Also, the angular summation can be split into incoming and outgoing angles on an edge, which brings in the scalar intensity from the neighboring cell across the edge.

$$\sum_{g=1}^{G} \frac{1}{4\pi} \left[\begin{array}{l} \displaystyle\sum_{e\in cell} \sum_{j=1}^{N+1} B^{cell(0)}_{g,j} \sum_{\vec{e}_n \bullet \vec{\Omega}_m > 0} w_m \sum_{k=type1} \partial A_{k,m,e}\left(\vec{e}_n \bullet \vec{\Omega}_m\right) \frac{2\left(t_0 - t^{n+1/2}\right)}{\Delta t^n} b_{j,k} - \\ \displaystyle\sum_{e\in cell} \sum_{j=1}^{N+1} B^{neigh(0)}_{g,j} \sum_{\vec{e}_n \bullet \vec{\Omega}_m < 0} w_m \sum_{k=type1} \partial A_{k,m,e}\left|\left(\vec{e}_n \bullet \vec{\Omega}_m\right)\right| \frac{2\left(t_0 - t^{n+1/2}\right)}{\Delta t^n} b_{j,k} \end{array} \right] = 0 \qquad \text{(A.86)}$$

This expression is an approximation of the analytic integral on a spatial edge of the time-moment of the current. The integration of the spatial basis functions, $i = 1,..., N$, in this expression will go to zero and only the N+1 basis function remains. The following time-moment of the scalar intensity can then be defined, which holds for either half angle summation

$$\rho_3 A_e B_g^{t,(0)} \equiv \sum_{\vec{e}_n \bullet \vec{\Omega}_m > 0} w_m \sum_{k=type1} \partial A_{k,m,e} \left(\vec{e}_n \bullet \vec{\Omega}_m \right) \frac{2\left(t_0 - t^{n+1/2}\right)}{4\pi \Delta t^n} b_{N+1,k} B_{g,N+1}^{(0)} \qquad \text{(A.87)}$$

Note that in the limit of fine track spacing, the extra factor, ρ_3, would limit to unity because the summation over tracks on an edge for all angles would adequately integrate the area of the edge and half of the symmetric quadrature set. Using this definition, Eq. (A.86) becomes the following expression.

$$\sum_{g=1}^{G} \sum_{e \in cell} \left[\rho_3 A_e B_g^{t,cell,(0)} - \rho_3 A_e B_g^{t,neigh,(0)} \right] = 0 \qquad \text{(A.88)}$$

This expression means that the time-moment of the Planckian is determined only by the boundary condition on the cell.

Equation (A.88) can also be written for a cell on the boundary by using the incident intensity on a boundary edge. Equation (A.86) for a boundary cell is given in the following expression.

$$\sum_{g=1}^{G} \left[\begin{array}{l} \sum_{e \notin bdry} \left(\rho_3 A_e B_g^{t,cell,(0)} - \rho_3 A_e B_g^{t,neigh,(0)} \right) + \\ \sum_{e \in bdry} \left(\begin{array}{l} \sum_{\vec{e}_n \bullet \vec{\Omega}_m > 0} w_m \sum_{k=type1} \partial A_{k,m,e} \left(\vec{e}_n \bullet \vec{\Omega}_m \right) \frac{2\left(t_0 - t^{n+1/2}\right)}{\Delta t^n} \frac{1}{4\pi} B_{m,g,bdry,k}^{(0)} - \\ \sum_{\vec{e}_n \bullet \vec{\Omega}_m < 0} w_m \sum_{k=type1} \partial A_{k,m,e} \left| \left(\vec{e}_n \bullet \vec{\Omega}_m \right) \right| \frac{2\left(t_0 - t^{n+1/2}\right)}{\Delta t^n} \psi_{m,g,k,inc}^{(0)} \end{array} \right) \end{array} \right] = 0 \qquad \text{(A.89)}$$

Recall that the Planckian is a PWL function and it is assumed the incident intensity is a PWL function as well. With this in mind, the time-moment of the Planckian on the boundary is defined as the following:

$$\sum_{g=1}^{G}\sum_{\vec{e}_n\bullet\vec{\Omega}_m<0} w_m \sum_{k=type1} \partial A_{k,m,e}\left|\left(\vec{e}_n\bullet\vec{\Omega}_m\right)\right| B^{t,(0)}_{g,M,bdry,j} \equiv$$
$$\sum_{\vec{e}_n\bullet\vec{\Omega}_m<0} w_m \sum_{k=type1} \partial A_{k,m,e}\left|\left(\vec{e}_n\bullet\vec{\Omega}_m\right)\right| \frac{2\left(t_0 - t^{n+1/2}\right)}{\Delta t^n} \psi^{(0)}_{m,g,k,inc,j} \quad , \tag{A.90}$$

Eq. (A.88) now holds for all cells. This set of equations for all cells means that the time-slope of the Planckian in a cell is an interpolated value using the boundary condition and no other physical quantities. This unphysical time-slope behavior affects the accuracy of this PWL-L-STLC method in the thick diffusion limit as discussed later in this section.

Next, the $O(\varepsilon)$ terms are analyzed. Summing the transport equation over all groups, these terms are the following.

$$\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \left[\begin{array}{l} \left[\sum_{k=1}^{\#\text{ tracks}} A_{k,m} \int_0^{\Delta s_{k,m}} dsb_i(s) \frac{\partial \psi_{m,g,k}}{\partial s}\right]^{(1)} + \\ \left[\sum_{k=1}^{\#\text{ tracks}} A_{k,m} \int_0^{\Delta s_{k,m}} dsb_i(s) \sigma_{a,g}\left(\psi_{m,g,k} - \frac{1}{4\pi} B_{g,k}\right)\right]^{(2)} \end{array}\right] = 0 \tag{A.91}$$

The $O(\varepsilon)$ terms from the material energy equation are given in Eq. (A.92).

$$C_v \frac{\partial T_j^{(0)}}{\partial t} - Q_j = \sum_{g=1}^{G}\left[\sigma_{a,g}\left(\phi_{g,j} - B_{g,j}\right)\right]^{(2)} \tag{A.92}$$

Recall that the unknowns in Eq. (A.92) are found by the least-squares procedure. Therefore, the second order terms from Eq. (A.92) can be substituted into Eq. (A.91).

$$\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \left[\begin{array}{l} \left[\sum_{k=1}^{\#\text{ tracks}} A_{k,m} \int_0^{\Delta s_{k,m}} dsb_i(s) \frac{\partial \psi_{m,g,k}}{\partial s} \right]^{(1)} + \\ C_v \frac{1}{4\pi} \left[\sum_{k=1}^{\#\text{ tracks}} A_{k,m} \int_0^{\Delta s_{k,m}} dsb_i(s) \frac{\partial T}{\partial t} \right]^{(0)} - \\ \frac{1}{4\pi} \sum_{k=1}^{\#\text{ tracks}} A_{k,m} \int_0^{\Delta s_{k,m}} dsb_i(s) Q_{cell} \end{array} \right] = 0 \qquad \text{(A.93)}$$

Tracks of type 1, 3, and 4 must be considered when analyzing this expression. First, the source term is analyzed. This source can be written in terms of the basis and is written as the following expression:

$$\sum_{g=1}^{G}\sum_{j=1}^{N+1} Q_{cell,j} \frac{1}{4\pi} \sum_{m=1}^{M} w_m J_m \sum_{k=\text{type1,3,4}} A_{k,m} \int_0^{\Delta s_{k,m}} dsb_i(s) b_j(s). \qquad \text{(A.94)}$$

Similar to the analysis for the PWL-C-STLC method, a track-based mass matrix is defined in Eq. (A.95) for $i = 1,\ldots,N$.

$$[A]_{ij} \equiv \frac{1}{4\pi} \sum_{l=1}^{N_{sides}} \sum_{m=1}^{M} w_m J_m \sum_{k=\text{type1,3,4}} A_{kl,m} \int_0^{\Delta s_{kl,m}} dsb_i(s) b_j(s) \qquad \text{(A.95)}$$

This definition turns Eq. (A.93) into the following expression.

$$\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \left[\begin{array}{l} \left[\sum_{k=1}^{\#\text{ tracks}} A_{k,m} \int_0^{\Delta s_{k,m}} dsb_i(s) \frac{\partial \psi_{m,g,k}}{\partial s} \right]^{(1)} + \\ C_v \frac{1}{4\pi} \left[\sum_{k=1}^{\#\text{ tracks}} A_{k,m} \int_0^{\Delta s_{k,m}} dsb_i(s) \frac{\partial T}{\partial t} \right]^{(0)} \end{array} \right] - \sum_{g=1}^{G} \left\{ [A] \vec{Q}_{cell} \right\}_i = 0 \qquad \text{(A.96)}$$

Now the leading-order temperature term in this expression is analyzed. Since tracks of type 3 and 4 are an $O(\varepsilon)$ of the total cell volume, they are higher order when considering this term and therefore do not contribute. The leading-order temperature can

be written in terms of the N spatial basis functions. For the $i = 1,\ldots,N$ basis functions, the weighted temperature derivative can be written as given in Eq. (A.97), which uses the expression for the j^{th} temperature as was given in Eq. (A.74).

$$b_i(s)\frac{\partial T^{(0)}}{\partial t} \to b_i(s)\sum_{j=1}^{N} b_j(s)\frac{T_{cell,j}^{(0),n+1/2} + T_z^{(0),n+1} - T_z^{(0),n+1/2} - T_{cell,j}^{(0),n}}{\Delta t^n} \tag{A.97}$$

Putting this expression back into the temperature terms in Eq. (A.96) and remembering only tracks of type 1 are considered, this expression becomes the following for $i = 1,\ldots,N$.

$$\begin{aligned}&\sum_{g=1}^{G} C_v \frac{1}{4\pi}\sum_{m=1}^{M} w_m J_m \sum_{k=type1} A_{k,m}\int_0^{\Delta s_{k,m}} ds b_i(s)\frac{\partial T^{(0)}}{\partial t} = \\ &\sum_{g=1}^{G} \frac{C_v}{4\pi\Delta t^n}\sum_{m=1}^{M} w_m J_m \sum_{k=type1} A_{k,m}\int_0^{\Delta s_{k,m}} ds b_i \sum_{j=1}^{N} b_j \begin{pmatrix} T_{cell,j}^{(0),n+1/2} + T_z^{(0),n+1} - \\ T_z^{(0),n+1/2} - T_{cell,j}^{(0),n}\end{pmatrix}\end{aligned} \tag{A.98}$$

The average temperature at the center of the cell is defined as given in Eq. (A.99).

$$\begin{aligned}&\sum_{m=1}^{M} w_m J_m \sum_{k=type1} A_{k,m}\int_0^{\Delta s_{k,m}} ds \sum_{j=1}^{N} b_j T_{cell,j}^{(0),n+1/2} \equiv \\ &T_z^{(0),n+1/2}\sum_{m=1}^{M} w_m J_m \sum_{k=type1} A_{k,m}\int_0^{\Delta s_{k,m}} ds \sum_{j=1}^{N} b_j\end{aligned} \tag{A.99}$$

Returning to Eq. (A.98), the same mass matrix defined in Eq. (A.95) can be applied here for $i = 1,\ldots,N$. Applying this mass matrix definition, Eq. (A.98) becomes the following expression:

$$\sum_{g=1}^{G} C_v \frac{1}{4\pi} \sum_{m=1}^{M} w_m J_m \sum_{k=type1} A_{k,m} \int_0^{\Delta s_{k,m}} dsb_i(s) \frac{\partial T^{(0)}}{\partial t} = \sum_{g=1}^{G} \frac{C_v}{\Delta t^n} \left\{ [A] \left(\vec{T}_{cell}^{(0),n+1/2} - \vec{T}_{cell}^{(0),n} + \vec{T}_z^{(0),n+1} - \vec{T}_z^{(0),n+1/2} \right) \right\}_i , \quad \text{(A.100)}$$

where,

$$\vec{T}_{cell}^{(0),n+1/2} = \begin{bmatrix} T_{cell,1}^{(0),n+1/2} \\ \vdots \\ T_{cell,N}^{(0),n+1/2} \end{bmatrix}, \quad \vec{T}_{cell}^{(0),n} = \begin{bmatrix} T_{cell,1}^{(0),n} \\ \vdots \\ T_{cell,N}^{(0),n} \end{bmatrix},$$
$$\vec{T}_z^{(0),n+1} = \begin{bmatrix} T_z^{(0),n+1} \\ \vdots \\ T_z^{(0),n+1} \end{bmatrix}, \quad \vec{T}_z^{(0),n+1/2} = \begin{bmatrix} T_z^{(0),n+1/2} \\ \vdots \\ T_z^{(0),n+1/2} \end{bmatrix}. \quad \text{(A.101)}$$

Finally, putting Eq. (A.100) back into Eq. (A.96), the following expression has been found for $i = 1,\ldots,N$.

$$\sum_{g=1}^{G} \sum_{m=1}^{M} w_m J_m \left[\sum_{k=1}^{\#\text{ tracks}} A_{k,m} \int_0^{\Delta s_{k,m}} dsb_i(s) \frac{\partial \psi_{m,g,k}}{\partial s} \right]^{(1)} + \sum_{g=1}^{G} \frac{C_v}{\Delta t^n} \left\{ [A] \left(\vec{T}_{cell}^{(0),n+1/2} - \vec{T}_{cell}^{(0),n} + \vec{T}_z^{(0),n+1} - \vec{T}_z^{(0),n+1/2} \right) \right\}_i - \sum_{g=1}^{G} \left\{ [A] \vec{Q}_{cell} \right\}_i = 0 \quad \text{(A.102)}$$

Note that the equation for $i = N+1$ is not considered for these terms of $O(\varepsilon)$ because it will not be needed to fully define the system of unknowns which include the following: the temperature at each spatial basis point (N), the temperature at the center of the space-time cell ($T_z^{(0),n+1/2}$) defined in Eq. (A.99), and the temperature at the center of the cell at the end of the time step ($T_z^{(0),n+1}$). These unknowns will be fully defined using the N

balance equations (one for each spatial basis function) found shortly, Eq. (A.99), and Eq. (A.88).

Finally, the first term in Eq. (A.96) and Eq. (A.102) involving the derivative of the angular intensity is analyzed. First, integration by parts is performed to give the following expression.

$$\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \left[\sum_{k=1}^{\#\text{ tracks}} A_{k,m} \int_0^{\Delta s_{k,m}} ds b_i(s) \frac{\partial \psi_{m,g,k}}{\partial s} \right]^{(1)} = \sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \left[\sum_{k=1}^{\#\text{ tracks}} A_{k,m} \left(b_i(s)\psi_{m,g,k}(s)\Big|_0^{\Delta s_{k,m}} - \int_0^{\Delta s_{k,m}} ds \frac{\partial b_i}{\partial s}\psi_{m,g,k}(s) \right) \right]^{(1)} \quad \text{(A.103)}$$

This term must be split over tracks of type 1 and tracks of type 3 and 4.

$$\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \left[\sum_{k=1}^{\#\text{ tracks}} A_{k,m} \left(b_i(s)\psi_{m,g,k}(s)\Big|_0^{\Delta s_{k,m}} - \int_0^{\Delta s_{k,m}} ds \frac{\partial b_i}{\partial s}\psi_{m,g,k}(s) \right) \right]^{(1)} = \sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=type1} A_{k,m} \left(b_i(s)\psi^{(1)}_{m,g,k}(s)\Big|_0^{\Delta s_{k,m}} - \int_0^{\Delta s_{k,m}} ds \vec{\nabla} b_i \bullet \vec{\Omega}_m \psi^{(1)}_{m,g,k}(s) \right) + \sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=type3\&4} A_{k,m} \left(b_i(s)\psi^{(0)}_{m,g,k}(s)\Big|_0^{\Delta s_{k,m}} - \int_0^{\Delta s_{k,m}} ds \vec{\nabla} b_i \bullet \vec{\Omega}_m \psi^{(0)}_{m,g,k}(s) \right) \quad \text{(A.104)}$$

Considering the middle line of expression (A.104), this term turns into a summation over the spatial edges of the cell for these tracks of type 1. The exiting position of type 3 tracks and the incoming position of type 4 tracks also occur on these spatial edges and can be put with the type 1 tracks.

$$\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=type1} A_{k,m}\left(b_i(s)\psi^{(1)}_{m,g,k}(s)\Big|_0^{\Delta s_{k,m}}\right) = \sum_{g=1}^{G}\sum_{m=1}^{M} w_m \sum_{e\in cell}\sum_{k=\text{type1(3,4)}} \partial A_{k,m,e} b_{i,k}\left(\vec{e}_n \bullet \vec{\Omega}_m\right)\psi^{(1)}_{m,g,k} \tag{A.105}$$

For the $i = 1,\ldots,N$ PWL basis functions, this expression is a weighted net current for the edges of the cell that b_i is nonzero. A compact notation for this can be defined as was done in Eq. (A.46).

$$\sum_{g=1}^{G}\sum_{e\in cell}\int_{\partial A_e} dA_e b_{i,e} J^{(1)}_e \equiv \sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{e\in cell}\sum_{k=\text{type1(3,4)}} \partial A_{k,m,e} b_{i,k}\left(\vec{e}_n \bullet \vec{\Omega}_m\right)\psi^{(1)}_{m,g,k} \tag{A.106}$$

Putting the results of Eq. (A.106) in Eq. (A.104), this expression has become the following expression for $i = 1,\ldots,N$.

$$\begin{aligned}
&\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \left[\sum_{k=1}^{\#\text{ tracks}} A_{k,m}\left(b_i(s)\psi_{m,g,k}(s)\Big|_0^{\Delta s_{k,m}} - \int_0^{\Delta s_{k,m}} ds \frac{\partial b_i}{\partial s}\psi_{m,g,k}(s)\right)\right]^{(1)} = \\
&\sum_{g=1}^{G}\sum_{e\in cell}\int_{\partial A_e} dA_e b_{i,e} J^{(1)}_e - \\
&\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=type1} A_{k,m}\left(\int_0^{\Delta s_{k,m}} ds \vec{\nabla} b_i \bullet \vec{\Omega}_m \psi^{(1)}_{m,g,k}(s)\right) + \\
&\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=type3\&4} A_{k,m}\left(\begin{array}{l} b_i(s)\psi^{(0)}_{m,g,k}(s)\Big|_0^{\Delta s_{k,m}} - \\ \int_0^{\Delta s_{k,m}} ds \vec{\nabla} b_i \bullet \vec{\Omega}_m \psi^{(0)}_{m,g,k}(s)\end{array}\right)
\end{aligned} \tag{A.107}$$

For the moment, the weighted first-order angular intensity term remaining in this equation will be left as it is.

The terms containing the leading-order intensity from this expression are now analyzed. The leading-order angular intensity is isotropic and this is substituted into these terms from Eq. (A.107).

$$\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=type3\&4} A_{k,m}\left(\begin{array}{l} b_i(s)\psi^{(0)}_{m,g,k}(s)\Big|_0^{\Delta s_{k,m}} - \\ \int_0^{\Delta s_{k,m}} ds\vec{\nabla} b_i \bullet \vec{\Omega}_m \psi^{(0)}_{m,g,k}(s) \end{array}\right) =$$
$$\sum_{g=1}^{G}\sum_{j=1}^{N+1}\frac{1}{4\pi}\sum_{m=1}^{M} w_m J_m \sum_{k=type3\&4} A_{k,m}\left(\begin{array}{l} b_i(s)b_j(s)\phi^{(0)}_{cell,j,g}\Big|_0^{\Delta s_{k,m}} - \\ \int_0^{\Delta s_{k,m}} ds\vec{\nabla} b_i \bullet \vec{\Omega}_m b_j(s)\phi^{(0)}_{cell,j,g} \end{array}\right) \quad \text{(A.108)}$$

The second term in Eq. (A.108) can be analyzed for the different basis functions in this method (spatial and time). As was done in the previous PWL-C-STLC method, for i = 1,…,N and j = 1,…,N this second term will cancel given a symmetric quadrature set and fine track spacing, because the track length of tracks of opposing angles that exit and enter on a time boundary should be the same and therefore cancel in this summation. Finally, for i = 1,…, N and j = N+1, the b_j basis function is a constant with a positive value for tracks of type 4 and negative value for tracks of type 3. Also, the gradient of the spatial b_i functions is a constant on each side. Therefore, the integral over s on each side of the cell will be approximately zero for opposing angles. All together, the second term in expression (A.108) cancels or is not included for all basis functions. The remaining term in expression (A.108) is the following:

$$\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=type3\&4} A_{k,m}\left(\begin{array}{l} b_i(s)\psi^{(0)}_{m,g,k}(s)\Big|_0^{\Delta s_{k,m}} - \\ \int_0^{\Delta s_{k,m}} ds\vec{\nabla} b_i \bullet \vec{\Omega}_m \psi^{(0)}_{m,g,k}(s) \end{array}\right) =$$
$$\sum_{g=1}^{G}\sum_{j=1}^{N+1}\frac{1}{4\pi}\sum_{m=1}^{M} w_m J_m \sum_{k=type3\&4} A_{k,m} b_i(s)b_j(s)\phi^{(0)}_{cell,j,g}\Big|_0^{\Delta s_{k,m}} \quad \text{(A.109)}$$

This term can be thought of as a summation of the starting and exiting positions of these tracks on the cell edges, on the beginning and ending time boundaries, because

the type 3&4 tracks are within O(ε) of the time boundaries. With this in mind, this term will be written separately for tracks of type 3 and tracks of type 4.

$$\sum_{g=1}^{G}\sum_{j=1}^{N+1}\frac{1}{4\pi}\sum_{m=1}^{M}w_m J_m \sum_{k=type3\&4} A_{k,m}\, b_i(s) b_j(s) \phi^{(0)}_{cell,j,g}\Big|_0^{\Delta s_{k,m}} =$$

$$\sum_{g=1}^{G}\frac{1}{4\pi}\sum_{m=1}^{M}w_m J_m \left[\begin{array}{l} \displaystyle\sum_{k\in \text{type3}} A_{k,m}\sum_{j=1}^{N+1}\left(\begin{array}{l}\left(b_i(s) b_j(s)\phi^{(0)}_{cell,j,g}\right)_{s=\Delta s_{k,m}} - \\ \left(b_i(s)\phi^{(0),n}_{g}\right)_{s=0}\end{array}\right) + \\ \displaystyle\sum_{k\in \text{type4}} A_{k,m}\sum_{j=1}^{N+1}\left(\begin{array}{l}\left(b_i(s) b_j(s)\phi^{(0)}_{cell,j,g}\right)_{s=\Delta s_{k,m}} - \\ \left(b_i(s) b_j(s)\phi^{(0)}_{cell,j,g}\right)_{s=0}\end{array}\right)\end{array}\right] \qquad \text{(A.110)}$$

Note that the incoming scalar intensity from the previous time step is denoted as $\phi_g^{(0),n}$, and it can be written in terms of the spatial basis as well. These terms are now analyzed considering different basis functions and assuming fine track spacing with a symmetric quadrature set. Since both of these types of tracks are within $O(\varepsilon)$ of their respective time boundaries, the scalar intensity for the tracks of type 3 is the scalar intensity at the bottom of the cell, or the start of the time step. Similarly, for tracks of type 4, the scalar intensity considered in these terms is the intensity at the top of the space-time cell, at the end of the time step. For $i = 1,\ldots, N$, and $j = 1,\ldots, N$ the exiting positions of tracks of type 3, which occur on a spatial boundary of the cell, cancel with the incoming position of tracks of type 4 for opposing angles. For $i = 1,\ldots, N$ and $j = N+1$, the ending position of tracks of type 3 occur on a spatial edge of the cell and this term can be written as the following.

$$\sum_{g=1}^{G}\frac{1}{4\pi}\sum_{m=1}^{M}w_m\sum_{e\in cell}\sum_{k\in \text{type}3}\partial A_{k,m,e}\left(\vec{e}_n\bullet\vec{\Omega}_m\right)\left(b_i(s)b_{N+1}(s)\phi^{(0)}_{cell,N+1,g}\right)_{s=\Delta s_{k,m}} =$$
$$\sum_{g=1}^{G}\frac{1}{4\pi}\sum_{m=1}^{M}w_m\sum_{e\in cell}\sum_{k\in \text{type}3}\partial A_{k,m,e}\left(\vec{e}_n\bullet\vec{\Omega}_m\right)b_{i,e}(-1)\left(\phi^{(0),n+1}_{z,g}-\phi^{(0),n+1/2}_{z,g}\right) \quad \text{(A.111)}$$

A similar term also appears for the beginning position of tracks of type 4 for $i = 1,\ldots, N$ and $j = N+1$, which occur on the spatial edges of the cell.

$$\sum_{g=1}^{G}\frac{1}{4\pi}\sum_{m=1}^{M}w_m\sum_{e\in cell}\sum_{k\in \text{type}4}\partial A_{k,m,e}\left(\vec{e}_n\bullet\vec{\Omega}_m\right)\left(-\left(b_i(s)b_{N+1}(s)\phi^{(0)}_{cell,N+1,g}\right)_{s=0}\right) =$$
$$-\sum_{g=1}^{G}\frac{1}{4\pi}\sum_{m=1}^{M}w_m\sum_{e\in cell}\sum_{k\in \text{type}4}\partial A_{k,m,e}\left(\vec{e}_n\bullet\vec{\Omega}_m\right)b_{i,e}(1)\left(\phi^{(0),n+1}_{z,g}-\phi^{(0),n+1/2}_{z,g}\right) \quad \text{(A.112)}$$

Given fine track spacing, the terms of Eq. (A.111) and Eq. (A.112) add together and become the following term.

$$\sum_{g=1}^{G}\frac{1}{4\pi}\sum_{m=1}^{M}w_m\sum_{e\in cell}\sum_{k\in \text{type}3}\partial A_{k,m,e}\left(\vec{e}_n\bullet\vec{\Omega}_m\right)\left(b_i(s)b_{N+1}(s)\phi^{(0)}_{cell,N+1,g}\right)_{s=\Delta s_{k,m}} +$$
$$\sum_{g=1}^{G}\frac{1}{4\pi}\sum_{m=1}^{M}w_m\sum_{e\in cell}\sum_{k\in \text{type}4}\partial A_{k,m,e}\left(\vec{e}_n\bullet\vec{\Omega}_m\right)\left(-\left(b_i(s)b_{N+1}(s)\phi^{(0)}_{cell,N+1,g}\right)_{s=0}\right) =$$
$$-\sum_{g=1}^{G}\frac{1}{4\pi}\sum_{m=1}^{M}w_m\sum_{e\in cell}\sum_{k\in \text{type}3\&4}\partial A_{k,m,e}\left(\vec{e}_n\bullet\vec{\Omega}_m\right)b_{i,e}2\left(\phi^{(0),n+1}_{z,g}-\phi^{(0),n+1/2}_{z,g}\right) \quad \text{(A.113)}$$

The expression in Eq. (A.113) cancels due to the summation over angle, because with fine track spacing, opposing angles yield the same weighted scalar intensity with opposite signs on each spatial edge.

The ending position of tracks of type 4 must also be analyzed. For $i = 1,\ldots, N$ and $j = 1,\ldots, N$, this term from Eq. (A.110) can be written in terms of the scalar intensity at the end of the time-step.

$$\sum_{g=1}^{G}\frac{1}{4\pi}\sum_{m=1}^{M}w_m\sum_{k\in\text{type4}}\partial A_{k,m}^{n+1}\left(b_i(s)\sum_{j=1}^{N}b_j(s)\phi_{cell,j,g}^{(0)}\right)_{s=\Delta s_{k,m}}=$$
$$\sum_{g=1}^{G}\frac{1}{4\pi}\sum_{m=1}^{M}w_m\sum_{k\in\text{type4}}\partial A_{k,m}^{n+1}b_{i,k}\sum_{j=1}^{N}b_{j,k}\phi_{cell,j,g}^{(0),n+1} \qquad \text{(A.114)}$$

For $j = N+1$, this scalar intensity can be written as the following expression for the ending position of type 4 tracks.

$$\sum_{g=1}^{G}\frac{1}{4\pi}\sum_{m=1}^{M}w_m\sum_{k\in\text{type4}}\partial A_{k,m}^{n+1}\left(b_i(s)b_{N+1}(s)\phi_{cell,N+1,g}^{(0)}\right)_{s=\Delta s_{k,m}}=$$
$$\sum_{g=1}^{G}\frac{1}{4\pi}\sum_{m=1}^{M}w_m\sum_{k\in\text{type4}}\partial A_{k,m}^{n+1}b_{i,k}(1)\left(\phi_{z,g}^{(0),n+1}-\phi_{z,g}^{(0),n+1/2}\right)= \qquad \text{(A.115)}$$
$$\sum_{g=1}^{G}\frac{1}{4\pi}\sum_{m=1}^{M}w_m\sum_{k\in\text{type4}}\partial A_{k,m}^{n+1}b_{i,k}\sum_{j=1}^{N}b_{j,k}\left(\phi_{z,g}^{(0),n+1}-\phi_{z,g}^{(0),n+1/2}\right)$$

The summation over b_j for $j = 1,\ldots, N$ sums to unity in Eq. (A.115) so it can be included in this expression. A mass matrix is defined in Eq. (A.116).

$$\frac{1}{c\Delta t^n}[M]_{ij}^{n+1}\equiv\frac{1}{4\pi}\sum_{m=1}^{M}w_m\sum_{k\in\text{type4}}\partial A_{k,m}^{n+1}b_{i,k}b_{j,k} \qquad \text{(A.116)}$$

Combining the results of Eq. (A.114) and Eq. (A.115), the ending position of tracks of type 4 are written in terms of a mass matrix as shown in Eq. (A.117),

$$\sum_{g=1}^{G}\frac{1}{4\pi}\sum_{m=1}^{M}w_m\sum_{k\in\text{type4}}\partial A_{k,m}^{n+1}\left(b_i(s)b_j(s)\phi_{cell,j,g}^{(0)}\right)_{s=\Delta s_{k,m}}=$$
$$\frac{1}{c\Delta t^n}\sum_{g=1}^{G}[M]^{n+1}\vec{\phi}_{cell,g}^{(0),n+1}, \qquad \text{(A.117)}$$

where,

$$\vec{\phi}_{cell,g}^{(0),n+1} \equiv \begin{bmatrix} \phi_{cell,1,g}^{(0),n+1} \\ \vdots \\ \phi_{cell,N+1,g}^{(0),n+1} \end{bmatrix}. \tag{A.118}$$

Returning to Eq. (A.110), the results of the previous analysis are applied. For i = 1,…, N, this expression becomes the following.

$$\begin{aligned} &\sum_{g=1}^{G}\sum_{j=1}^{N+1}\frac{1}{4\pi}\sum_{m=1}^{M}w_m J_m \sum_{k=type3\&4} A_{k,m}\, b_i(s) b_j(s) \phi_{cell,j,g}^{(0)}\Big|_0^{\Delta s_{k,m}} = \\ &\frac{1}{c\Delta t^n}\sum_{g=1}^{G}\left\{[M]^{n+1}\vec{\phi}_{cell,g}^{(0),n+1}\right\}_i - \sum_{g=1}^{G}\frac{1}{4\pi}\sum_{m=1}^{M}w_m \sum_{k\in \text{type3}} A_{k,m}^n \sum_{j=1}^{N+1}\left(b_i(s)\phi_g^{(0),n}\right)_{s=0} \end{aligned} \tag{A.119}$$

Putting the result of Eq. (A.109) and this previous equation back into Eq. (A.107), these $O(\varepsilon)$ terms are the following expression for i = 1,…, N.

$$\begin{aligned} &\sum_{g=1}^{G}\sum_{m=1}^{M}w_m J_m\left[\sum_{k=1}^{\#\text{ tracks}} A_{k,m}\int_0^{\Delta s_{k,m}} ds b_i(s)\frac{\partial \psi_{m,g,k}}{\partial s}\right]^{(1)} = \\ &\sum_{g=1}^{G}\sum_{e\in cell}\int_{\partial A_e} dA_e b_{i,e} J_e^{(1)} - \\ &\sum_{g=1}^{G}\sum_{m=1}^{M}w_m J_m \sum_{k=type1} A_{k,m}\left(\int_0^{\Delta s_{k,m}} ds \vec{\nabla} b_i \bullet \vec{\Omega}_m \psi_{m,g,k}^{(1)}(s)\right) + \quad i = 1,...,N \\ &\frac{1}{c\Delta t^n}\sum_{g=1}^{G}\left\{[M]^{n+1}\vec{\phi}_{cell,g}^{(0),n+1}\right\}_i - \\ &\sum_{g=1}^{G}\frac{1}{4\pi}\sum_{m=1}^{M}w_m \sum_{k\in \text{type3}} A_{k,m}^n \sum_{j=1}^{N+1}\left(b_i(s)\phi_g^{(0),n}\right)_{s=0}, \end{aligned} \tag{A.120}$$

This expression is put back into the balance equation given in Eq. (A.102). For i = 1,…,N this balance equation is the following.

$$\frac{1}{c\Delta t^n}\sum_{g=1}^{G}\left\{\left[M\right]^{n+1}\vec{\phi}_{cell,g}^{(0),n+1}\right\}_i-\sum_{g=1}^{G}\sum_{e\in cell}A_e b_{i,e}\left(\phi_{z,g}^{(0),n+1}-\phi_{z,g}^{(0),n+1/2}\right)-$$
$$\sum_{g=1}^{G}\frac{1}{4\pi}\sum_{m=1}^{M}w_m\sum_{k\in \text{type}3}A_{k,m}^{n}\sum_{j=1}^{N+1}\left(b_i(s)\phi_g^{(0),n}\right)_{s=0}+\sum_{g=1}^{G}\sum_{e\in cell}\int_{\partial A_e}dA_e b_{i,e}J_e^{(1)}-$$
$$\sum_{g=1}^{G}\sum_{m=1}^{M}w_m J_m\sum_{k=type1}A_{k,m}\left(\int_0^{\Delta s_{k,m}}ds\vec{\nabla}b_i\bullet\vec{\Omega}_m\psi_{m,g,k}^{(1)}(s)\right)+ \qquad \text{(A.121)}$$
$$\sum_{g=1}^{G}\frac{C_v}{\Delta t^n}\left\{\left[A\right]\left(\vec{T}_{cell}^{(0),n+1/2}-\vec{T}_{cell}^{(0),n}+\vec{T}_z^{(0),n+1}-\vec{T}_z^{(0),n+1/2}\right)\right\}_i-$$
$$\sum_{g=1}^{G}\left\{[A]\vec{Q}_{cell}\right\}_i=0$$

The incoming scalar intensity from the previous time step is written in terms of basis functions as given in Eq. (A.122), which is similar to Eq. (A.116).

$$\frac{1}{c\Delta t^n}\sum_{g=1}^{G}\left\{\left[M\right]^{n}\vec{\phi}_{cell,g}^{(0),n}\right\}_i\equiv\sum_{g=1}^{G}\frac{1}{4\pi}\sum_{m=1}^{M}w_m\sum_{k\in \text{type}3}\partial A_{k,m}^{n}b_{i,k}\sum_{j=1}^{N+1}b_{j,k}\phi_{cell,j,g}^{(0),n}=$$
$$\sum_{g=1}^{G}\frac{1}{4\pi}\sum_{m=1}^{M}w_m\sum_{k\in \text{type}3}A_{k,m}^{n}\sum_{j=1}^{N+1}\left(b_i(s)\phi_g^{(0),n}\right)_{s=0} \qquad \text{(A.122)}$$

Equation (A.121) is summed over all of the cells that touch the vertex at which the basis function is defined for $i = 1,\ldots, N$. In doing so, the net edge currents and edge intensities cancel for neighboring cells and the following expression is found.

$$\sum_{cells\in v_i}\left[\begin{array}{l}\frac{1}{c\Delta t^n}\sum_{g=1}^{G}\left\{\left[M\right]^{n+1}\vec{\phi}_{cell,g}^{(0),n+1}\right\}_i-\frac{1}{c\Delta t^n}\sum_{g=1}^{G}\left\{\left[M\right]^{n}\vec{\phi}_{cell,g}^{(0),n}\right\}_i-\\ \sum_{g=1}^{G}\sum_{m=1}^{M}w_m J_m\sum_{k=type1}A_{k,m}\left(\int_0^{\Delta s_{k,m}}ds\vec{\nabla}b_i\bullet\vec{\Omega}_m\psi_{m,g,k}^{(1)}(s)\right)+\\ \sum_{g=1}^{G}\frac{C_v}{\Delta t^n}\left\{\left[A\right]\left(\vec{T}_{cell}^{(0),n+1/2}-\vec{T}_{cell}^{(0),n}+\vec{T}_z^{(0),n+1}-\vec{T}_z^{(0),n+1/2}\right)\right\}_i-\\ \sum_{g=1}^{G}\left\{[A]\vec{Q}_{cell}\right\}_i\end{array}\right]=0 \qquad \text{(A.123)}$$

Note that due to the use of finite track spacing, the area of the beginning and ending time boundaries of an *x*-*y*-*vt* cell is usually different, and this is denoted in the equation above. This fact is also the reason why this method may oscillate around a steady state solution when considering a time-dependent problem in the steady-state limit. Now an expression for the weighted first-order current in Eq. (A.123) is determined for an interior cell.

To find an expression for this weighted first-order current over type 1 tracks, the $O(1)$ terms from the transport equation for a track are multiplied by $\frac{db_i}{ds}$ and the least-squares summation is applied as well as a summation over groups, giving the terms in Eq. (A.124).

$$\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=\text{type1}} A_{k,m} \int_0^{\Delta s_{k,m}} ds \vec{\Omega}_m \bullet \vec{\nabla} b_i(s) \left[\begin{array}{l} \frac{\partial \psi_{m,g,k}^{(0)}(s)}{\partial s} + \\ \sigma_{a,g}^{(0)} \left(\psi_{m,g,k}^{(1)}(s) - \frac{1}{4\pi} B_{g,k}^{(1)}(s) \right) + \\ \sigma_{a,g}^{(1)} \left(\psi_{m,g,k}^{(0)}(s) - \frac{1}{4\pi} B_{g,k}^{(0)}(s) \right) \end{array} \right] = 0 \qquad \text{(A.124)}$$

Considering the last terms of Eq. (A.124), tracks of type 1 are growing like $1/\varepsilon$ in the cell, so the error that occurs in integrating this term over s for all angles is of $O(\varepsilon)$ and will not be included with these $O(1)$ terms. Next, rearranging other terms left in Eq. (A.124) and writing the Planckian in terms of the basis, the following expression remains.

$$\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=\text{type1}} A_{k,m} \int_0^{\Delta s_{k,m}} ds\vec{\Omega}_m \bullet \vec{\nabla} b_i \psi_{m,g,k}^{(1)} =$$
$$-\sum_{g=1}^{G} \frac{1}{\sigma_{a,g}^{(0)}} \sum_{m=1}^{M} w_m J_m \sum_{k=\text{type1}} A_{k,m} \int_0^{\Delta s_{k,m}} ds\vec{\Omega}_m \bullet \vec{\nabla} b_i(s) \frac{\partial \psi_{m,g,k}^{(0)}(s)}{\partial s} + \qquad \text{(A.125)}$$
$$+\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=\text{type1}} A_{k,m} \frac{1}{4\pi} \sum_{j=1}^{N+1} \int_0^{\Delta s_{k,m}} ds\vec{\Omega}_m \bullet \vec{\nabla} b_i(s) b_j(s) B_{g,cell,j}^{(1)}$$

Again, with fine track spacing and a symmetric quadrature set, the integral over *s* on the Planckian term in Eq. (A.125) becomes zero when summed over all angles for type 1 tracks and $i = 1,\ldots,N$ basis functions.

In the remaining terms of Eq. (A.125), the expression for the leading-order intensity is substituted and written in terms of the basis functions as given in Eq. (A.126).

$$\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=\text{type1}} A_{k,m} \int_0^{\Delta s_{k,m}} ds\vec{\Omega}_m \bullet \vec{\nabla} b_i \psi_{m,g,k}^{(1)} =$$
$$-\sum_{g=1}^{G} \frac{1}{\sigma_{a,g}^{(0)}} \sum_{m=1}^{M} w_m J_m \sum_{k=\text{type1}} A_{k,m} \sum_{j=1}^{N+1} \frac{1}{4\pi} \int_0^{\Delta s_{k,m}} ds\vec{\nabla} b_i \bullet \vec{\Omega}_m \vec{\Omega}_m \bullet \vec{\nabla} b_j B_{g,j}^{(0)} \qquad \text{(A.126)}$$

The spatial gradient of the $j = N+1$ basis function is zero so this term is not included in the summation over *j* and Eq. (A.126) becomes the following expression.

$$\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=\text{type1}} A_{k,m} \int_0^{\Delta s_{k,m}} ds\vec{\Omega}_m \bullet \vec{\nabla} b_i \psi_{m,g,k}^{(1)} =$$
$$-\sum_{g=1}^{G} \frac{1}{\sigma_{a,g}^{(0)}} \sum_{m=1}^{M} w_m J_m \sum_{k=\text{type1}} A_{k,m} \sum_{j=1}^{N} \frac{1}{4\pi} \int_0^{\Delta s_{k,m}} ds\vec{\nabla} b_i \bullet \vec{\Omega}_m \vec{\Omega}_m \bullet \vec{\nabla} b_j B_{g,j}^{(0)} \qquad \text{(A.127)}$$

This expression was analyzed in section 4.3 beginning with Eq. (4.80). This analysis results in the definition of a stiffness matrix and a Rosseland mean opacity. Both of

these definitions are applied to this term along with the rest of the analysis in this section to give the following expression for this weighted O(ε) current,

$$\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=\text{type1}} A_{k,m} \int_0^{\Delta s_{k,m}} ds\vec{\Omega}_m \bullet \vec{\nabla} b_i \psi_{m,g,k}^{(1)} = \frac{1}{3}\sum_{l=1}^{\#\text{sides}} \frac{ac}{\sigma_{R,l}^{(0)}} [K]_l \left(\vec{T}_{cell}^{(0),n+1/2}\right)^4 \quad \text{(A.128)}$$

where,

$$\vec{T}^{(0),n+1/2} = \begin{bmatrix} T_1^{(0),n+1/2} \\ \vdots \\ T_N^{(0),n+1/2} \end{bmatrix}. \quad \text{(A.129)}$$

The expression for the weighted first-order current found in Eq. (A.128) for i =1,…,N is substituted into the balance equation given in Eq. (A.123).

$$\sum_{cells\in v_i} \left[\begin{array}{l} \frac{1}{c\Delta t^n}\sum_{g=1}^{G}[M]^{n+1}\vec{\phi}_{cell,g}^{(0),n+1} - \frac{1}{c\Delta t^n}\sum_{g=1}^{G}[M]^{n}\vec{\phi}_{cell,g}^{(0),n} - \\ \frac{1}{3}\sum_{l=1}^{\#\text{sides}} \frac{ac}{\sigma_{R,l}^{(0)}}[K]_l \left(\vec{T}_{cell}^{(0),n+1/2}\right)^4 + \\ \sum_{g=1}^{G}\frac{C_v}{\Delta t^n}[A]\left(\vec{T}_{cell}^{(0),n+1/2} - \vec{T}_{cell}^{(0),n} + \vec{T}_z^{(0),n+1} - \vec{T}_z^{(0),n+1/2}\right) - \\ \sum_{g=1}^{G}[A]\vec{Q}_{cell} \end{array} \right] = 0 \quad \text{(A.130)}$$

The leading-order scalar intensity is Planckian and is written in terms of the temperature and summed over energy groups. The temperature at the end of the time step can be written as the following expression.

$$\begin{aligned} &\frac{1}{c\Delta t^n}\sum_{g=1}^{G}[M]^{n+1}\vec{\phi}_{cell,g}^{(0),n+1} = \frac{1}{c\Delta t^n}[M]^{n+1} ac\left(\vec{T}_{cell}^{(0),n+1}\right)^4 = \\ &[M]^{n+1}\frac{a}{\Delta t^n}\left[\left(\vec{T}_{cell}^{(0),n+1/2}\right)^4 + \left(\vec{T}_z^{(0),n+1}\right)^4 - \left(\vec{T}_z^{(0),n+1/2}\right)^4\right] \end{aligned} \quad \text{(A.131)}$$

There is a similar expression to Eq. (A.131) for the scalar intensity at the beginning of the time step. Substituting these expressions into Eq. (A.130) yields the following balance equation for interior cells.

$$\sum_{cells\in v_i}\left[\begin{array}{l}\frac{a}{\Delta t^n}[M]^{n+1}\left[\left(\vec{T}_{cell}^{(0),n+1/2}\right)^4+\left(\vec{T}_z^{(0),n+1}\right)^4-\left(\vec{T}_z^{(0),n+1/2}\right)^4\right]- \\ \frac{a}{\Delta t^n}[M]^{n}\left[\left(\vec{T}_{cell}^{(0),n-1/2}\right)^4+\left(\vec{T}_z^{(0),n}\right)^4-\left(\vec{T}_z^{(0),n-1/2}\right)^4\right]- \\ \frac{1}{3}\sum_{l=1}^{\#sides}\frac{ac}{\sigma_{R,l}^{(0)}}[K]_l\left(\vec{T}_{cell}^{(0),n+1/2}\right)^4+ \\ \sum_{g=1}^{G}\frac{C_v}{\Delta t^n}[A]\left(\vec{T}_{cell}^{(0),n+1/2}-\vec{T}_{cell}^{(0),n}+\vec{T}_z^{(0),n+1}-\vec{T}_z^{(0),n+1/2}\right)- \\ \sum_{g=1}^{G}[A]\vec{Q}_{cell}\end{array}\right]=0 \qquad \text{(A.132)}$$

The balance equation given in Eq. (A.132), which is defined for $i=1,\ldots,N$, along with the defined temperature at the center of the space-time cell given in Eq. (A.99), and the unphysical result for the time-slope of the Planckian found in Eq. (A.88), form the system for the N+2 unknowns for an interior cell. The time-slope terms given in Eq. (A.88) are related to the temperature at the spatial center of the cell as shown in Eq. (A.133).

$$\begin{array}{l}\sum_{g=1}^{G}\sum_{e\in cell}\left[\rho_3 A_e B_g^{t,cell,(0)}-\rho_3 A_e B_g^{t,neigh,(0)}\right]\Rightarrow \\ ac\sum_{e\in cell}\left[\rho_3 A_e\left(T_z^{t,cell,(0)}\right)^4-\rho_3 A_e\left(T_z^{t,neigh,(0)}\right)^4\right]=0\end{array}, \qquad \text{(A.133)}$$

where,

$$\left(\vec{T}_z^{(0),t}\right)^4 \equiv \left(\vec{T}_z^{(0),n+1}\right)^4 - \left(\vec{T}_z^{(0),n+1/2}\right)^4, \quad \vec{T}_z^{(0),t} \equiv \vec{T}_z^{(0),n+1} - \vec{T}_z^{(0),n+1/2} \tag{A.134}$$

Without the extra time slope terms present in this balance equation, it resembles a fully-implicit FEM discretized balance equation. These extra time slope terms can make the solution to a given problem more or less accurate than a fully-implicit method. Since these time slope terms are not determined by anything physical, they will most likely lead to a less accurate solution. Note that if the given boundary condition is PWC in time, these extra time slope terms go to zero as can be seen from Eq. (A.88). This balance equation must also be defined in the boundary layer of the problem.

Considering the balance equation given in Eq. (A.132) for $i = 1,\ldots,N$, for a cell on the boundary, the edge terms that previously cancelled must be considered. Returning to the expression for the first-order weighted current for tracks of type 1, the weighted current is defined as was done in the previous analysis for ease of notation, as given in Eq. (A.135).

$$J_{tracks}^{(1),i} \equiv \sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=type1} A_{k,m}\left(\int_0^{\Delta s_{k,m}} ds \vec{\nabla} b_i \bullet \vec{\Omega}_m \psi_{m,g,k}^{(1)}(s)\right) = -\sum_{g=1}^{G}\frac{1}{\sigma_{a,g}^{(0)}}\sum_{m=1}^{M} w_m J_m \sum_{k=\text{type1}} A_{k,m}\int_0^{\Delta s_{k,m}} ds \vec{\Omega}_m \bullet \vec{\nabla} b_i(s)\frac{\partial \psi_{m,g,k}^{(0)}(s)}{\partial s} \tag{A.135}$$

This summation is split over the sides of a boundary cell to yield the following expression.

$$J_{tracks}^{(1),i} = -\sum_{g=1}^{G}\frac{1}{\sigma_{a,g}^{(0)}}\sum_{m=1}^{M} w_m J_m \sum_{l=1}^{N_{sides}}\sum_{k=\text{type1}} A_{kl,m}\int_0^{\Delta s_{kl,m}} ds \vec{\Omega}_m \bullet \vec{\nabla} b_i(s)\vec{\Omega}_m \bullet \vec{\nabla}\psi_{m,g,k}^{(0)}(s) \tag{A.136}$$

Next, the leading-order intensity is written in the following form:

$$\psi^{(0)}_{m,g,k}(s) = \psi^{(0)}_{m,g,k}(s) - \frac{B^{(0)}_k(s)}{4\pi} + \frac{B^{(0)}_k(s)}{4\pi}. \tag{A.137}$$

Substituting this expression into Eq. (A.136) leads to the following expression for a cell on the boundary.

$$J^{(1),i}_{tracks} = -\sum_{g=1}^{G} \frac{1}{\sigma^{(0)}_{a,g}} \sum_{m=1}^{M} w_m J_m \sum_{l=1}^{Nsides} \sum_{k=\text{type1}} A_{kl,m} \rightarrow$$
$$\int_0^{\Delta s_{kl,m}} ds \vec{\Omega}_m \bullet \vec{\nabla} b_i(s) \vec{\Omega}_m \bullet \vec{\nabla} \left[\begin{array}{l} \psi^{(0)}_{m,g,k}(s) - \\ \frac{B^{(0)}_k(s)}{4\pi} + \\ \frac{B^{(0)}_k(s)}{4\pi} \end{array} \right] \tag{A.138}$$

This expression has been analyzed on the boundary in section 4.3 beginning with Eq. (4.103). The results of this boundary analysis hold for this PWL-L-STLC method as well, therefore the interior cells see a different boundary condition from the Marshak boundary condition. Also it is again noted that only the surface-averaged boundary condition affects the interior solution, therefore only one value is needed per spatial boundary edge instead of a PWL representation. With this defined boundary expression for the weighted first-order current, the balance equation given in Eq. (A.132) is defined for all cells for $i = 1,\ldots, N$.

Finally, the leading-order solution for this PWL-L-STLC method is considered for the initial time step. The analog of Eq. (A.132) for the first time step is given in Eq. (A.139). It is assumed the initial condition for a cell can be written in terms of the spatial basis. The superscript i means the initial condition specified for the original

transport problem, whereas *n* means the corresponding initial condition for the diffusion-limit problem.

$$\sum_{cells\in v_i}\left[\begin{array}{l}\frac{a}{\Delta t^n}[M]^{n+1}\left[\left(\vec{T}_{cell}^{(0),n+1/2}\right)^4+\left(\vec{T}_z^{(0),n+1}\right)^4-\left(\vec{T}_z^{(0),n+1/2}\right)^4\right]- \\ \frac{a}{\Delta t^n}[M]^n\left[\left(\vec{T}_{cell}^{(0),i}\right)^4\right]- \\ \frac{1}{3}\sum_{l=1}^{\#sides}\frac{ac}{\sigma_{R,l}^{(0)}}[K]_l\left(\vec{T}_{cell}^{(0),n+1/2}\right)^4+ \\ \sum_{g=1}^{G}\frac{C_v}{\Delta t^n}[A]\left(\vec{T}_{cell}^{(0),n+1/2}-\vec{T}_{cell}^{(0),i}+\vec{T}_z^{(0),n+1}-\vec{T}_z^{(0),n+1/2}\right)- \\ \sum_{g=1}^{G}[A]\vec{Q}_{cell}\end{array}\right]=0 \qquad \text{(A.139)}$$

A general initial condition on each cell must be able to reproduce Eq. (A.132). If the initial intensity is Planckian at the initial temperature, Eq. (A.139) would revert back to the balance equation of Eq. (A.132). For a general initial condition to reproduce this equation, the initial condition must satisfy the following expression.

$$\left\{\begin{array}{l}\frac{a}{\Delta t^n}[M]^n\left[\left(\vec{T}_{cell}^{(0),n-1/2}\right)^4+\left(\vec{T}_z^{(0),n}\right)^4-\left(\vec{T}_z^{(0),n-1/2}\right)^4\right]- \\ \sum_{g=1}^{G}\frac{C_v}{\Delta t^n}[A]\left(\vec{T}_{cell}^{(0),n}\right)\end{array}\right\}_{n=0}= \qquad \text{(A.140)}$$

$$\frac{a}{\Delta t^n}[M]^n\left(\vec{T}_{cell}^{(0),i}\right)^4-\sum_{g=1}^{G}\frac{C_v}{\Delta t^n}[A]\left(\vec{T}_{cell}^{(0),i}\right)$$

A.5. PPWL-STLC Asymptotic Analysis in the Diffusion Limit

This section gives a detailed asymptotic analysis of the PPWL-STLC method in the thick diffusion limit. In this analysis, the source in a cell is approximated as PWL in space with each of these bases being linear in time, therefore leading to 2*N* unknowns

per cell. With this basis, the PPWL-STLC scaled transport equation for a cell can be written using the least-squares procedure as given in Eq. (A.141), for $i = 1,\ldots, 2N$.

$$\sum_{m=1}^{M} w_m J_m \sum_{k=1}^{\#\text{ tracks}} A_{k,m} \int_0^{\Delta s_{k,m}} b_i(s) \left[\frac{\partial \psi_{m,g,k}}{\partial s} + \frac{\sigma_{a,g}}{\varepsilon} \psi_{m,g,k} - \frac{\sigma_{a,g}}{\varepsilon} \frac{1}{4\pi} B_{g,k} \right] ds = 0 \quad \text{(A.141)}$$

The temperature for this source approximation will be considered to be linear and discontinuous. Therefore, the temperature at a given spatial point in a time step can be written in terms of the temperatures at the beginning and end of the time step, using the linear basis in time as given in Eq. (A.142),

$$T_j(t) = T_j \alpha_n(t) + T_{j+N} \alpha_{n+1}(t), \qquad j = 1,\ldots, N, \quad \text{(A.142)}$$

where α_n and α_{n+1} represent the linear basis functions in time given in Eq. (A.143).

$$\begin{aligned} \alpha_n(t) &= \frac{t^{n+1} - t}{\Delta t} \\ \alpha_{n+1}(t) &= \frac{t - t^n}{\Delta t} = 1 - \alpha_n(t) \end{aligned} \quad \text{(A.143)}$$

The scaled material energy equation that holds for any spatial point is given in Eq. (A.144).

$$\varepsilon C_v \frac{\partial T_j}{\partial t} = \sum_{g=1}^{G} \frac{\sigma_{a,g}}{\varepsilon} \left[\phi_{g,j} - B_{g,j} \right] + \varepsilon Q_j \quad \text{(A.144)}$$

Proceeding with the asymptotic analysis, it is again found that the leading-order intensity is isotropic and Planckian from the $O(1/\varepsilon)$ terms in the transport equation for a track. Looking at the $O(1)$ terms from the transport equation for a cell and summing over all groups gives the following terms.

$$\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=1}^{\#\text{ tracks}} A_{k,m} \int_0^{\Delta s_{k,m}} b_i(s)\left[\frac{\partial \psi_{m,g,k}^{(0)}}{\partial s} + \left[\sigma_{a,g}\left(\psi_{m,g,k} - \frac{1}{4\pi} B_{g,k}\right)\right]^{(1)}\right] ds = 0 \qquad \text{(A.145)}$$

The $O(1)$ terms from the material energy equation are given in Eq. (A.146).

$$\sum_{g=1}^{G}\left[\sigma_{a,g}\left(\phi_{g,j} - B_{g,j}\right)\right]^{(1)} = 0 \qquad \text{(A.146)}$$

Since the scalar intensity and the Planckian are written in terms of the basis functions, they are found through the least-squares procedure and the terms in Eq. (A.146) can be written in terms of the weighted summation over tracks and angles. Substituting these terms into Eq. (A.145), the $O(1)$ terms remaining are the following.

$$\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=1}^{\#\text{ tracks}} A_{k,m} \int_0^{\Delta s_{k,m}} b_i(s) \frac{\partial \psi_{m,g,k}^{(0)}}{\partial s} ds = 0 \qquad \text{(A.147)}$$

This term is analyzed for tracks of type 1 since other track types contribute $O(\varepsilon)$ or less. After integration by parts, the following terms are found.

$$\begin{aligned} &\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=type1} A_{k,m} b_i(s) \psi_{m,g,k}^{(0)}(s)\Big|_0^{\Delta s_{k,m}} - \\ &\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=type1} A_{k,m} \int_0^{\Delta s_{k,m}} ds \vec{\nabla} b_i(s) \bullet \vec{\Omega}_m \psi_{m,g,k}^{(0)}(s) = 0 \end{aligned} \qquad \text{(A.148)}$$

As in the previous section, assuming fine track spacing, the error of approximating the analytic integral using the track-based volume as a "quadrature" set in the second term of Eq. (A.148) is at most of $O(\varepsilon)$ and does not need to be considered with these $O(1)$ terms. The analytic integral is zero because of the quadrature sum of Ω. The remaining

term turns into a weighted summation on each spatial edge of the cell at which the i^{th} basis function is nonzero.

$$\sum_{g=1}^{G}\sum_{m=1}^{M} w_m \sum_{k=type1} b_i(s) \sum_{e\in cell} \partial A_{k,m,e} \left(\vec{e}_n \bullet \vec{\Omega}_m\right) \psi_{m,g,k}^{(0)} \Big|_0^{\Delta s_{k,m}} = 0, \qquad i = 1,...,2N \qquad \text{(A.149)}$$

This equation brings in the unknowns from neighboring cells across these edges at which a basis function is nonzero. A similar term was analyzed in section 4.3 (and this applies for this PPWL-STLC method) beginning with Eq. (4.65). This analysis found that assuming the incident partial currents are continuous spatially on the boundary and given fine track spacing, the leading-order unknowns at a vertex are continuous and can be defined at this vertex. The unknowns on the boundary can then be defined in terms of a Marshak boundary condition assuming the incident intensity can be written in terms of the PWL basis as given in Eq. (A.150). Note that this is a 4x4 system because we have unknowns at the beginning and end of the time step for each spatial vertex. This analysis also finds that without fine track spacing, there is loss of continuity of unknowns at a vertex even with continuous incident partial currents on the boundary.

$$\sum_{g=1}^{G} \vec{B}_g^{(0)} = 4\pi \left[M_e\right]^{-1} \sum_{\vec{e}_n \bullet \vec{\Omega}_m < 0} w_m \left|\left(\vec{e}_n \bullet \vec{\Omega}_m\right)\right| \left[C_{m,e}\right] \sum_{g=1}^{G} \vec{\psi}_{m,g,inc}\,, \qquad \text{(A.150)}$$

where,

$$\vec{B}_g^{(0)} = \begin{bmatrix} B_{g,1}^{(0),n} \\ B_{g,2}^{(0),n} \\ B_{g,1}^{(0),n+1} \\ B_{g,2}^{(0),n+1} \end{bmatrix},$$
$$\vec{\psi}_{m,g,inc} = \begin{bmatrix} \psi_{m,g,inc,1}^{n} \\ \psi_{m,g,inc,2}^{n} \\ \psi_{m,g,inc,1}^{n+1} \\ \psi_{m,g,inc,2}^{n+1} \end{bmatrix}. \tag{A.151}$$

Next, the $O(\varepsilon)$ terms are analyzed for this PPWL-STLC method. Before proceeding with this analysis the temperature derivative term from the material energy equation is analyzed when considering an equivalent PPWL-DFEM. Applying a PPWL-DFEM in space-time to the material energy equation would yield the following weighted time derivative term.

$$C_v \int dr \int_{c\Delta t^n} cdtb_i(\vec{r},t) \frac{\partial T}{\partial t} \tag{A.152}$$

This term can be integrated by parts to yield the following expression.

$$C_v \int dr \int_{c\Delta t^n} cdtb_i(\vec{r},t) \frac{\partial T}{\partial t} = C_v \int dr \left\{ cb_i(\vec{r},t) T(\vec{r},t) \Big|_{t^n}^{t^{n+1}} - c \int_{c\Delta t^n} T \frac{\partial b_i}{\partial t} dt \right\} \tag{A.153}$$

This integration can be performed by splitting the basis function between spatial and time components (note this is the same basis used for the PPWL-STLC method). The time derivative of the basis functions is given in Eq. (A.154), where the *sp* superscript indicates the spatial part of the basis function.

$$\frac{\partial b_i}{\partial t} = \begin{cases} -\dfrac{1}{\Delta t^n} b_i^{sp}\left(\vec{r}\right), & i \le N, \\ \dfrac{1}{\Delta t^n} b_i^{sp}\left(\vec{r}\right), & i > N \end{cases} \tag{A.154}$$

Putting this derivative and simplifying Eq. (A.153), we find the following expression.

$$\begin{aligned} & C_v \int dr \int\limits_{c\Delta t^n} cdtb_i\left(\vec{r},t\right)\frac{\partial T}{\partial t} = \\ & C_v c \int dr \left\{ \begin{matrix} b_i^{sp}\left(\vec{r}\right)T^{n+1}\left(\vec{r}\right) - b_i^{sp}\left(\vec{r}\right)T^{n+1/2}\left(\vec{r}\right), & i > N, \\ -b_i^{sp}\left(\vec{r}\right)T^{n}\left(\vec{r}\right) + b_i^{sp}\left(\vec{r}\right)T^{n+1/2}\left(\vec{r}\right), & i \le N \end{matrix} \right\} = \\ & C_v c \int dr \left\{ \begin{matrix} b_i^{sp}\left(\vec{r}\right)\left(T^{n+1}\left(\vec{r}\right) - T^{n+1/2}\left(\vec{r}\right)\right), & i > N, \\ b_i^{sp}\left(\vec{r}\right)\left(T^{n+1/2}\left(\vec{r}\right) - T^{n}\left(\vec{r}\right)\right), & i \le N \end{matrix} \right\} \end{aligned} \tag{A.155}$$

An expression for the time derivative is defined to yield the results of Eq. (A.155).

$$\begin{aligned} & C_v \int dr \int\limits_{c\Delta t^n} cdtb_i\left(\vec{r},t\right)\frac{\partial T}{\partial t} \equiv \\ & C_v \int dr \int\limits_{c\Delta t^n} cdt \left\{ \begin{matrix} b_i\left(\vec{r},t\right)\dfrac{2\left(T^{n+1} - T^{n+1/2}\right)}{\Delta t^n}, & i > N, \\ b_i\left(\vec{r},t\right)\dfrac{2\left(T^{n+1/2} - T^{n}\right)}{\Delta t^n}, & i \le N \end{matrix} \right\} \end{aligned} \tag{A.156}$$

The analog of Eq. (A.156) can be written for the PPWL-STLC method as given in Eq. (A.157).

$$C_v J_m \sum_{k=1}^{\#\,\text{tracks}} A_{k,m} \int_0^{\Delta s_{k,m}} ds b_i(s) \frac{\partial T}{\partial t} \equiv$$

$$C_v J_m \left\{ \begin{array}{ll} \displaystyle\sum_{k=type3}^{\#\text{tracks}} A_{k,m} \int_0^{\Delta s_{k,m}} ds b_i(s) \frac{2\left(T^{n+1/2}(\vec{r}) - T^n(\vec{r})\right)}{\Delta t^n}, & i \le N, \\ \displaystyle\sum_{k=type1}^{\#\text{tracks}} A_{k,m} \int_0^{\Delta s_{k,m}} ds b_i(s) \frac{2\left(T^{n+1}(\vec{r}) - T^{n+1/2}(\vec{r})\right)}{\Delta t^n}, & i > N \end{array} \right\} \tag{A.157}$$

This is the expression used in the following analysis of the $O(\varepsilon)$ terms when considering the time derivative term.

The $O(\varepsilon)$ terms from the transport equation summed over all groups are the following.

$$\begin{aligned} &\sum_{g=1}^{G}\sum_{m=1}^{M} w_m \left[J_m \sum_{k=1}^{\#\,\text{tracks}} A_{k,m} \int_0^{\Delta s_{k,m}} b_i(s) \frac{\partial \psi_{m,g,k}}{\partial s} ds \right]^{(1)} + \\ &\sum_{g=1}^{G}\sum_{m=1}^{M} w_m \left[J_m \sum_{k=1}^{\#\,\text{tracks}} A_{k,m} \int_0^{\Delta s_{k,m}} b_i(s) \left[\sigma_{a,g} \left(\psi_{m,g,k} - \frac{1}{4\pi} B_{g,k} \right) \right] ds \right]^{(2)} = 0 \end{aligned} \tag{A.158}$$

From the material energy equation, the $O(\varepsilon)$ terms are given in the following equation.

$$\left[C_v \frac{\partial T_j(t)}{\partial t} \right]^{(0)} = \sum_{g=1}^{G} \sigma_{a,g} \left[\phi_{g,j}(t) - B_{g,j}(t) \right]^{(2)} + Q_j \tag{A.159}$$

The source, Planckian, and scalar intensity are written in terms of the basis functions and found through the least-squares procedure. These terms are put into Eq. (A.158) to yield the following equations, where the definition for the derivative from Eq. (A.157) is applied.

$$\sum_{g=1}^{G}\sum_{m=1}^{M} w_m \left[J_m \sum_{k=1}^{\#\text{ tracks}} A_{k,m} \int_0^{\Delta s_{k,m}} b_i(s) \frac{\partial \psi_{m,g,k}}{\partial s} ds \right]^{(1)} +$$

$$C_v \sum_{g=1}^{G} \frac{1}{4\pi} \sum_{m=1}^{M} w_m J_m \sum_{k=type3}^{\#\text{tracks}} A_{k,m} \int_0^{\Delta s_{k,m}} ds b_i(s) \frac{2\left(T^{(0),n+1/2}(\vec{r}) - T^{(0),n}(\vec{r})\right)}{\Delta t^n} - \qquad \text{(A.160)}$$

$$\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \frac{1}{4\pi} \sum_{k=1}^{\#\text{ tracks}} A_{k,m} \int_0^{\Delta s_{k,m}} ds b_i(s) Q ds = 0, \qquad i \le N$$

$$\sum_{g=1}^{G}\sum_{m=1}^{M} w_m \left[J_m \sum_{k=1}^{\#\text{ tracks}} A_{k,m} \int_0^{\Delta s_{k,m}} b_i(s) \frac{\partial \psi_{m,g,k}}{\partial s} ds \right]^{(1)} +$$

$$C_v \sum_{g=1}^{G} \frac{1}{4\pi} \sum_{m=1}^{M} w_m J_m \sum_{k=type1}^{\#\text{tracks}} A_{k,m} \int_0^{\Delta s_{k,m}} ds b_i(s) \frac{2\left(T^{(0),n+1}(\vec{r}) - T^{(0),n+1/2}(\vec{r})\right)}{\Delta t^n} - \qquad \text{(A.161)}$$

$$\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \frac{1}{4\pi} \sum_{k=1}^{\#\text{ tracks}} A_{k,m} \int_0^{\Delta s_{k,m}} ds b_i(s) Q ds = 0, \qquad i > N$$

Notation for top and bottom values in a space-time cell is now adopted. The *top* superscript refers to the unknowns (N+1:2N) that are defined at the end of the time step and the *bot* superscript refers to the unknowns (1:N) that are defined at the beginning of the time step. With this notation, the previous two equations become the following expressions.

$$\sum_{g=1}^{G}\sum_{m=1}^{M} w_m \left[J_m \sum_{k=1}^{\#\text{ tracks}} A_{k,m} \int_0^{\Delta s_{k,m}} b_i^{bot}(s) \frac{\partial \psi_{m,g,k}}{\partial s} ds \right]^{(1)} +$$

$$C_v \sum_{g=1}^{G} \frac{1}{4\pi} \sum_{m=1}^{M} w_m J_m \sum_{k=type3}^{\#\text{tracks}} A_{k,m} \int_0^{\Delta s_{k,m}} ds b_i^{bot}(s) \frac{2\left(T^{(0),n+1/2}(\vec{r}) - T^{(0),n}(\vec{r})\right)}{\Delta t^n} - \quad \text{(A.162)}$$

$$\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \frac{1}{4\pi} \sum_{k=1}^{\#\text{ tracks}} A_{k,m} \int_0^{\Delta s_{k,m}} ds b_i^{bot}(s) Q ds = 0, \qquad i \le N$$

$$\sum_{g=1}^{G}\sum_{m=1}^{M} w_m \left[J_m \sum_{k=1}^{\#\text{ tracks}} A_{k,m} \int_0^{\Delta s_{k,m}} b_i^{top}(s) \frac{\partial \psi_{m,g,k}}{\partial s} ds \right]^{(1)} +$$

$$C_v \sum_{g=1}^{G} \frac{1}{4\pi} \sum_{m=1}^{M} w_m J_m \sum_{k=type1}^{\#\text{tracks}} A_{k,m} \int_0^{\Delta s_{k,m}} ds b_i^{top}(s) \frac{2\left(T^{(0),n+1}(\vec{r}) - T^{(0),n+1/2}(\vec{r})\right)}{\Delta t^n} - \quad \text{(A.163)}$$

$$\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \frac{1}{4\pi} \sum_{k=1}^{\#\text{ tracks}} A_{k,m} \int_0^{\Delta s_{k,m}} ds b_i^{top}(s) Q ds = 0, \qquad i > N$$

The source, *Q*, can be written in terms of the basis and top and bottom values. The source term from Eq. (A.162) can be written as the following.

$$\sum_{g=1}^{G} \frac{1}{4\pi} \sum_{m=1}^{M} w_m J_m \sum_{k=1}^{\#\text{ tracks}} A_{k,m} \int_0^{\Delta s_{k,m}} ds b_i^{bot}(s) Q =$$

$$\sum_{g=1}^{G} \frac{1}{4\pi} \sum_{m=1}^{M} w_m J_m \sum_{k=1}^{\#\text{ tracks}} A_{k,m} \int_0^{\Delta s_{k,m}} ds b_i^{bot}(s) \left(\begin{array}{l} \sum_{j=1}^{N} Q_{cell,j}^{bot} b_j^{bot}(s) + \\ \sum_{j=N+1}^{2N} Q_{cell,j}^{top} b_j^{top}(s) \end{array} \right) \quad \text{(A.164)}$$

Separating the spatial and time parts of the basis functions, a spatial mass matrix is defined as given in Eq. (A.165).

$$\left[A_0\right]_{ij} \equiv \frac{1}{4\pi}\sum_{m=1}^{M} w_m J_m \sum_{k=1}^{\#\,\text{tracks}} A_{k,m} \int_0^{\Delta s_{k,m}} ds b_i^{bot,sp} b_j^{bot,sp} = \frac{1}{4\pi}\sum_{m=1}^{M} w_m J_m \sum_{k=1}^{\#\,\text{tracks}} A_{k,m} \int_0^{\Delta s_{k,m}} ds b_i^{top,sp} b_j^{top,sp} \tag{A.165}$$

Because of the large number of tracks with the same spatial trajectory but slightly different times in the space-time cell for a small ε, the integration of the time part of the basis functions can be done separately; therefore, the expression in Eq. (A.164) is written in terms of the spatial mass matrix as given in Eq. (A.166),

$$\begin{aligned}
&\sum_{g=1}^{G}\frac{1}{4\pi}\sum_{m=1}^{M} w_m J_m \sum_{k=1}^{\#\,\text{tracks}} A_{k,m} \int_0^{\Delta s_{k,m}} ds b_i^{bot}(s) Q = \\
&\sum_{g=1}^{G}\frac{1}{4\pi}\sum_{m=1}^{M} w_m J_m \sum_{k=1}^{\#\,\text{tracks}} A_{k,m} \int_0^{\Delta s_{k,m}} ds b_i^{bot}(s) \left(\begin{array}{l} \sum_{j=1}^{N} Q_{cell,j}^{bot} b_j^{bot}(s) + \\ \sum_{j=N+1}^{2N} Q_{cell,j}^{top} b_j^{top}(s) \end{array} \right) = , \\
&\sum_{g=1}^{G} \left\{ \left[A_0\right] \left(\frac{1}{3}\vec{Q}_{cell}^{bot} + \frac{1}{6}\vec{Q}_{cell}^{top} \right) \right\}_i
\end{aligned} \tag{A.166}$$

where the source vectors are defined in Eq. (A.167).

$$\vec{Q}_{cell}^{bot} = \begin{bmatrix} Q_{cell,1}^{bot} \\ \vdots \\ Q_{cell,N}^{bot} \end{bmatrix}, \quad \vec{Q}_{cell}^{top} = \begin{bmatrix} Q_{cell,N+1}^{top} \\ \vdots \\ Q_{cell,2N}^{top} \end{bmatrix} \tag{A.167}$$

The source term in Eq. (A.163) can also be written in terms of this mass matrix. Equation (A.162) and Eq. (A.163) are rewritten as the following expressions.

$$\sum_{g=1}^{G}\sum_{m=1}^{M} w_m \left[J_m \sum_{k=1}^{\#\,\text{tracks}} A_{k,m} \int_0^{\Delta s_{k,m}} b_i^{bot}(s) \frac{\partial \psi_{m,g,k}}{\partial s} ds \right]^{(1)} +$$

$$C_v \sum_{g=1}^{G} \frac{1}{4\pi} \sum_{m=1}^{M} w_m J_m \sum_{k=type3}^{\#\text{tracks}} A_{k,m} \int_0^{\Delta s_{k,m}} ds b_i^{bot}(s) \frac{2\left(T^{(0),n+1/2}(\vec{r}) - T^{(0),n}(\vec{r})\right)}{\Delta t^n} - \quad \text{(A.168)}$$

$$\sum_{g=1}^{G} \left\{ [A_0] \left(\frac{1}{3} \vec{Q}_{cell}^{bot} + \frac{1}{6} \vec{Q}_{cell}^{top} \right) \right\}_i = 0, \qquad i \le N$$

$$\sum_{g=1}^{G}\sum_{m=1}^{M} w_m \left[J_m \sum_{k=1}^{\#\,\text{tracks}} A_{k,m} \int_0^{\Delta s_{k,m}} b_i^{top}(s) \frac{\partial \psi_{m,g,k}}{\partial s} ds \right]^{(1)} +$$

$$C_v \sum_{g=1}^{G} \frac{1}{4\pi} \sum_{m=1}^{M} w_m J_m \sum_{k=type1}^{\#\text{tracks}} A_{k,m} \int_0^{\Delta s_{k,m}} ds b_i^{top}(s) \frac{2\left(T^{(0),n+1}(\vec{r}) - T^{(0),n+1/2}(\vec{r})\right)}{\Delta t^n} - \quad \text{(A.169)}$$

$$\sum_{g=1}^{G} \left\{ [A_0] \left(\frac{1}{6} \vec{Q}_{cell}^{bot} + \frac{1}{3} \vec{Q}_{cell}^{top} \right) \right\}_i = 0, \qquad i > N$$

The leading-order temperature terms in these expressions are analyzed next. Beginning with the terms from Eq. (A.168), the temperatures are written as a basis expansion as given in expression (A.170).

$$\frac{C_v}{\Delta t^n} \sum_{g=1}^{G} \frac{1}{4\pi} \sum_{m=1}^{M} w_m J_m \sum_{k=type3}^{\#\text{tracks}} A_{k,m} \int_0^{\Delta s_{k,m}} ds b_i^{bot} 2\left(T^{(0),n+1/2}(\vec{r}) - T^{(0),n}(\vec{r})\right) =$$

$$\frac{C_v}{\Delta t^n} \sum_{g=1}^{G} \frac{1}{4\pi} \sum_{m=1}^{M} w_m J_m \sum_{k=type3}^{\#\text{tracks}} A_{k,m} \int_0^{\Delta s_{k,m}} ds b_i^{bot} \left(\begin{array}{l} \sum_{j=1}^{N} b_j^{bot,sp} T_{cell,j}^{(0),bot,n+1/2} + \\ \sum_{j=N+1}^{2N} b_j^{top,sp} T_{cell,j}^{(0),top,n+1/2} - \\ 2\sum_{j=1}^{N} b_j^{bot,sp} T_{cell,j}^{(0),top,n} \end{array} \right) \quad \text{(A.170)}$$

Separating the spatial and time part of the basis function and assuming the track summation and integration can be performed, Eq. (A.170) simplifies to the following expression,

$$\begin{aligned}&\frac{C_v}{\Delta t^n}\sum_{g=1}^{G}\frac{1}{4\pi}\sum_{m=1}^{M}w_m J_m \sum_{k=type3}^{\#\text{tracks}} A_{k,m}\int_0^{\Delta s_{k,m}} ds b_i^{bot}\, 2\left(T^{(0),n+1/2}\left(\vec{r}\right)-T^{(0),n}\left(\vec{r}\right)\right)= \\ &\frac{C_v}{\Delta t^n}\sum_{g=1}^{G}\left[M\right]^n\left(\vec{T}_{cell}^{(0),n+1/2}-\vec{T}_{cell}^{(0),n}\right)\end{aligned}, \quad \text{(A.171)}$$

where the temperature vectors are defined in Eq. (A.172) and a spatial-only mass matrix is defined in Eq. (A.173).

$$T_{cell}^{(0),n+1/2}=\begin{bmatrix}T_{cell,1}^{(0),n+1/2}\\ \vdots \\ T_{cell,N}^{(0),n+1/2}\end{bmatrix},\quad T_{cell}^{(0),n}=\begin{bmatrix}T_{cell,N+1}^{(0),n}\\ \vdots \\ T_{cell,2N}^{(0),n}\end{bmatrix} \quad \text{(A.172)}$$

$$\begin{aligned}&\frac{1}{c\Delta t^n}\left[M\right]_{ij}^n \equiv \frac{1}{4\pi}\sum_{m=1}^{M}w_m \sum_{k\in\text{type3}}\partial A_{k,m}^{n+1} b_{i,k}^{bot,sp} b_{j,k}^{bot,sp} = \\ &\frac{1}{4\pi}\sum_{m=1}^{M}w_m \sum_{k\in\text{type3}}\partial A_{k,m}^{n+1} b_{i,k}^{bot,sp} b_{j,k}^{top,sp}\end{aligned} \quad \text{(A.173)}$$

This matrix is very similar to the mass matrix defined in Eq. (A.165), except it is defined for a different set of tracks. It will be assumed that these matrices are approximately equal and only one notation will be used. An analogous equation to Eq. (A.171) can be written for the terms in Eq. (A.169) as given in Eq. (A.174).

$$\begin{aligned}&\frac{C_v}{\Delta t^n}\sum_{g=1}^{G}\frac{1}{4\pi}\sum_{m=1}^{M}w_m J_m \sum_{k=type1}^{\#\text{tracks}} A_{k,m}\int_0^{\Delta s_{k,m}} ds b_i^{top}\, 2\left(T^{(0),n+1}\left(\vec{r}\right)-T^{(0),n+1/2}\left(\vec{r}\right)\right)= \\ &\frac{C_v}{\Delta t^n}\sum_{g=1}^{G}\left\{\left[A_0\right]\left(\vec{T}_{cell}^{(0),n+1}-\vec{T}_{cell}^{(0),n+1/2}\right)\right\}_i\end{aligned} \quad \text{(A.174)}$$

These leading-order temperature expressions are put back into Eq. (A.168) and Eq. (A.169) to yield the following equation.

$$\sum_{g=1}^{G}\sum_{m=1}^{M} w_m \left[J_m \sum_{k=1}^{\#\text{ tracks}} A_{k,m} \int_0^{\Delta s_{k,m}} b_i^{bot}(s) \frac{\partial \psi_{m,g,k}}{\partial s} ds \right]^{(1)} +$$
$$\frac{C_v}{\Delta t^n} \sum_{g=1}^{G} \left\{ [A_0] \left(\vec{T}_{cell}^{(0),n+1/2} - \vec{T}_{cell}^{(0),n} \right) \right\}_i - \tag{A.175}$$
$$\sum_{g=1}^{G} \left\{ [A_0] \left(\frac{1}{3} \vec{Q}_{cell}^{bot} + \frac{1}{6} \vec{Q}_{cell}^{top} \right) \right\}_i = 0, \qquad i \le N$$

$$\sum_{g=1}^{G}\sum_{m=1}^{M} w_m \left[J_m \sum_{k=1}^{\#\text{ tracks}} A_{k,m} \int_0^{\Delta s_{k,m}} b_i^{top}(s) \frac{\partial \psi_{m,g,k}}{\partial s} ds \right]^{(1)} +$$
$$\frac{C_v}{\Delta t^n} \sum_{g=1}^{G} \left\{ [A_0] \left(\vec{T}_{cell}^{(0),n+1} - \vec{T}_{cell}^{(0),n+1/2} \right) \right\}_i - \tag{A.176}$$
$$\sum_{g=1}^{G} \left\{ [A_0] \left(\frac{1}{6} \vec{Q}_{cell}^{bot} + \frac{1}{3} \vec{Q}_{cell}^{top} \right) \right\}_i = 0, \qquad i > N$$

Next, the first-order angular intensity derivative terms are analyzed. The first-order term from Eq. (A.175) is analyzed first. Applying integration by parts yields the following expression.

$$\sum_{g=1}^{G}\sum_{m=1}^{M} w_m \left[J_m \sum_{k=1}^{\#\text{ tracks}} A_{k,m} \int_0^{\Delta s_{k,m}} b_i^{bot}(s) \frac{\partial \psi_{m,g,k}}{\partial s} ds \right]^{(1)} =$$
$$\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \left[\sum_{k=1}^{\#\text{ tracks}} A_{k,m} \left(b_i^{bot}(s) \psi_{m,g,k}(s) \Big|_0^{\Delta s_{k,m}} - \int_0^{\Delta s_{k,m}} ds \frac{\partial b_i^{bot}}{\partial s} \psi_{m,g,k}(s) \right) \right]^{(1)} \tag{A.177}$$

Considering the first term involving the basis-weighted angular intensity in Eq. (A.177), this turns into a summation over the spatial edges of the cell for all tracks that enter or exit on a spatial edge of the cell (this includes tracks of type 1, 3, and 4) plus the contribution of the tracks entering and exiting on the time boundary of the cell (this includes tracks of type 3 and 4). Since the tracks entering and exiting on the time boundary make up an $O(\varepsilon)$ of the total track volume of the cell, they are higher order

than the tracks on the spatial edges and the leading-order intensity must be included with these $O(\varepsilon)$ terms. The same notation for the basis function evaluated at the track position on the spatial edge has been used as in previous analyses.

$$\begin{aligned}
&\sum_{g=1}^{G}\sum_{m=1}^{M} w_m \left[J_m \sum_{k=1}^{\#\text{ tracks}} A_{k,m} \left(b_i^{bot}(s)\psi_{m,g,k}(s)\Big|_0^{\Delta s_{k,m}} \right) \right]^{(1)} = \\
&\sum_{g=1}^{G}\sum_{m=1}^{M} w_m \sum_{e\in cell} \sum_{k=1}^{\#\text{ tracks}} \partial A_{k,m,e} b_{i,k}^{bot} \left(\vec{e}_n \bullet \vec{\Omega}_m \right) \psi_{m,g,k}^{(1)} + \\
&\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=type4} A_{k,m} b_i^{bot}\left(s=\Delta s_k\right)\psi_{m,g,k}^{(0)}\left(s=\Delta s_k\right) - \\
&\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=type3} A_{k,m} b_i^{bot}\left(s=0\right)\psi_{m,g,k}^{(0)}\left(s=0\right)
\end{aligned} \tag{A.178}$$

The spatial edge term in the expression in Eq. (A.178) is a weighted net current for the spatial edges of the cell at which b_i is nonzero and can be defined as the following:

$$\sum_{g=1}^{G}\sum_{e\in cell}\int_{\partial A_e} dA_e b_{i,e}^{bot} J_e^{(1)} \equiv \sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{e\in cell}\sum_{k=1}^{\#\text{ tracks}} \partial A_{k,m,e} b_{i,k}^{bot}\left(\vec{e}_n \bullet \vec{\Omega}_m\right)\psi_{m,g,k}^{(1)}. \tag{A.179}$$

This definition is put back into Eq. (A.177) to yield the following expression.

$$\begin{aligned}
&\sum_{g=1}^{G}\sum_{m=1}^{M} w_m \left[J_m \sum_{k=1}^{\#\text{ tracks}} A_{k,m} \int_0^{\Delta s_{k,m}} b_i^{bot}(s) \frac{\partial \psi_{m,g,k}}{\partial s} ds \right]^{(1)} = \\
&\sum_{g=1}^{G}\sum_{e\in cell}\int_{\partial A_e} dA_e b_{i,e}^{bot} J_e^{(1)} - \sum_{g=1}^{G}\sum_{m=1}^{M} w_m \left[J_m \sum_{k=1}^{\#\text{ tracks}} A_{k,m} \int_0^{\Delta s_{k,m}} ds \frac{\partial b_i^{bot}}{\partial s} \psi_{m,g,k}(s) \right]^{(1)} + \\
&\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=type4} A_{k,m} b_i^{bot}\left(s=\Delta s_k\right)\psi_{m,g,k}^{(0)}\left(s=\Delta s_k\right) - \\
&\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=type3} A_{k,m} b_i^{bot}\left(s=0\right)\psi_{m,g,k}^{(0)}\left(s=0\right)
\end{aligned} \tag{A.180}$$

The remaining first-order angular intensity terms in this equation are split over tracks of type 1 and tracks of type 3 and 4. Also, the derivative of the basis function

with respect to *s* is written in terms of the spatial and time part. The time part of this partial derivative brings in the leading-order intensity. Since tracks of type 3 and 4 make up a higher-order fraction of the total track volume, the leading-order intensity must also be included for these tracks.

$$
\begin{aligned}
&-\sum_{g=1}^{G}\sum_{m=1}^{M} w_m \left[J_m \sum_{k=1}^{\#\text{ tracks}} A_{k,m} \int_0^{\Delta s_{k,m}} ds \frac{\partial b_i^{bot}}{\partial s} \psi_{m,g,k}(s) \right]^{(1)} = \\
&-\sum_{g=1}^{G}\sum_{m=1}^{M} w_m \left[J_m \sum_{k=1}^{\#\text{ tracks}} A_{k,m} \int_0^{\Delta s_{k,m}} ds \left(\vec{\nabla} b_i^{bot} \bullet \vec{\Omega}_m + \frac{\varepsilon}{c} \frac{\partial b_i^{bot}}{\partial t} \right) \psi_{m,g,k}(s) \right]^{(1)} = \\
&-\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=type1} A_{k,m} \int_0^{\Delta s_{k,m}} ds \vec{\nabla} b_i^{bot} \bullet \vec{\Omega}_m \psi_{m,g,k}^{(1)}(s) - \\
&\frac{1}{c}\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=type1} A_{k,m} \int_0^{\Delta s_{k,m}} ds \frac{\partial b_i^{bot}}{\partial t} \psi_{m,g,k}^{(0)}(s) - \\
&\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=type3\&4} A_{k,m} \int_0^{\Delta s_{k,m}} ds \vec{\nabla} b_i^{bot} \bullet \vec{\Omega}_m \psi_{m,g,k}^{(0)}(s)
\end{aligned}
\tag{A.181}
$$

The leading-order intensity is isotropic and Planckian and is written in terms of the basis functions.

$$
\begin{aligned}
&-\sum_{g=1}^{G}\sum_{m=1}^{M} w_m \left[J_m \sum_{k=1}^{\#\text{ tracks}} A_{k,m} \int_0^{\Delta s_{k,m}} ds \vec{\nabla} b_i^{bot} \bullet \vec{\Omega}_m \psi_{m,g,k}(s) \right]^{(1)} = \\
&-\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=type1} A_{k,m} \int_0^{\Delta s_{k,m}} ds \vec{\nabla} b_i^{bot} \bullet \vec{\Omega}_m \psi_{m,g,k}^{(1)}(s) - \\
&\frac{1}{c}\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=type1} A_{k,m} \int_0^{\Delta s_{k,m}} ds \frac{\partial b_i^{bot}}{\partial t} \sum_{j=1}^{2N} b_j(s) B_{g,j}^{(0)} - \\
&\frac{1}{4\pi}\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=type3\&4} A_{k,m} \int_0^{\Delta s_{k,m}} ds \vec{\nabla} b_i^{bot} \bullet \vec{\Omega}_m \sum_{j=1}^{2N} b_j(s) B_{g,j}^{(0)}
\end{aligned}
\tag{A.182}
$$

For tracks of type 3 and 4, the basis functions in the integral over *s* are evaluated at either the top or bottom of the space-time cell. Therefore, only the spatial component of

the basis function is evaluated in this integral and the summation over these tracks will cancel for opposing angles, at least to within an $O(\varepsilon)$ error. The remaining weighted first-order current in Eq. (A.182) will be considered further in this analysis and with the cancellation for type 3 and 4 tracks in Eq. (A.182), Eq. (A.180) becomes the following expression.

$$\begin{aligned}
&\sum_{g=1}^{G}\sum_{m=1}^{M} w_m \left[J_m \sum_{k=1}^{\#\text{ tracks}} A_{k,m} \int_0^{\Delta s_{k,m}} b_i^{bot}(s) \frac{\partial \psi_{m,g,k}}{\partial s} ds \right]^{(1)} = \\
&\sum_{g=1}^{G} \sum_{e \in cell} \int_{\partial A_e} dA_e b_{i,e}^{bot} J_e^{(1)} - \sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=type1} A_{k,m} \int_0^{\Delta s_{k,m}} ds \vec{\nabla} b_i^{bot} \bullet \vec{\Omega}_m \psi_{m,g,k}^{(1)}(s) - \\
&\frac{1}{c}\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=type1} A_{k,m} \int_0^{\Delta s_{k,m}} ds \frac{\partial b_i^{bot}}{\partial t} \sum_{j=1}^{2N} b_j(s) B_{g,j}^{(0)} + \\
&\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=type4} A_{k,m} b_i^{bot}(s=\Delta s_k) \psi_{m,g,k}^{(0)}(s=\Delta s_k) - \\
&\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=type3} A_{k,m} b_i^{bot}(s=0) \psi_{m,g,k}^{(0)}(s=0)
\end{aligned} \tag{A.183}$$

The leading-order terms in this expression are considered next.

The leading-order intensity is isotropic and Planckian and is written in terms of the basis functions, yielding the following expression for the terms involving tracks of type 3 and 4.

$$\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=type4} A_{k,m} b_i^{bot}\left(s=\Delta s_k\right)\psi_{m,g,k}^{(0)}\left(s=\Delta s_k\right) -$$
$$\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=type3} A_{k,m} b_i^{bot}\left(s=0\right)\psi_{m,g,k}^{(0)}\left(s=0\right) =$$
$$\frac{1}{4\pi}\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=type4} A_{k,m}\left(b_i^{bot}\sum_{j=1}^{2N} b_j B_{g,j}^{(0)}\right)_{s=\Delta s_k} -$$
$$\frac{1}{4\pi}\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=type3} A_{k,m}\left(b_i^{bot}\sum_{j=1}^{2N} b_j B_{g,j}^{(0),n}\right)_{s=0} \tag{A.184}$$

The incoming condition from the previous time step is denoted by $B_g^{(0),n}$. The value of b_i^{bot} is zero at the exiting position of type 4 tracks in the expression in Eq. (A.184) for i = 1,…, N and all j. Considering this same expression from Eq. (A.176) that involves b_i^{top}, the exiting position of type 4 tracks go away as well for $i > N$ and j = 1,…, N, because b_j^{bot} is zero at the "top" time boundary. This leaves an expression for the exiting position of type 4 tracks for $i > N$ and $j > N$ as given in Eq. (A.185). This expression involves the spatial part of the basis and Planckian at the end of the time step.

$$\sum_{g=1}^{G}\frac{1}{4\pi}\sum_{m=1}^{M} w_m J_m \sum_{k\in\text{type4}} A_{k,m}\sum_{j=N+1}^{2N}\left(b_i^{top}\left(s\right)b_j^{top}\left(s\right)B_{g,j}^{(0)}\right)_{s=\Delta s_{k,m}} =$$
$$\sum_{g=1}^{G}\frac{1}{4\pi}\sum_{m=1}^{M} w_m \sum_{k\in\text{type4}} \partial A_{k,m}^{n+1}\sum_{j=N+1}^{2N} b_{i,k}^{top,sp} b_{j,k}^{top,sp} B_{g,j}^{(0)} \tag{A.185}$$

The mass matrix previously defined can be used to simplify this expression as follows, where the Planckian at the top of the current space-time cell has been written in vector form given in Eq. (A.187). The top superscript refers to the unknowns (N+1:2N) that are defined at the end of the time step.

$$\sum_{g=1}^{G}\frac{1}{4\pi}\sum_{m=1}^{M}w_m\sum_{k\in\text{type4}}\partial A_{k,m}^{n+1}\sum_{j=N+1}^{2N}b_{i,k}^{top}b_{j,k}^{top}B_{g,j}^{(0)}=$$
$$\frac{1}{c\Delta t^n}\sum_{g=1}^{G}\left\{\left[A_0\right]\vec{B}_{cell,g}^{(0),top,n+1}\right\}_i \tag{A.186}$$

$$\vec{B}_{cell}^{(0),top,n}=\begin{bmatrix}B_{N+1}^{(0),top,n}\\ \vdots\\ B_{2N}^{(0),top,n}\end{bmatrix} \tag{A.187}$$

Similarly, the type 3 tracks entering the time step in Eq. (A.184) cancel for $j > N$ and $i < N$ and for $i > N$ and all j. The type 3 tracks incoming on the bottom boundary of the time step in Eq. (A.184) become the following expression defined for $i = 1,\ldots, N$ and $j = 1,\ldots, N$.

$$\frac{1}{4\pi}\sum_{g=1}^{G}\sum_{m=1}^{M}w_m J_m\sum_{k=type3}A_{k,m}\left(b_i^{bot}\sum_{j=1}^{2N}b_j^{bot}B_{g,j}^{(0),n}\right)_{s=0}=$$
$$\frac{1}{c\Delta t^n}\sum_{g=1}^{G}\left\{\left[A_0\right]\vec{B}_{cell,g}^{(0),top,n}\right\}_i \tag{A.188}$$

Putting together the results of the previous equations, Eq. (A.184) becomes the following expression.

$$\sum_{g=1}^{G}\sum_{m=1}^{M}w_m J_m\sum_{k=type4}A_{k,m}b_i\left(s=\Delta s_k\right)\psi_{m,g,k}^{(0)}\left(s=\Delta s_k\right)-$$
$$\sum_{g=1}^{G}\sum_{m=1}^{M}w_m J_m\sum_{k=type3}A_{k,m}b_i\left(s=0\right)\psi_{m,g,k}^{(0)}\left(s=0\right)= \tag{A.189}$$
$$\left\{\begin{array}{l}-\dfrac{1}{c\Delta t^n}\displaystyle\sum_{g=1}^{G}\left\{\left[A_0\right]\vec{B}_{cell,g}^{(0),top,n}\right\}_i,\quad i\le N,\\ \dfrac{1}{c\Delta t^n}\displaystyle\sum_{g=1}^{G}\left\{\left[A_0\right]\vec{B}_{cell,g}^{(0),top,n+1}\right\}_i,\quad i>N\end{array}\right\}$$

Next, the remaining leading-order term in Eq. (A.183) (along with the analog for the top unknowns from Eq. (A.176)) is analyzed. First, the derivative of the basis function with respect to time is written for the top and bottom values in the cell as done previously.

$$-\frac{1}{c}\sum_{g=1}^{G}\sum_{m=1}^{M}w_m J_m \sum_{k=type1} A_{k,m}\int_0^{\Delta s_{k,m}} ds\,\frac{\partial b_i}{\partial t}\sum_{j=1}^{2N} b_j(s)B_{g,j}^{(0)} =$$

$$\begin{Bmatrix} \frac{1}{c\Delta t^n}\sum_{g=1}^{G}\sum_{m=1}^{M}w_m J_m \sum_{k=type1} A_{k,m}\int_0^{\Delta s_{k,m}} ds b_i^{bot,sp}\sum_{j=1}^{2N} b_j(s)B_{g,j}^{(0)}, & i\le N, \\ -\frac{1}{c\Delta t^n}\sum_{g=1}^{G}\sum_{m=1}^{M}w_m J_m \sum_{k=type1} A_{k,m}\int_0^{\Delta s_{k,m}} ds b_i^{top,sp}\sum_{j=1}^{2N} b_j(s)B_{g,j}^{(0)}, & i>N \end{Bmatrix} \tag{A.190}$$

The Planckian terms are then written as a summation of top and bottom values.

$$-\frac{1}{c}\sum_{g=1}^{G}\sum_{m=1}^{M}w_m J_m \sum_{k=type1} A_{k,m}\int_0^{\Delta s_{k,m}} ds\,\frac{\partial b_i}{\partial t}\sum_{j=1}^{2N} b_j(s)B_{g,j}^{(0)} =$$

$$\begin{Bmatrix} \frac{1}{c\Delta t^n}\sum_{g=1}^{G}\sum_{m=1}^{M}w_m J_m \sum_{k=type1} A_{k,m}\int_0^{\Delta s_{k,m}} ds b_i^{bot,sp}\left[\sum_{j=1}^{N} b_j^{bot}B_{g,j}^{(0),bot} + \sum_{j=N+1}^{2N} b_j^{top}B_{g,j}^{(0),top}\right], & i\le N, \\ -\frac{1}{c\Delta t^n}\sum_{g=1}^{G}\sum_{m=1}^{M}w_m J_m \sum_{k=type1} A_{k,m}\int_0^{\Delta s_{k,m}} ds b_i^{top,sp}\left[\sum_{j=1}^{N} b_j^{bot}B_{g,j}^{(0),bot} + \sum_{j=N+1}^{2N} b_j^{top}B_{g,j}^{(0),top}\right], & i>N \end{Bmatrix} \tag{A.191}$$

This expression is put back into Eq. (A.183) along with the results of Eq. (A.189) to yield an expression for the first-order angular intensity derivative as given below.

$$\sum_{g=1}^{G}\sum_{m=1}^{M} w_m \left[J_m \sum_{k=1}^{\#\text{ tracks}} A_{k,m} \int_0^{\Delta s_{k,m}} b_i(s) \frac{\partial \psi_{m,g,k}}{\partial s} ds \right]^{(1)} =$$

$$\begin{cases} \sum_{g=1}^{G} \sum_{e \in cell} \int_{\partial A_e} dA_e b_{i,e}^{bot} J_e^{(1)} - \\ \sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=type1} A_{k,m} \int_0^{\Delta s_{k,m}} ds \vec{\nabla} b_i^{bot} \bullet \vec{\Omega}_m \psi_{m,g,k}^{(1)}(s) - \\ \frac{1}{c\Delta t^n} \sum_{g=1}^{G} \left\{ [A_0] \vec{B}_{cell,g}^{(0),top,n} \right\}_i + & i \le N, \\ \frac{1}{c\Delta t^n} \sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=type1} A_{k,m} \int_0^{\Delta s_{k,m}} ds b_i^{bot,sp} \left[\sum_{j=1}^{N} b_j^{bot} B_{g,j}^{(0),bot} + \sum_{j=N+1}^{2N} b_j^{top} B_{g,j}^{(0),top} \right], \\ \sum_{g=1}^{G} \sum_{e \in cell} \int_{\partial A_e} dA_e b_{i,e}^{top} J_e^{(1)} - \\ \sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=type1} A_{k,m} \int_0^{\Delta s_{k,m}} ds \vec{\nabla} b_i^{top} \bullet \vec{\Omega}_m \psi_{m,g,k}^{(1)}(s) + \\ \frac{1}{c\Delta t^n} \sum_{g=1}^{G} \left\{ [A_0] \vec{B}_{cell,g}^{(0),top,n+1} \right\}_i - & i > N \\ \frac{1}{c\Delta t^n} \sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=type1} A_{k,m} \int_0^{\Delta s_{k,m}} ds b_i^{top,sp} \left[\sum_{j=1}^{N} b_j^{bot} B_{g,j}^{(0),bot} + \sum_{j=N+1}^{2N} b_j^{top} B_{g,j}^{(0),top} \right], \end{cases} \quad \text{(A.192)}$$

Using this expression, the balance equations of Eq. (A.175) and Eq. (A.176) can be rewritten.

$$\sum_{g=1}^{G}\sum_{e\in cell}\int_{\partial A_e} dA_e b_{i,e}^{bot} J_e^{(1)} -$$

$$\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=type1} A_{k,m} \int_0^{\Delta s_{k,m}} ds \vec{\nabla} b_i^{bot} \bullet \vec{\Omega}_m \psi_{m,g,k}^{(1)}(s) -$$

$$\frac{1}{c\Delta t^n}\sum_{g=1}^{G}\left\{\left[A_0\right]\vec{B}_{cell,g}^{(0),top,n}\right\}_i +$$

$$\frac{1}{c\Delta t^n}\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=type1} A_{k,m} \int_0^{\Delta s_{k,m}} ds b_i^{bot,sp} \left[\sum_{j=1}^{N} b_j^{bot} B_{g,j}^{(0),bot} + \sum_{j=N+1}^{2N} b_j^{top} B_{g,j}^{(0),top}\right] +$$

$$\frac{C_v}{\Delta t^n}\sum_{g=1}^{G}\left\{\left[A_0\right]\left(\vec{T}_{cell}^{(0),n+1/2} - \vec{T}_{cell}^{(0),n}\right)\right\}_i -$$

$$\sum_{g=1}^{G}\left\{\left[A_0\right]\left(\frac{1}{3}\vec{Q}_{cell}^{bot} + \frac{1}{6}\vec{Q}_{cell}^{top}\right)\right\}_i = 0, \qquad i \le N \tag{A.193}$$

$$\sum_{g=1}^{G}\sum_{e\in cell}\int_{\partial A_e} dA_e b_{i,e}^{top} J_e^{(1)} -$$

$$\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=type1} A_{k,m} \int_0^{\Delta s_{k,m}} ds \vec{\nabla} b_i^{top} \bullet \vec{\Omega}_m \psi_{m,g,k}^{(1)}(s) +$$

$$\frac{1}{c\Delta t^n}\sum_{g=1}^{G}\left\{\left[A_0\right]\vec{B}_{cell,g}^{(0),top,n+1}\right\}_i -$$

$$\frac{1}{c\Delta t^n}\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=type1} A_{k,m} \int_0^{\Delta s_{k,m}} ds b_i^{top,sp} \left[\sum_{j=1}^{N} b_j^{bot} B_{g,j}^{(0),bot} + \sum_{j=N+1}^{2N} b_j^{top} B_{g,j}^{(0),top}\right] +$$

$$\frac{C_v}{\Delta t^n}\sum_{g=1}^{G}\left\{\left[A_0\right]\left(\vec{T}_{cell}^{(0),n+1} - \vec{T}_{cell}^{(0),n+1/2}\right)\right\}_i -$$

$$\sum_{g=1}^{G}\left\{\left[A_0\right]\left(\frac{1}{6}\vec{Q}_{cell}^{bot} + \frac{1}{3}\vec{Q}_{cell}^{top}\right)\right\}_i = 0, \qquad i > N \tag{A.194}$$

For ease in dealing with these equations, average values and slope values for the unknowns are defined. In order to do this, Eq. (A.193) is added to Eq. (A.194) to yield

the following equation. It will be assumed for ease of notation that a fine enough track spacing is used with the symmetric quadrature set such that the track area of the top and bottom time boundaries of a cell are the same (up to higher order). Note that due to the use of finite track spacing, the area of the beginning and ending time boundaries of an *x*-*y*-*vt* cell is usually different. This fact is the reason why this method may oscillate around a steady state solution when considering a time-dependent problem in the steady-state limit.

$$\begin{aligned}
&\sum_{g=1}^{G}\sum_{e\in cell}\int_{\partial A_e} dA_e\left(b_{i,e}^{top}+b_{i,e}^{bot}\right)J_e^{(1)}- \\
&\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=type1} A_{k,m}\int_0^{\Delta s_{k,m}} ds\left(\vec{\nabla}b_i^{top}+\vec{\nabla}b_i^{bot}\right)\bullet\vec{\Omega}_m\psi_{m,g,k}^{(1)}\left(s\right)+ \\
&\frac{1}{c\Delta t^n}\sum_{g=1}^{G}\left\{\left[A_0\right]\vec{B}_{cell,g}^{(0),top,n+1}\right\}_i-\frac{1}{c\Delta t^n}\sum_{g=1}^{G}\left\{\left[A_0\right]\vec{B}_{cell,g}^{(0),top,n}\right\}_i+ \\
&\frac{C_v}{\Delta t^n}\sum_{g=1}^{G}\left\{\left[A_0\right]\left(\vec{T}_{cell}^{(0),n+1}-\vec{T}_{cell}^{(0),n}\right)\right\}_i-\sum_{g=1}^{G}\left\{\left[A_0\right]\frac{1}{2}\left(\vec{Q}_{cell}^{bot}+\vec{Q}_{cell}^{top}\right)\right\}_i=0
\end{aligned} \tag{A.195}$$

The Planckian terms in Eq. (A.195) can be written in terms of temperature unknowns and combined. In doing so, the following new variables are defined, where these definitions hold for all unknowns (not just temperature) in the balance equation.

$$\begin{aligned}
\vec{T}^{avg} &= \frac{\vec{T}^{top}+\vec{T}^{bot}}{2} \\
\vec{T}^{slope} &= \frac{\vec{T}^{top}-\vec{T}^{bot}}{2}
\end{aligned} \tag{A.196}$$

Using these variable definitions, Eq. (A.195) is written as given in Eq. (A.197).

$$\sum_{g=1}^{G}\sum_{e\in cell}\int_{\partial A_e} dA_e\left(b_{i,e}^{top}+b_{i,e}^{bot}\right)J_e^{(1)}-$$
$$\sum_{g=1}^{G}\sum_{m=1}^{M}w_m J_m\sum_{k=type1}A_{k,m}\int_0^{\Delta s_{k,m}} ds\left(\vec{\nabla}b_i^{top}+\vec{\nabla}b_i^{bot}\right)\bullet\vec{\Omega}_m\psi_{m,g,k}^{(1)}\left(s\right)+$$
$$\frac{a}{\Delta t^n}\left\{\left[A_0\right]\left[\begin{array}{l}\left(\left(\vec{T}_{cell}^{(0),n+1}\right)^{4,avg}+\left(\vec{T}_{cell}^{(0),n+1}\right)^{4,slope}\right)-\\ \left(\left(\vec{T}_{cell}^{(0),n}\right)^{4,avg}+\left(\vec{T}_{cell}^{(0),n}\right)^{4,slope}\right)\end{array}\right]\right\}_i+ \qquad \text{(A.197)}$$
$$\frac{C_v}{\Delta t^n}\sum_{g=1}^{G}\left\{\left[A_0\right]\left(\begin{array}{l}\left(\vec{T}_{cell}^{(0),avg,n+1}+\vec{T}_{cell}^{(0),slope,n+1}\right)-\\ \left(\vec{T}_{cell}^{(0),avg,n}+\vec{T}_{cell}^{(0),slope,n}\right)\end{array}\right)\right\}_i-\sum_{g=1}^{G}\left\{\left[A_0\right]\left(\vec{Q}_{cell}^{avg}\right)\right\}_i=0$$

Next, Eq. (A.193) is subtracted from Eq. (A.194) to yield the expression given in Eq. (A.198), which also combines terms using the variable definitions of Eq. (A.196).

$$\sum_{g=1}^{G}\sum_{e\in cell}\int_{\partial A_e} dA_e\left(b_{i,e}^{top}-b_{i,e}^{bot}\right)J_e^{(1)}-$$

$$\sum_{g=1}^{G}\sum_{m=1}^{M}w_m J_m\sum_{k=type1}A_{k,m}\int_0^{\Delta s_{k,m}} ds\left(\vec{\nabla}b_i^{top}-\vec{\nabla}b_i^{bot}\right)\bullet\vec{\Omega}_m\psi_{m,g,k}^{(1)}(s)+$$

$$\frac{a}{\Delta t^n}\left\{[A_0]\left[\begin{array}{l}\left(\left(\vec{T}_{cell}^{(0),n+1}\right)^{4,avg}+\left(\vec{T}_{cell}^{(0),n+1}\right)^{4,slope}\right)+\\ \left(\left(\vec{T}_{cell}^{(0),n}\right)^{4,avg}+\left(\vec{T}_{cell}^{(0),n}\right)^{4,slope}\right)\end{array}\right]\right\}_i-$$

$$\frac{2}{c\Delta t^n}\sum_{g=1}^{G}\sum_{m=1}^{M}w_m J_m\sum_{k=type1}A_{k,m}\int_0^{\Delta s_{k,m}} ds\left(b_{i,spatial}^{bot}+b_{i,spatial}^{top}\right)\left[\begin{array}{l}\sum_{j=1}^{N}b_j^{bot}B_{g,j}^{(0),bot}+\\ \sum_{j=N+1}^{2N}b_j^{top}B_{g,j}^{(0),top}\end{array}\right]+$$

$$\frac{C_v}{\Delta t^n}\sum_{g=1}^{G}\left\{[A_0]\left(\begin{array}{l}\left(\vec{T}_{cell}^{(0),avg,n+1}+\vec{T}_{cell}^{(0),slope,n+1}\right)+\\ \left(\vec{T}_{cell}^{(0),avg,n}+\vec{T}_{cell}^{(0),slope,n}\right)-2\vec{T}_{cell}^{(0),avg}\end{array}\right)\right\}_i+\sum_{g=1}^{G}\left\{[A_0]\frac{1}{3}\vec{Q}_{cell}^{slope}\right\}_i=0 \quad \text{(A.198)}$$

The leading-order Planckian terms are combined and written in vector notation. The integrals and summations in this term are performed and the leading-order Planckian is simplified in Eq. (A.198) to give the following expression.

$$-\frac{2}{c\Delta t^n}\sum_{g=1}^{G}\sum_{m=1}^{M}w_m J_m\sum_{k=type1}A_{k,m}\int_0^{\Delta s_{k,m}} ds\left(b_{i,spatial}^{bot}+b_{i,spatial}^{top}\right)\left[\begin{array}{l}\sum_{j=1}^{N}b_j^{bot}B_{g,j}^{(0),bot}+\\ \sum_{j=N+1}^{2N}b_j^{top}B_{g,j}^{(0),top}\end{array}\right]= \quad \text{(A.199)}$$

$$-\frac{2a}{\Delta t^n}\left\{[A_0]\left(\vec{T}_{cell}^{(0)}\right)^{4,avg}\right\}_i$$

The first-order edge current in Eq. (A.197) and Eq. (A.198) is written in terms of slope and average values. Returning to the definition of the edge current given in Eq. (A.179), the first-order angular intensity in this expression is written in terms of the basis functions.

$$\sum_{g=1}^{G}\sum_{e\in cell}\int_{\partial A_e} dA_e b_{i,e} J_e^{(1)} \equiv \sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{e\in cell}\sum_{k=1}^{\#\text{ tracks}} \partial A_{k,m,e} b_{i,k} \left(\vec{e}_n \bullet \vec{\Omega}_m\right) \sum_{j=1}^{2N} b_j \psi_{m,g,j}^{(1)} \quad \text{(A.200)}$$

The summation over *j* is then split into top and bottom unknowns as given in Eq. (A.201).

$$\begin{aligned} &\sum_{g=1}^{G}\sum_{e\in cell}\int_{\partial A_e} dA_e b_{i,e} J_e^{(1)} \equiv \\ &\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{e\in cell}\sum_{k=1}^{\#\text{ tracks}} \partial A_{k,m,e} b_{i,k} \left(\vec{e}_n \bullet \vec{\Omega}_m\right) \left(\begin{array}{l} \sum_{j=1}^{N} b_j^{bot} \psi_{m,g,j}^{(1),bot} + \\ \sum_{j=N+1}^{2N} b_j^{top} \psi_{m,g,j}^{(1),top} \end{array} \right) \end{aligned} \quad \text{(A.201)}$$

Using the slope and average definitions in Eq. (A.196), the top and bottom terms are rewritten as given in Eq. (A.202).

$$\begin{aligned} &\sum_{g=1}^{G}\sum_{e\in cell}\int_{\partial A_e} dA_e b_{i,e} J_e^{(1)} \equiv \\ &\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{e\in cell}\sum_{k=1}^{\#\text{ tracks}} \partial A_{k,m,e} b_{i,k} \left(\vec{e}_n \bullet \vec{\Omega}_m\right) \left(\begin{array}{l} \sum_{j=1}^{N} b_j^{bot} \left(\psi_{m,g,j}^{(1),avg} - \psi_{m,g,j}^{(1),slope}\right) + \\ \sum_{j=N+1}^{2N} b_j^{top} \left(\psi_{m,g,j}^{(1),avg} + \psi_{m,g,j}^{(1),slope}\right) \end{array} \right) \end{aligned} \quad \text{(A.202)}$$

Splitting the j^{th} basis function into its spatial and time parts, the average and slope terms are combined.

$$\begin{aligned} &\sum_{g=1}^{G}\sum_{e\in cell}\int_{\partial A_e} dA_e b_{i,e} J_e^{(1)} \equiv \\ &\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{e\in cell}\sum_{k=1}^{\#\text{ tracks}} \partial A_{k,m,e} b_{i,k} \left(\vec{e}_n \bullet \vec{\Omega}_m\right) \sum_{j=1}^{N} b_{j,k}^{spatial} \left(\begin{array}{l} \psi_{m,g,j}^{(1),avg} + \\ \psi_{m,g,j}^{(1),slope} 2\left(\frac{t - t^{n+1/2}}{\Delta t^n}\right) \end{array} \right) \end{aligned} \quad \text{(A.203)}$$

Next, the edge current terms from the balance equations involving the summation and difference of top and bottom values in Eqs. (A.197) and (A.198) are written using this form.

$$\sum_{g=1}^{G}\sum_{e\in cell}\int_{\partial A_e} dA_e\left(b_{i,e}^{top}+b_{i,e}^{bot}\right)J_e^{(1)} \equiv$$

$$\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{e\in cell}\sum_{k=1}^{\#\text{ tracks}} \partial A_{k,m,e}\begin{pmatrix} b_{i,k}^{top}+ \\ b_{i,k}^{bot}\end{pmatrix}\left(\vec{e}_n\bullet\vec{\Omega}_m\right)\sum_{j=1}^{N} b_{j,k}^{spatial}\begin{pmatrix} \psi_{m,g,j}^{(1),avg}+ \\ \psi_{m,g,j}^{(1),slope}\dfrac{2\left(t-t^{n+1/2}\right)}{\Delta t^n}\end{pmatrix} \quad \text{(A.204)}$$

$$\sum_{g=1}^{G}\sum_{e\in cell}\int_{\partial A_e} dA_e\left(b_{i,e}^{top}-b_{i,e}^{bot}\right)J_e^{(1)} \equiv$$

$$\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{e\in cell}\sum_{k=1}^{\#\text{ tracks}} \partial A_{k,m,e}\begin{pmatrix} b_{i,k}^{top}- \\ b_{i,k}^{bot}\end{pmatrix}\left(\vec{e}_n\bullet\vec{\Omega}_m\right)\sum_{j=1}^{N} b_{j,k}^{spatial}\begin{pmatrix} \psi_{m,g,j}^{(1),avg}+ \\ \psi_{m,g,j}^{(1),slope}\dfrac{2\left(t-t^{n+1/2}\right)}{\Delta t^n}\end{pmatrix} \quad \text{(A.205)}$$

Again splitting the basis into spatial and time terms, the following expressions are found.

$$\sum_{g=1}^{G}\sum_{e\in cell}\int_{\partial A_e} dA_e\left(b_{i,e}^{top}+b_{i,e}^{bot}\right)J_e^{(1)} \equiv$$

$$\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{e\in cell}\sum_{k=1}^{\#\text{ tracks}} \partial A_{k,m,e} b_{i,k}^{spatial}\left(\vec{e}_n\bullet\vec{\Omega}_m\right)\sum_{j=1}^{N} b_{j,k}^{spatial}\psi_{m,g,j}^{(1),avg}+ \quad \text{(A.206)}$$

$$\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{e\in cell}\sum_{k=1}^{\#\text{ tracks}} \partial A_{k,m,e} b_{i,k}^{spatial}\left(\vec{e}_n\bullet\vec{\Omega}_m\right)\sum_{j=1}^{N} b_{j,k}^{spatial}\psi_{m,g,j}^{(1),slope}\frac{2\left(t_k-t^{n+1/2}\right)}{\Delta t^n}$$

$$\sum_{g=1}^{G}\sum_{e\in cell}\int_{\partial A_e} dA_e\left(b_{i,e}^{top}-b_{i,e}^{bot}\right)J_e^{(1)} \equiv$$

$$\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{e\in cell}\sum_{k=1}^{\#\text{ tracks}} \partial A_{k,m,e} b_{i,k}^{spatial} \frac{2\left(t_k - t^{n+1/2}\right)}{\Delta t^n}\left(\vec{e}_n \bullet \vec{\Omega}_m\right)\sum_{j=1}^{N} b_{j,k}^{spatial}\psi_{m,g,j}^{(1),avg} + \quad \text{(A.207)}$$

$$\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{e\in cell}\sum_{k=1}^{\#\text{ tracks}} \partial A_{k,m,e} b_{i,k}^{spatial} 4\left(\frac{t_k - t^{n+1/2}}{\Delta t^n}\right)^2\left(\vec{e}_n \bullet \vec{\Omega}_m\right)\sum_{j=1}^{N} b_{j,k}^{spatial}\psi_{m,g,j}^{(1),slope}$$

Looking at Eq. (A.206), the slope term cancels because a summation over the tracks on a spatial edge, given fine track spacing , causes the time term to be zero. Therefore, the expression for the edge current in Eq. (A.206) is redefined as given in Eq. (A.208).

$$\sum_{g=1}^{G}\sum_{e\in cell}\int_{\partial A_e} dA_e\left(b_{i,e}^{top}+b_{i,e}^{bot}\right)\vec{J}_e^{(1),avg} \equiv$$

$$\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{e\in cell}\sum_{k=1}^{\#\text{ tracks}} \partial A_{k,m,e} b_{i,k}^{spatial}\left(\vec{e}_n \bullet \vec{\Omega}_m\right)\sum_{j=1}^{N} b_{j,k}^{spatial}\psi_{m,g,j}^{(1),avg} \quad \text{(A.208)}$$

Looking at Eq. (A.207), the average term cancels because it sums to zero as the slope term did in the previous equation. The summation over all tracks on a spatial edge of the quadratic time term yields a similar expression to Eq. (A.208) with an added factor of 1/3 as defined in Eq. (A.209).

$$\sum_{g=1}^{G}\sum_{e\in cell}\frac{1}{3}\int_{\partial A_e} dA_e\left(b_{i,e}^{top}-b_{i,e}^{bot}\right)\vec{J}_e^{(1),slope} \equiv$$

$$\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{e\in cell}\sum_{k=1}^{\#\text{ tracks}} \partial A_{k,m,e} b_{i,k}^{spatial} 4\left(\frac{t_k - t^{n+1/2}}{\Delta t^n}\right)^2\left(\vec{e}_n \bullet \vec{\Omega}_m\right)\sum_{j=1}^{N} b_{j,k}^{spatial}\psi_{m,g,j}^{(1),slope} \quad \text{(A.209)}$$

The new definitions for the edge currents given in Eqs. (A.208) and (A.209) are put back into the balance equations given in Eq. (A.197) and Eq. (A.198) along with the expression found in Eq. (A.199).

$$\sum_{g=1}^{G}\sum_{e\in cell}\int_{\partial A_e} dA_e\left(b_{i,e}^{top}+b_{i,e}^{bot}\right)\vec{J}_e^{(1),avg}-$$
$$\sum_{g=1}^{G}\sum_{m=1}^{M}w_m J_m\sum_{k=type1}A_{k,m}\int_0^{\Delta s_{k,m}} ds\left(\vec{\nabla}b_i^{top}+\vec{\nabla}b_i^{bot}\right)\bullet\vec{\Omega}_m\psi_{m,g,k}^{(1)}(s)+$$
$$\frac{a}{\Delta t^n}\left\{[A_0]\left[\begin{array}{l}\left(\left(\vec{T}_{cell}^{(0),n+1}\right)^{4,avg}+\left(\vec{T}_{cell}^{(0),n+1}\right)^{4,slope}\right)-\\ \left(\left(\vec{T}_{cell}^{(0),n}\right)^{4,avg}+\left(\vec{T}_{cell}^{(0),n}\right)^{4,slope}\right)\end{array}\right]\right\}_i+ \qquad \text{(A.210)}$$
$$\frac{C_v}{\Delta t^n}\sum_{g=1}^{G}\left\{[A_0]\left(\begin{array}{l}\left(\vec{T}_{cell}^{(0),avg,n+1}+\vec{T}_{cell}^{(0),slope,n+1}\right)-\\ \left(\vec{T}_{cell}^{(0),avg,n}+\vec{T}_{cell}^{(0),slope,n}\right)\end{array}\right)\right\}_i-\sum_{g=1}^{G}\left\{[A_0]\left(\vec{Q}_{cell}^{avg}\right)\right\}_i=0$$

$$\sum_{g=1}^{G}\sum_{e\in cell}\frac{1}{3}\int_{\partial A_e} dA_e\left(b_{i,e}^{top}-b_{i,e}^{bot}\right)\vec{J}_e^{(1),slope}-$$
$$\sum_{g=1}^{G}\sum_{m=1}^{M}w_m J_m\sum_{k=type1}A_{k,m}\int_0^{\Delta s_{k,m}} ds\left(\vec{\nabla}b_i^{top}-\vec{\nabla}b_i^{bot}\right)\bullet\vec{\Omega}_m\psi_{m,g,k}^{(1)}(s)+$$
$$\frac{a}{\Delta t^n}\left\{[A_0]\left[\begin{array}{l}\left(\left(\vec{T}_{cell}^{(0),n+1}\right)^{4,avg}+\left(\vec{T}_{cell}^{(0),n+1}\right)^{4,slope}\right)+\\ \left(\left(\vec{T}_{cell}^{(0),avg,n}\right)^{4}+\left(\vec{T}_{cell}^{(0),slope,n}\right)^{4}\right)\end{array}\right]\right\}_i-\frac{2a}{\Delta t^n}\left\{[A_0]\left(\vec{T}_{cell}^{(0)}\right)^{4,avg}\right\}_i+ \qquad \text{(A.211)}$$
$$\frac{C_v}{\Delta t^n}\sum_{g=1}^{G}\left\{[A_0]\left(\begin{array}{l}\left(\vec{T}_{cell}^{(0),avg,n+1}+\vec{T}_{cell}^{(0),slope,n+1}\right)+\\ \left(\vec{T}_{cell}^{(0),avg,n}+\vec{T}_{cell}^{(0),slope,n}\right)-2\vec{T}_{cell}^{(0),avg}\end{array}\right)\right\}_i+\sum_{g=1}^{G}\left\{[A_0]\frac{1}{3}\vec{Q}_{cell}^{slope}\right\}_i=0$$

These balance equations in Eq. (A.210) and Eq. (A.211) are summed over all of the interior cells that touch the vertex v_i at which the i^{th} basis function is defined. In doing this summation the first-order net average and slope currents cancel for neighboring cells.

$$\sum_{cell\in v_i}\left[\begin{array}{l}
-\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=type1} A_{k,m} \int_0^{\Delta s_{k,m}} ds\left(\vec{\nabla} b_i^{top} + \vec{\nabla} b_i^{bot}\right)\bullet\vec{\Omega}_m \psi_{m,g,k}^{(1)}(s) + \\
\frac{a}{\Delta t^n}\left\{[A_0]\left[\begin{array}{l}\left(\left(\vec{T}_{cell}^{(0),n+1}\right)^{4,avg} + \left(\vec{T}_{cell}^{(0),n+1}\right)^{4,slope}\right) - \\ \left(\left(\vec{T}_{cell}^{(0),n}\right)^{4,avg} + \left(\vec{T}_{cell}^{(0),n}\right)^{4,slope}\right)\end{array}\right]\right\}_i + \\
\frac{C_v}{\Delta t^n}\sum_{g=1}^{G}\left\{[A_0]\left(\begin{array}{l}\left(\vec{T}_{cell}^{(0),avg,n+1} + \vec{T}_{cell}^{(0),slope,n+1}\right) - \\ \left(\vec{T}_{cell}^{(0),avg,n} + \vec{T}_{cell}^{(0),slope,n}\right)\end{array}\right)\right\}_i - \sum_{g=1}^{G}\left\{[A_0]\left(\vec{Q}_{cell}^{avg}\right)\right\}_i
\end{array}\right] = 0 \quad \text{(A.212)}$$

$$\sum_{cells\in v_i}\left[\begin{array}{l}
-\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=type1} A_{k,m} \int_0^{\Delta s_{k,m}} ds\left(\vec{\nabla} b_i^{top} - \vec{\nabla} b_i^{bot}\right)\bullet\vec{\Omega}_m \psi_{m,g,k}^{(1)}(s) + \\
\frac{a}{\Delta t^n}\left\{[A_0]\left[\begin{array}{l}\left(\left(\vec{T}_{cell}^{(0),n+1}\right)^{4,avg} + \left(\vec{T}_{cell}^{(0),n+1}\right)^{4,slope}\right) + \\ \left(\left(\vec{T}_{cell}^{(0),n}\right)^{4,avg} + \left(\vec{T}_{cell}^{(0),n}\right)^{4,slope}\right) - 2\left(\vec{T}_{cell}^{(0)}\right)^{4,avg}\end{array}\right]\right\}_i + \\
\frac{C_v}{\Delta t^n}\sum_{g=1}^{G}\left\{[A_0]\left(\begin{array}{l}\left(\vec{T}_{cell}^{(0),avg,n+1} + \vec{T}_{cell}^{(0),slope,n+1}\right) + \\ \left(\vec{T}_{cell}^{(0),avg,n} + \vec{T}_{cell}^{(0),slope,n}\right) - 2\vec{T}_{cell}^{(0),avg}\end{array}\right)\right\}_i - \\
\sum_{g=1}^{G}\left\{[A_0]\frac{1}{3}\vec{Q}_{cell}^{slope}\right\}_i
\end{array}\right] = 0 \quad \text{(A.213)}$$

An expression is needed for the weighted first-order current term in these two balance equations for an interior cell.

To find this expression for tracks of type 1, the $O(1)$ terms from the transport equation for a track are multiplied by $\Omega_m\bullet\vec{\nabla} b_i$ and then the least-squares summation is applied as well as a summation over groups, leading to the terms in Eq. (A.214).

$$\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=\text{type1}} A_{k,m} \int_0^{\Delta s_{k,m}} ds\vec{\Omega}_m \bullet \vec{\nabla} b_i(s) \left[\begin{array}{l} \dfrac{\partial \psi_{m,g,k}^{(0)}(s)}{\partial s} + \\ \sigma_{a,g}^{(0)} \left(\psi_{m,g,k}^{(1)}(s) - \dfrac{1}{4\pi} B_{g,k}^{(1)}(s) \right) + \\ \sigma_{a,g}^{(1)} \left(\psi_{m,g,k}^{(0)}(s) - \dfrac{1}{4\pi} B_{g,k}^{(0)}(s) \right) \end{array} \right] = 0 \qquad \text{(A.214)}$$

Considering the last two terms of Eq. (A.214), tracks of type 1 are growing like $1/\varepsilon$ in the cell, so the error that occurs in integrating this term over s for all angles is of $O(\varepsilon)$ and will not be included with these $O(1)$ terms. The remaining terms are rearranged and the Planckian is written in terms of the basis functions as given in Eq. (A.215).

$$\begin{aligned} &\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=\text{type1}} A_{k,m} \int_0^{\Delta s_{k,m}} ds\vec{\Omega}_m \bullet \vec{\nabla} b_i \psi_{m,g,k}^{(1)} = \\ &-\sum_{g=1}^{G} \frac{1}{\sigma_{a,g}^{(0)}} \sum_{m=1}^{M} w_m J_m \sum_{k=\text{type1}} A_{k,m} \int_0^{\Delta s_{k,m}} ds\vec{\Omega}_m \bullet \vec{\nabla} b_i(s) \frac{\partial \psi_{m,g,k}^{(0)}(s)}{\partial s} + \\ &+\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=\text{type1}} A_{k,m} \frac{1}{4\pi} \sum_{j=1}^{2N} \int_0^{\Delta s_{k,m}} ds\vec{\Omega}_m \bullet \vec{\nabla} b_i(s) b_j(s) B_{g,cell,j}^{(1)} \end{aligned} \qquad \text{(A.215)}$$

The first-order Planckian term in this equation goes away because, as was done in previous analyses, this track-volume summation over the space-time cell is equivalent to a volume integral over the space-time cell and this term has opposite signs for opposing angles. Therefore, the error in the track summation of this Planckian term is higher order and is not included here.

The remaining terms of Eq. (A.215) are rewritten with the leading-order intensity written in terms of the basis functions as given in Eq. (A.216).

$$\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=\text{type1}} A_{k,m} \int_0^{\Delta s_{k,m}} ds\vec{\Omega}_m \bullet \vec{\nabla} b_i \psi_{m,g,k}^{(1)} =$$
$$-\sum_{g=1}^{G} \frac{1}{\sigma_{a,g}^{(0)}} \sum_{m=1}^{M} w_m J_m \sum_{k=\text{type1}} A_{k,m} \sum_{j=1}^{2N} \frac{1}{4\pi} \int_0^{\Delta s_{k,m}} ds\vec{\nabla} b_i \bullet \vec{\Omega}_m \vec{\Omega}_m \bullet \vec{\nabla} b_j B_{g,cell,j}^{(0)} \quad \text{(A.216)}$$

Also, in order to correspond to the terms in the balance equation, this remaining term is split over a sum and difference between top and bottom values as shown in Eqs. (A.217) and (A.218).

$$\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=\text{type1}} A_{k,m} \int_0^{\Delta s_{k,m}} ds\vec{\Omega}_m \bullet \vec{\nabla} b_i^{top} \psi_{m,g,k}^{(1)} +$$
$$\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=\text{type1}} A_{k,m} \int_0^{\Delta s_{k,m}} ds\vec{\Omega}_m \bullet \vec{\nabla} b_i^{bot} \psi_{m,g,k}^{(1)} =$$
$$-\sum_{g=1}^{G} \frac{1}{\sigma_{a,g}^{(0)}} \sum_{m=1}^{M} w_m J_m \sum_{k=\text{type1}} A_{k,m} \frac{1}{4\pi} \int_0^{\Delta s_{k,m}} ds\vec{\nabla} b_i^{top} \bullet \vec{\Omega}_m \vec{\Omega}_m \bullet \left(\sum_{j=1}^{N} \vec{\nabla} b_j^{bot} B_{g,cell,j}^{(0),bot} + \sum_{j=N+1}^{2N} \vec{\nabla} b_j^{top} B_{g,cell,j}^{(0),top} \right) - \quad \text{(A.217)}$$
$$\sum_{g=1}^{G} \frac{1}{\sigma_{a,g}^{(0)}} \sum_{m=1}^{M} w_m J_m \sum_{k=\text{type1}} A_{k,m} \frac{1}{4\pi} \int_0^{\Delta s_{k,m}} ds\vec{\nabla} b_i^{bot} \bullet \vec{\Omega}_m \vec{\Omega}_m \bullet \left(\sum_{j=1}^{N} \vec{\nabla} b_j^{bot} B_{g,cell,j}^{(0),bot} + \sum_{j=N+1}^{2N} \vec{\nabla} b_j^{top} B_{g,cell,j}^{(0),top} \right)$$

$$\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=\text{type1}} A_{k,m} \int_0^{\Delta s_{k,m}} ds\vec{\Omega}_m \bullet \vec{\nabla} b_i^{top} \psi_{m,g,k}^{(1)} -$$

$$\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=\text{type1}} A_{k,m} \int_0^{\Delta s_{k,m}} ds\vec{\Omega}_m \bullet \vec{\nabla} b_i^{bot} \psi_{m,g,k}^{(1)} =$$

$$-\sum_{g=1}^{G} \frac{1}{\sigma_{a,g}^{(0)}} \sum_{m=1}^{M} w_m J_m \sum_{k=\text{type1}} A_{k,m} \frac{1}{4\pi} \int_0^{\Delta s_{k,m}} ds\vec{\nabla} b_i^{top} \bullet \vec{\Omega}_m \vec{\Omega}_m \bullet \left(\begin{array}{l} \sum_{j=1}^{N} \vec{\nabla} b_j^{bot} B_{g,cell,j}^{(0),bot} + \\ \sum_{j=N+1}^{2N} \vec{\nabla} b_j^{top} B_{g,cell,j}^{(0),top} \end{array} \right) + \quad \text{(A.218)}$$

$$\sum_{g=1}^{G} \frac{1}{\sigma_{a,g}^{(0)}} \sum_{m=1}^{M} w_m J_m \sum_{k=\text{type1}} A_{k,m} \frac{1}{4\pi} \int_0^{\Delta s_{k,m}} ds\vec{\nabla} b_i^{bot} \bullet \vec{\Omega}_m \vec{\Omega}_m \bullet \left(\begin{array}{l} \sum_{j=1}^{N} \vec{\nabla} b_j^{bot} B_{g,cell,j}^{(0),bot} + \\ \sum_{j=N+1}^{2N} \vec{\nabla} b_j^{top} B_{g,cell,j}^{(0),top} \end{array} \right)$$

These expressions are simplified and the Planckian terms are written in terms of an average and slope as given in Eqs. (A.219) and (A.220).

$$\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=\text{type1}} A_{k,m} \int_0^{\Delta s_{k,m}} ds \left(\vec{\Omega}_m \bullet \vec{\nabla} b_i^{top} \psi_{m,g,k}^{(1)} + \vec{\Omega}_m \bullet \vec{\nabla} b_i^{bot} \psi_{m,g,k}^{(1)} \right) =$$

$$-\sum_{g=1}^{G} \frac{1}{\sigma_{a,g}^{(0)}} \sum_{m=1}^{M} w_m J_m \sum_{k=\text{type1}} A_{k,m} \frac{1}{4\pi} \int_0^{\Delta s_{k,m}} ds \left(\vec{\nabla} b_i^{top} + \vec{\nabla} b_i^{bot} \right) \bullet \vec{\Omega}_m \vec{\Omega}_m \bullet \Rightarrow \quad \text{(A.219)}$$

$$\left(\begin{array}{l} \sum_{j=1}^{N} \vec{\nabla} b_j^{bot} \left(B_{g,cell,j}^{(0),avg} - B_{g,cell,j}^{(0),slope} \right) + \\ \sum_{j=N+1}^{2N} \vec{\nabla} b_j^{top} \left(B_{g,cell,j}^{(0),avg} + B_{g,cell,j}^{(0),slope} \right) \end{array} \right)$$

$$\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=\text{type1}} A_{k,m} \int_0^{\Delta s_{k,m}} ds\left(\vec{\Omega}_m \bullet \vec{\nabla} b_i^{top} \psi_{m,g,k}^{(1)} - \vec{\Omega}_m \bullet \vec{\nabla} b_i^{bot} \psi_{m,g,k}^{(1)}\right) =$$

$$-\sum_{g=1}^{G} \frac{1}{\sigma_{a,g}^{(0)}} \sum_{m=1}^{M} w_m J_m \sum_{k=\text{type1}} A_{k,m} \frac{1}{4\pi} \int_0^{\Delta s_{k,m}} ds\left(\vec{\nabla} b_i^{top} - \vec{\nabla} b_i^{bot}\right) \bullet \vec{\Omega}_m \vec{\Omega}_m \bullet \Rightarrow \qquad \text{(A.220)}$$

$$\left(\begin{array}{l} \sum_{j=1}^{N} \vec{\nabla} b_j^{bot} \left(B_{g,cell,j}^{(0),avg} - B_{g,cell,j}^{(0),slope}\right) + \\ \sum_{j=N+1}^{2N} \vec{\nabla} b_j^{top} \left(B_{g,cell,j}^{(0),avg} + B_{g,cell,j}^{(0),slope}\right) \end{array}\right)$$

Next, the summation and difference between the gradient of the basis functions is evaluated and the previous two equations are rewritten as given in Eq. (A.221) and Eq. (A.222). Note that the time part of the basis functions has been separated from the spatial part in these equations with the spatial part represented by a *sp* superscript.

$$\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=\text{type1}} A_{k,m} \int_0^{\Delta s_{k,m}} ds\left(\vec{\Omega}_m \bullet \vec{\nabla} b_i^{top} \psi_{m,g,k}^{(1)} + \vec{\Omega}_m \bullet \vec{\nabla} b_i^{bot} \psi_{m,g,k}^{(1)}\right) =$$

$$-\sum_{g=1}^{G} \frac{1}{\sigma_{a,g}^{(0)}} \sum_{m=1}^{M} w_m J_m \sum_{k=\text{type1}} A_{k,m} \frac{1}{4\pi} \int_0^{\Delta s_{k,m}} ds \vec{\nabla} b_i^{sp} \bullet \vec{\Omega}_m \vec{\Omega}_m \bullet \Rightarrow \qquad \text{(A.221)}$$

$$\left(\sum_{j=1}^{N} \vec{\nabla} b_j^{sp} \left(B_{g,cell,j}^{(0),avg} + B_{g,cell,j}^{(0),slope} 2\left(\frac{t - t^{n+1/2}}{\Delta t^n}\right)\right)\right)$$

$$\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=\text{type1}} A_{k,m} \int_0^{\Delta s_{k,m}} ds\left(\vec{\Omega}_m \bullet \vec{\nabla} b_i^{top} \psi_{m,g,k}^{(1)} - \vec{\Omega}_m \bullet \vec{\nabla} b_i^{bot} \psi_{m,g,k}^{(1)}\right) =$$

$$-\sum_{g=1}^{G} \frac{1}{\sigma_{a,g}^{(0)}} \sum_{m=1}^{M} w_m J_m \sum_{k=\text{type1}} A_{k,m} \frac{1}{4\pi} \int_0^{\Delta s_{k,m}} ds 2\left(\frac{t - t^{n+1/2}}{\Delta t^n}\right) \vec{\nabla} b_i^{sp} \bullet \vec{\Omega}_m \vec{\Omega}_m \bullet \Rightarrow \qquad \text{(A.222)}$$

$$\left(\sum_{j=1}^{N} \vec{\nabla} b_j^{sp} \left(B_{g,cell,j}^{(0),avg} + B_{g,cell,j}^{(0),slope} 2\left(\frac{t - t^{n+1/2}}{\Delta t^n}\right)\right)\right)$$

First, the terms from Eq. (A.221) are separated and analyzed. This expression is split into the following two summations.

$$\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=\text{type1}} A_{k,m} \int_0^{\Delta s_{k,m}} ds \left(\vec{\Omega}_m \bullet \vec{\nabla} b_i^{top} \psi_{m,g,k}^{(1)} + \vec{\Omega}_m \bullet \vec{\nabla} b_i^{bot} \psi_{m,g,k}^{(1)} \right) =$$

$$-\sum_{g=1}^{G} \frac{1}{\sigma_{a,g}^{(0)}} \sum_{m=1}^{M} w_m J_m \sum_{k=\text{type1}} A_{k,m} \frac{1}{4\pi} \int_0^{\Delta s_{k,m}} ds \vec{\nabla} b_i^{sp} \bullet \vec{\Omega}_m \vec{\Omega}_m \bullet \sum_{j=1}^{N} \vec{\nabla} b_j^{sp} B_{g,cell,j}^{(0),avg} - \qquad \text{(A.223)}$$

$$\sum_{g=1}^{G} \frac{1}{\sigma_{a,g}^{(0)}} \sum_{m=1}^{M} w_m J_m \sum_{k=\text{type1}} A_{k,m} \left[\begin{array}{l} \frac{1}{4\pi} \vec{\nabla} b_i^{sp} \bullet \vec{\Omega}_m \vec{\Omega}_m \bullet \sum_{j=1}^{N} \vec{\nabla} b_j^{sp} B_{g,cell,j}^{(0),slope} \ldots \\ \int_0^{\Delta s_{k,m}} ds \frac{2\left(t - t^{n+1/2}\right)}{\Delta t^n} \end{array} \right]$$

Looking at the first term of this previous expression, this summation is split over the sides of the cell, as given in Eq. (A.224).

$$\begin{aligned} &-\sum_{g=1}^{G} \frac{1}{\sigma_{a,g}^{(0)}} \sum_{m=1}^{M} w_m J_m \sum_{k=\text{type1}} A_{k,m} \frac{1}{4\pi} \int_0^{\Delta s_{k,m}} ds \vec{\nabla} b_i^{sp} \bullet \vec{\Omega}_m \vec{\Omega}_m \bullet \sum_{j=1}^{N} \vec{\nabla} b_j^{sp} B_{g,cell,j}^{(0),avg} = \\ &-\sum_{g=1}^{G} \frac{1}{4\pi} \frac{1}{\sigma_{a,g}^{(0)}} \sum_{l=1}^{N_{sides}} \sum_{j=1}^{N} B_{g,cell,j}^{(0),avg} \sum_{m=1}^{M} w_m J_m \sum_{k=\text{type1}}^{\#\text{tracks on } l} A_{kl,m} \int_0^{\Delta s_{kl,m}} ds \vec{\nabla} b_{i,l}^{sp} \bullet \vec{\Omega}_m \vec{\Omega}_m \bullet \vec{\nabla} b_{j,l}^{sp} \end{aligned} \qquad \text{(A.224)}$$

This expression was analyzed in section 4.3 for these PWL basis functions. The resulting expression uses a defined stiffness matrix and Rosseland mean opacity from this section and is written as given in Eq. (A.225), where the temperature vector has length *N*.

$$\begin{aligned} &-\sum_{g=1}^{G} \frac{1}{\sigma_{a,g}^{(0)}} \sum_{m=1}^{M} w_m J_m \sum_{k=\text{type1}} A_{k,m} \frac{1}{4\pi} \int_0^{\Delta s_{k,m}} ds \vec{\nabla} b_i^{sp} \bullet \vec{\Omega}_m \vec{\Omega}_m \bullet \sum_{j=1}^{N} \vec{\nabla} b_j^{sp} B_{g,cell,j}^{(0),avg} = \\ &-\frac{1}{3} \sum_{l=1}^{\#\text{sides}} \frac{ac}{\sigma_{R,l}^{(0)}} [K]_l \left(\vec{T}_{cell}^{(0),n+1} \right)^{4,avg} \end{aligned} \qquad \text{(A.225)}$$

This expression is put back into Eq. (A.223) to yield an expression for the summation of the weighted first-order current as given in Eq. (A.226).

$$\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=\text{type1}} A_{k,m} \int_0^{\Delta s_{k,m}} ds\left(\vec{\Omega}_m \bullet \vec{\nabla} b_i^{top} \psi_{m,g,k}^{(1)} + \vec{\Omega}_m \bullet \vec{\nabla} b_i^{bot} \psi_{m,g,k}^{(1)}\right) =$$
$$-\frac{1}{3}\sum_{l=1}^{\#\text{sides}} \frac{ac}{\sigma_{R,l}^{(0)}} [K]_l \left(\vec{T}_{cell}^{(0),n+1}\right)^{4,avg} - \qquad \text{(A.226)}$$
$$\sum_{g=1}^{G} \frac{1}{\sigma_{a,g}^{(0)}} \sum_{m=1}^{M} w_m J_m \sum_{k=\text{type1}} \left[A_{k,m} \frac{1}{4\pi} \int_0^{\Delta s_{k,m}} \frac{ds \vec{\nabla} b_i^{sp} \bullet \vec{\Omega}_m \vec{\Omega}_m \bullet \ldots}{\sum_{j=1}^{N} \vec{\nabla} b_j^{sp} B_{g,cell,j}^{(0),slope} \frac{2\left(t - t^{n+1/2}\right)}{\Delta t^n}} \right]$$

Next, the second term in Eq. (A.226) is analyzed. This summation is split over the sides of the cell as was done in Eq. (A.224).

$$\left(-\sum_{g=1}^{G} \frac{1}{\sigma_{a,g}^{(0)}} \sum_{m=1}^{M} w_m J_m \sum_{k=\text{type1}} A_{k,m} \frac{1}{4\pi} \vec{\nabla} b_i^{sp} \bullet \vec{\Omega}_m \vec{\Omega}_m \bullet \ldots \right.$$
$$\left. \sum_{j=1}^{N} \vec{\nabla} b_j^{sp} \int_0^{\Delta s_{k,m}} ds B_{g,cell,j}^{(0),slope} 2\left(\frac{t - t^{n+1/2}}{\Delta t^n}\right) \right) =$$
$$-\left(\frac{1}{4\pi} \frac{1}{\sigma_{a,g}^{(0)}} \sum_{l=1}^{N_{sides}} \sum_{j=1}^{N} B_{g,cell,j}^{(0),slope} \sum_{m=1}^{M} w_m J_m \sum_{k=\text{type1}}^{\#\text{tracks on } l} A_{kl,m} \ldots \right. \qquad \text{(A.227)}$$
$$\left. \sum_{g=1}^{G} \int_0^{\Delta s_{kl,m}} ds 2\left(\frac{t - t^{n+1/2}}{\Delta t^n}\right) \vec{\nabla} b_{i,l}^{sp} \bullet \vec{\Omega}_m \vec{\Omega}_m \bullet \vec{\nabla} b_{j,l}^{sp} \right)$$

It is noted that the time term can be written as was done in Eq. (A.83), therefore part of this expression is higher order and only the following lower order terms remain.

$$\left(-\sum_{g=1}^{G} \frac{1}{\sigma_{a,g}^{(0)}} \sum_{m=1}^{M} \begin{array}{l} w_m J_m \sum_{k=\text{type1}} A_{k,m} \frac{1}{4\pi} \vec{\nabla} b_i^{sp} \bullet \vec{\Omega}_m \vec{\Omega}_m \bullet \ldots \\ \sum_{j=1}^{N} \vec{\nabla} b_j^{sp} \int_0^{\Delta s_{k,m}} ds B_{g,cell,j}^{(0),slope} 2\left(\frac{t - t^{n+1/2}}{\Delta t^n} \right) \end{array} \right) =$$

$$-\left(\begin{array}{l} \frac{1}{4\pi} \frac{1}{\sigma_{a,g}^{(0)}} \sum_{l=1}^{N_{sides}} \sum_{j=1}^{N} B_{g,cell,j}^{(0),slope} \sum_{m=1}^{M} w_m J_m \sum_{k=\text{type1}}^{\#\text{tracks on } l} A_{kl,m} \ldots \\ \sum_{g=1}^{G} \int_0^{\Delta s_{kl,m}} ds 2\left(\frac{t_{0,k} - t^{n+1/2}}{\Delta t^n} \right) \vec{\nabla} b_{i,l}^{sp} \bullet \vec{\Omega}_m \vec{\Omega}_m \bullet \vec{\nabla} b_{j,l}^{sp} \end{array} \right) \tag{A.228}$$

The summation of $t_{0,k} - t^{n+1/2}$ over type 1 tracks on a side for a given angle is equal to zero with fine track spacing so this term vanishes. Therefore, Eq. (A.226) reduces to the following expression.

$$\begin{array}{l} \sum_{g=1}^{G} \sum_{m=1}^{M} w_m J_m \sum_{k=\text{type1}} A_{k,m} \int_0^{\Delta s_{k,m}} ds \left(\vec{\Omega}_m \bullet \vec{\nabla} b_i^{top} \psi_{m,g,k}^{(1)} + \vec{\Omega}_m \bullet \vec{\nabla} b_i^{bot} \psi_{m,g,k}^{(1)} \right) = \\ -\frac{1}{3} \sum_{l=1}^{\#\text{sides}} \frac{ac}{\sigma_{R,l}^{(0)}} \left[K \right]_l \left(\vec{T}_{cell}^{(0),n+1} \right)^{4,avg} \end{array} \tag{A.229}$$

Returning to the difference between top and bottom for this weighted first-order current given in Eq. (A.222), some of the results from the previous analysis can be applied. This expression can again be split into two terms as a summation over the sides of the cell as given in Eq. (A.230) and the higher-order time terms can be dropped.

$$\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=\text{type1}} A_{k,m} \int_0^{\Delta s_{k,m}} ds\left(\vec{\Omega}_m \bullet \vec{\nabla} b_i^{top} \psi_{m,g,k}^{(1)} + \vec{\Omega}_m \bullet \vec{\nabla} b_i^{bot} \psi_{m,g,k}^{(1)}\right) =$$

$$-\left(\frac{1}{4\pi}\sum_{g=1}^{G}\frac{1}{\sigma_{a,g}^{(0)}}\sum_{l=1}^{N_{sides}}\sum_{j=1}^{N} B_{g,cell,j}^{(0),avg}\sum_{m=1}^{M}\left\{\begin{array}{l} w_m J_m \sum_{k=\text{type1}}^{\#\text{ tracks on } l} A_{k,m} 2\left(\frac{t_{0,k}-t^{n+1/2}}{\Delta t^n}\right)\ldots \\ \int_0^{\Delta s_{k,m}} ds \vec{\nabla} b_{i,l}^{sp} \bullet \vec{\Omega}_m \vec{\Omega}_m \bullet \vec{\nabla} b_{j,l}^{sp} \end{array}\right\}\right) - \quad \text{(A.230)}$$

$$\left(\frac{1}{4\pi}\sum_{g=1}^{G}\frac{1}{\sigma_{a,g}^{(0)}}\sum_{l=1}^{N_{sides}}\sum_{j=1}^{N} B_{g,cell,j}^{(0),slope}\sum_{m=1}^{M}\left\{\begin{array}{l} w_m J_m \sum_{k=\text{type1}}^{\#\text{ tracks on } l} A_{k,m} 4\left(\frac{t_{0,k}-t^{n+1/2}}{\Delta t^n}\right)^2\ldots \\ \int_0^{\Delta s_{k,m}} ds \vec{\nabla} b_{i,l}^{sp} \bullet \vec{\Omega}_m \vec{\Omega}_m \bullet \vec{\nabla} b_{j,l}^{sp} \end{array}\right\}\right)$$

From the results of the previous analysis, the term involving the average Planckian will go to zero on each side of the cell. In the remaining slope term, the summation over tracks is split on a side into a summation over sets of "stacked" tracks in time. These are tracks with the same spatial beginning and ending point on the side but different starting and ending points in time. In splitting this summation, one stacked track summation represents an integral over the time step of the space-time cell, and the integral of the quadratic function in time is equal to one third of the time step size.

A different Rosseland mean must be defined for this slope terms since it is weighted by a different gradient of the Planckian. The gradient of this slope term can be written using the Planckian at the end of the time step and the average in the time step as given in Eq. (A.231).

$$\sum_{g=1}^{G}\frac{1}{\sigma_{a,g}^{(0)}}\vec{\nabla}B_{g,l}^{(0),slope}=\sum_{g=1}^{G}\frac{1}{\sigma_{a,g}^{(0)}}\left(\vec{\nabla}B_{g,l}^{(0),n+1}-\vec{\nabla}B_{g,l}^{(0),avg}\right) \tag{A.231}$$

The same Rosseland mean manipulations can be applied to this term as were done in section 4.3 beginning with Eq. (4.83). A new Rosseland mean opacity is defined and used in Eq. (A.232) along with the Rosseland mean used for the gradient of the average Planckian in the previous equations,

$$\sum_{g=1}^{G}\frac{1}{\sigma_{a,g}^{(0)}}\vec{\nabla}B_{g,l}^{(0),slope}=\frac{ac}{\sigma_{R,l}^{(0),n+1}}\vec{\nabla}\left(T^{(0),n+1}\right)^{4}\Big|_{l}-\frac{ac}{\sigma_{R,l}^{(0)}}\left(\vec{\nabla}T^{(0)}\right)^{4,avg}\Big|_{l}, \tag{A.232}$$

where

$$\begin{aligned}\sigma_{R,l}^{(0),n+1}&\equiv\frac{\displaystyle\sum_{g=1}^{G}\frac{1}{\sigma_{a,g}^{(0)}}\frac{\left|\vec{\nabla}B_{g}^{(0),n+1}\right|_{l}}{\left|\vec{\nabla}T^{(0),n+1}\right|_{l}}}{\displaystyle\sum_{g=1}^{G}\frac{\left|\vec{\nabla}B_{g}^{(0),n+1}\right|_{l}}{\left|\vec{\nabla}T^{(0),n+1}\right|_{l}}},\\ \sigma_{R,l}^{(0)}&\equiv\frac{\displaystyle\sum_{g=1}^{G}\frac{1}{\sigma_{a,g}^{(0)}}\frac{\left|\vec{\nabla}B_{g}^{(0),avg}\right|_{l}}{\left|\vec{\nabla}T^{(0),n+1,avg}\right|_{l}}}{\displaystyle\sum_{g=1}^{G}\frac{\left|\vec{\nabla}B_{g}^{(0),avg}\right|_{l}}{\left|\vec{\nabla}T^{(0),avg}\right|_{l}}}\end{aligned} \tag{A.233}$$

Therefore, the expression involving the slope of the Planckian in Eq. (A.230) results in the usual stiffness matrix multiplied by an extra factor and a different Rosseland mean as given in Eq. (A.234).

$$\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=\text{type1}} A_{k,m} \int_0^{\Delta s_{k,m}} ds\left(\vec{\Omega}_m \bullet \vec{\nabla} b_i^{top} \psi_{m,g,k}^{(1)} + \vec{\Omega}_m \bullet \vec{\nabla} b_i^{bot} \psi_{m,g,k}^{(1)}\right) =$$

$$-\left(\frac{1}{4\pi}\sum_{g=1}^{G}\frac{1}{\sigma_{a,g}^{(0)}}\sum_{l=1}^{N_{sides}}\sum_{j=1}^{N} B_{g,cell,j}^{(0),slope}\sum_{m=1}^{M}\left\{\begin{array}{l} w_m J_m \sum_{k=\text{type1}}^{\#\text{ tracks on } l} A_{k,m} 4\left(\frac{t_{0,k}-t^{n+1/2}}{\Delta t^n}\right)^2 \ldots \\ \int_0^{\Delta s_{k,m}} ds \vec{\nabla} b_{i,l}^{sp} \bullet \vec{\Omega}_m \vec{\Omega}_m \bullet \vec{\nabla} b_{j,l}^{sp} \end{array}\right\}\right) = \text{(A.234)}$$

$$-\frac{1}{3}\sum_{l=1}^{\#\text{sides}}\frac{1}{3}\left[K\right]_l\left\{\frac{ac}{\sigma_{R,l}^{(0),n+1}}\left(\vec{T}_{cell}^{(0),n+1}\right)^{4,slope} + \left(\frac{ac}{\sigma_{R,l}^{(0),n+1}} - \frac{ac}{\sigma_{R,l}^{(0)}}\right)\left(\vec{T}_{cell}^{(0),n+1}\right)^{4,avg}\right\}$$

Finally, the expressions for the weighted first-order current from Eq. (A.229) and Eq. (A.234) is put back into the balance equations of Eq. (A.212) and (A.213).

$$\sum_{cell\in v_i}\left[\begin{array}{l} \frac{a}{\Delta t^n}\left[A_0\right]\left[\begin{array}{l}\left(\left(\vec{T}_{cell}^{(0),n+1}\right)^{4,avg} + \left(\vec{T}_{cell}^{(0),n+1}\right)^{4,slope}\right) - \\ \left(\left(\vec{T}_{cell}^{(0),n}\right)^{4,avg} + \left(\vec{T}_{cell}^{(0),n}\right)^{4,slope}\right)\end{array}\right] + \\ \frac{C_v}{\Delta t^n}\sum_{g=1}^{G}\left[A_0\right]\left[\begin{array}{l}\left(\vec{T}_{cell}^{(0),avg,n+1} + \vec{T}_{cell}^{(0),slope,n+1}\right) - \\ \left(\vec{T}_{cell}^{(0),avg,n} + \vec{T}_{cell}^{(0),slope,n}\right)\end{array}\right] - \\ \frac{1}{3}\sum_{l=1}^{\#\text{sides}}\frac{ac}{\sigma_{R,l}^{(0)}}\left[K\right]_l\left(\vec{T}_{cell}^{(0),n+1}\right)^{4,avg} - \sum_{g=1}^{G}\left[A_0\right]\left(\vec{Q}_{cell}^{avg}\right) \end{array}\right] = 0 \qquad \text{(A.235)}$$

$$\sum_{cells\in v_i}\left[\begin{array}{l}
\dfrac{a}{\Delta t^n}\left[A_0\right]\left[\begin{array}{l}\left(\left(\vec{T}_{cell}^{(0),n+1}\right)^{4,avg}+\left(\vec{T}_{cell}^{(0),n+1}\right)^{4,slope}\right)+\\ \left(\left(\vec{T}_{cell}^{(0),n}\right)^{4,avg}+\left(\vec{T}_{cell}^{(0),n}\right)^{4,slope}\right)-2\left(\vec{T}_{cell}^{(0)}\right)^{4,avg}\end{array}\right]+\\
\dfrac{C_v}{\Delta t^n}\sum_{g=1}^{G}\left[A_0\right]\left(\begin{array}{l}\left(\vec{T}_{cell}^{(0),avg,n+1}+\vec{T}_{cell}^{(0),slope,n+1}\right)+\\ \left(\vec{T}_{cell}^{(0),avg,n}+\vec{T}_{cell}^{(0),slope,n}\right)-2\vec{T}_{cell}^{(0),avg}\end{array}\right)-\\
\dfrac{1}{3}\sum_{l=1}^{\#sides}\dfrac{1}{3}\left[K\right]_l\left\{\begin{array}{l}\dfrac{ac}{\sigma_{R,l}^{(0),n+1}}\left(\vec{T}_{cell}^{(0),n+1}\right)^{4,slope}+\\ \left(\dfrac{ac}{\sigma_{R,l}^{(0),n+1}}-\dfrac{ac}{\sigma_{R,l}^{(0)}}\right)\left(\vec{T}_{cell}^{(0),n+1}\right)^{4,avg}\end{array}\right\}+\\
\sum_{g=1}^{G}\left[A_0\right]\dfrac{1}{3}\vec{Q}_{cell}^{slope}
\end{array}\right]=0 \qquad \text{(A.236)}$$

These balance equations give 2*N* equations to solve for the 2*N* leading-order unknowns for this PPWL-STLC method for an interior cell. They represent a discretized diffusion equation similar to an analogous PPWL-DFEM discretized diffusion equation. The leading-order temperature solution satisfies this discretized diffusion equation in the thick diffusion limit, at least for interior cells.

The balance equations given in Eq. (A.235) and Eq. (A.236) are now considered for a cell on the boundary of the problem after the initial time step. When considering the boundary edges of this cell, the current on these edges does not cancel and must be defined. In order to include these terms, the expressions for the weighted first-order currents for tracks of type 1 are returned to for a cell on the boundary. For ease of notation, the weighted first order current will be defined as given in Eqs. (A.237) and (A.238).

$$J_{tracks}^{(1),avg,i} \equiv \sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=\text{type1}} A_{k,m} \int_0^{\Delta s_{k,m}} ds\vec{\Omega}_m \bullet \left(\vec{\nabla} b_i^{top} + \vec{\nabla} b_i^{bot}\right) \psi_{m,g,k}^{(1)} \tag{A.237}$$

$$J_{tracks}^{(1),slope,i} \equiv \sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=\text{type1}} A_{k,m} \int_0^{\Delta s_{k,m}} ds\vec{\Omega}_m \bullet \left(\vec{\nabla} b_i^{top} - \vec{\nabla} b_i^{bot}\right) \psi_{m,g,k}^{(1)} \tag{A.238}$$

Based upon Eq. (A.215), these weighted currents are equal to the following expressions.

$$J_{tracks}^{(1),avg,i} = -\sum_{g=1}^{G} \frac{1}{\sigma_{a,g}^{(0)}} \sum_{m=1}^{M} w_m J_m \sum_{k=\text{type1}} A_{k,m} \int_0^{\Delta s_{k,m}} ds\vec{\Omega}_m \bullet \left(\vec{\nabla} b_i^{top} + \vec{\nabla} b_i^{bot}\right) \frac{\partial \psi_{m,g,k}^{(0)}(s)}{\partial s} \tag{A.239}$$

$$J_{tracks}^{(1),slope,i} = -\sum_{g=1}^{G} \frac{1}{\sigma_{a,g}^{(0)}} \sum_{m=1}^{M} w_m J_m \sum_{k=\text{type1}} A_{k,m} \int_0^{\Delta s_{k,m}} ds\vec{\Omega}_m \bullet \begin{pmatrix} \vec{\nabla} b_i^{top} - \\ \vec{\nabla} b_i^{bot} \end{pmatrix} \frac{\partial \psi_{m,g,k}^{(0)}(s)}{\partial s} \tag{A.240}$$

Next, the track summation is split over the sides of a boundary cell and the leading-order intensity is written in the form given in Eq. (A.137). Equations (A.239) and (A.240) can be rewritten as the following expressions.

$$\begin{aligned} J_{tracks}^{(1),avg,i} = & -\sum_{g=1}^{G} \frac{1}{\sigma_{a,g}^{(0)}} \sum_{m=1}^{M} w_m J_m \sum_{l=1}^{N\text{sides}} \sum_{k=\text{type1}} A_{kl,m} \cdots \\ & \int_0^{\Delta s_{kl,m}} ds\vec{\Omega}_m \bullet \left(\vec{\nabla} b_i^{top} + \vec{\nabla} b_i^{bot}\right) \vec{\Omega}_m \bullet \vec{\nabla} \begin{bmatrix} \psi_{m,g,k}^{(0)}(s) - \\ \frac{B_{g,k}^{(0)}(s)}{4\pi} + \\ \frac{B_{g,k}^{(0)}(s)}{4\pi} \end{bmatrix} \end{aligned} \tag{A.241}$$

$$J_{tracks}^{(1),slope,i} = -\sum_{g=1}^{G} \frac{1}{\sigma_{a,g}^{(0)}} \sum_{m=1}^{M} w_m J_m \sum_{l=1}^{N\text{sides}} \sum_{k=\text{type1}} A_{kl,m} \cdots$$

$$\int_0^{\Delta s_{kl,m}} ds \vec{\Omega}_m \bullet \left(\vec{\nabla} b_i^{top} - \vec{\nabla} b_i^{bot} \right) \vec{\Omega}_m \bullet \vec{\nabla} \begin{bmatrix} \psi_{m,g,k}^{(0)}(s) - \\ \frac{B_{g,k}^{(0)}(s)}{4\pi} + \\ \frac{B_{g,k}^{(0)}(s)}{4\pi} \end{bmatrix} \tag{A.242}$$

For a side of this cell that does not touch the boundary, the leading-order intensity is isotropic and disappears, leaving expressions which yield the stiffness matrix expressions found in Eq. (A.229) and Eq. (A.234). Equations (A.241) and (A.242) turn into the following expressions, where *l* indicates a side of the cell.

$$J_{tracks}^{(1),avg,i} = -\frac{1}{3} \left\{ \sum_{l=1}^{\#\text{sides}} \frac{ac}{\sigma_{R,l}^{(0)}} \left[K \right]_l \left(\vec{T}_{cell}^{(0),n+1} \right)^{4,avg} \right\}_i -$$

$$\sum_{g=1}^{G} \frac{1}{\sigma_{a,g}^{(0)}} \sum_{l \in bdry} \sum_{m=1}^{M} w_m J_m \sum_{k=\text{type1}}^{\#\text{tracks on } l} A_{kl,m} \int_0^{\Delta s_{kl,m}} ds \vec{\Omega}_m \bullet \begin{pmatrix} \vec{\nabla} b_i^{top} + \\ \vec{\nabla} b_i^{bot} \end{pmatrix} \vec{\Omega}_m \bullet \vec{\nabla} \begin{bmatrix} \psi_{m,g,k}^{(0)}(s) - \\ \frac{B_{g,k}^{(0)}(s)}{4\pi} \end{bmatrix} \tag{A.243}$$

$$J_{tracks}^{(1),slope,i} = -\frac{1}{3} \left\{ \sum_{l=1}^{\#\text{sides}} \frac{1}{3} \left[K \right]_l \left\{ \begin{array}{l} \frac{ac}{\sigma_{R,l}^{(0),n+1}} \left(\vec{T}_{cell}^{(0),n+1} \right)^{4,slope} + \\ \left(\frac{ac}{\sigma_{R,l}^{(0),n+1}} - \frac{ac}{\sigma_{R,l}^{(0)}} \right) \left(\vec{T}_{cell}^{(0),n+1} \right)^{4,avg} \end{array} \right\} \right\}_i - \tag{A.244}$$

$$\sum_{g=1}^{G} \frac{1}{\sigma_{a,g}^{(0)}} \sum_{l \in bdry} \sum_{m=1}^{M} w_m J_m \sum_{k=\text{type1}}^{\#\text{tracks on } l} A_{kl,m} \int_0^{\Delta s_{kl,m}} ds \vec{\Omega}_m \bullet \begin{pmatrix} \vec{\nabla} b_i^{top} - \\ \vec{\nabla} b_i^{bot} \end{pmatrix} \vec{\Omega}_m \bullet \vec{\nabla} \begin{bmatrix} \psi_{m,g,k}^{(0)}(s) - \\ \frac{B_{g,k}^{(0)}(s)}{4\pi} \end{bmatrix}$$

The i^{th} basis function in these expressions is not centered at a vertex on this boundary side and the gradient of the basis function is written as a constant dotted with the edge normal (with the basis function split between spatial and time components).

$$J_{tracks}^{(1),avg,i} = --\frac{1}{3}\left\{\sum_{l=1}^{\#\text{sides}} \frac{ac}{\sigma_{R,l}^{(0)}}\left[K\right]_l \left(\vec{T}_{cell}^{(0),n+1}\right)^{4,avg}\right\}_i - \sum_{g=1}^{G}\left\{\begin{array}{l} \sum_{m=1}^{M} w_m J_m \left|\vec{\nabla} b_{il}^{sp}\right| \sum_{k=\text{type1}}^{\#\text{tracks on } l} A_{kl,m}\ldots \\ \frac{1}{\sigma_{a,g}^{(0)}} \sum_{l\in bdry} \int_0^{\Delta s_{kl,m}} ds\left(-\vec{e}_l \bullet \vec{\Omega}_m\right)\vec{\Omega}_m \bullet \vec{\nabla}\left[\begin{array}{l}\psi_{m,g,k}^{(0)}(s) - \\ \frac{B_{g,k}^{(0)}(s)}{4\pi}\end{array}\right]\end{array}\right\} \quad \text{(A.245)}$$

$$J_{tracks}^{(1),slope,i} = -\frac{1}{3}\left\{\sum_{l=1}^{\#\text{sides}} \frac{1}{3}\left[K\right]_l \left\{\begin{array}{l}\frac{ac}{\sigma_{R,l}^{(0),n+1}}\left(\vec{T}_{cell}^{(0),n+1}\right)^{4,slope} + \\ \left(\frac{ac}{\sigma_{R,l}^{(0),n+1}} - \frac{ac}{\sigma_{R,l}^{(0)}}\right)\left(\vec{T}_{cell}^{(0),n+1}\right)^{4,avg}\end{array}\right\}\right\}_i - \left(\begin{array}{l}\sum_{g=1}^{G}\frac{1}{\sigma_{a,g}^{(0)}}\sum_{l\in bdry}\sum_{m=1}^{M} w_m J_m \left|\vec{\nabla} b_{il}^{sp}\right| \sum_{k=\text{type1}}^{\#\text{tracks on } l} A_{kl,m} 2\left(\frac{t_{0,k} - t^{n+1/2}}{\Delta t^n}\right)\ldots \\ \int_0^{\Delta s_{kl,m}} ds\left(-\vec{e}_n \bullet \vec{\Omega}_m\right)\vec{\Omega}_m \bullet \vec{\nabla}\left[\begin{array}{l}\psi_{m,g,k}^{(0)}(s) - \\ \frac{B_{g,k}^{(0)}(s)}{4\pi}\end{array}\right]\end{array}\right) \quad \text{(A.246)}$$

Next, the integral over *s* is performed and the relation given in Eq. (4.55) and Eq. (4.106) is applied to these expressions.

$$J_{tracks}^{(1),avg,i} = -\frac{1}{3}\left\{\sum_{l=1}^{\#\text{sides}} \frac{ac}{\sigma_{R,l}^{(0)}}[K]_l \left(\vec{T}_{cell}^{(0),n+1}\right)^{4,avg}\right\}_i +$$

$$\sum_{g=1}^{G}\frac{1}{\sigma_{a,g}^{(0)}}\sum_{l\in bdry}\sum_{m=1}^{M} w_m \left|\vec{\nabla} b_{il}^{sp}\right| \sum_{k=\text{type1}}^{\#\text{tracks on } l} \partial A_{kle,m} \left|\mu_{m,l}\right| \mu_{m,l} \left[\begin{array}{l} \left(\dfrac{\psi_{m,g,k}^{(0)}(s) - }{\dfrac{B_{g,k}^{(0)}(s)}{4\pi}}\right)_{s=\Delta s_{kl,m}} - \\ \left(\dfrac{\psi_{m,g,k}^{(0)}(s) - }{\dfrac{B_{g,k}^{(0)}(s)}{4\pi}}\right)_{s=0} \end{array}\right] \quad \text{(A.247)}$$

$$J_{tracks}^{(1),slope,i} = -\frac{1}{3}\left\{\sum_{l=1}^{\#\text{sides}} \frac{1}{3}[K]_l \left\{\begin{array}{l} \dfrac{ac}{\sigma_{R,l}^{(0),n+1}}\left(\vec{T}_{cell}^{(0),n+1}\right)^{4,slope} + \\ \left(\dfrac{ac}{\sigma_{R,l}^{(0),n+1}} - \dfrac{ac}{\sigma_{R,l}^{(0)}}\right)\left(\vec{T}_{cell}^{(0),n+1}\right)^{4,avg} \end{array}\right\}\right\}_i +$$

$$\sum_{g=1}^{G}\frac{1}{\sigma_{a,g}^{(0)}}\sum_{l\in bdry}\sum_{m=1}^{M} w_m \left|\vec{\nabla} b_{il}^{sp}\right| \sum_{k=\text{type1}}^{\#\text{tracks on } l} \left\{\begin{array}{l} \partial A_{kle,m} 2\left(\dfrac{t_{0,k} - t^{n+1/2}}{\Delta t^n}\right)\left|\mu_{m,l}\right| \mu_{m,l} \ldots \\ \left[\begin{array}{l} \left(\dfrac{\psi_{m,g,k}^{(0)}(s) - }{\dfrac{B_{g,k}^{(0)}(s)}{4\pi}}\right)_{s=\Delta s_{kl,m}} - \\ \left(\dfrac{\psi_{m,g,k}^{(0)}(s) - }{\dfrac{B_{g,k}^{(0)}(s)}{4\pi}}\right)_{s=0} \end{array}\right] \end{array}\right\} \quad \text{(A.248)}$$

For the tracks of type 1 entering this side from a spatial edge that is not on the boundary, the leading-order intensity is isotropic and the extra terms in the previous equations cancel, leaving the terms on the boundary edge of the boundary side to be analyzed. For tracks exiting on this boundary edge, the outgoing intensity is isotropic and the extra terms in these expressions cancel. Therefore, the only boundary terms remaining in the previous two expressions are for the incoming tracks on the boundary edge. These terms

are given in Eqs. (A.249) and (A.250) where the incident intensity and boundary condition found previously have been substituted.

$$J_{tracks}^{(1),avg,i} = -\frac{1}{3}\left\{\sum_{l=1}^{\#\text{sides}} \frac{ac}{\sigma_{R,l}^{(0)}}\left[K\right]_l \left(\vec{T}_{cell}^{(0),n+1}\right)^{4,avg}\right\}_i + \sum_{g=1}^{G}\frac{1}{3\sigma_{a,g}^{(0)}}\sum_{l\in bdry}\sum_{\mu_{m,l}<0} w_m\left|\vec{\nabla}b_{il}^{sp}\right| \sum_{k=\text{type1}}^{\#\text{tracks on e}_{l,\text{bdry}}} \partial A_{kle,m} 3\mu_{m,l}^2\left[\psi_{m,g,k,l,e,inc} - \frac{B_{g,M,bdry,k}^{(0)}}{4\pi}\right] \tag{A.249}$$

$$J_{tracks}^{(1),slope} = -\frac{1}{3}\left\{\sum_{l=1}^{\#\text{sides}}\frac{1}{3}\left[K\right]_l \left\{\begin{array}{l}\dfrac{ac}{\sigma_{R,l}^{(0),n+1}}\left(\vec{T}_{cell}^{(0),n+1}\right)^{4,slope} + \\ \left(\dfrac{ac}{\sigma_{R,l}^{(0),n+1}} - \dfrac{ac}{\sigma_{R,l}^{(0)}}\right)\left(\vec{T}_{cell}^{(0),n+1}\right)^{4,avg}\end{array}\right\}\right\}_i + \sum_{g=1}^{G}\frac{1}{3\sigma_{a,g}^{(0)}}\sum_{l\in bdry}\sum_{\mu_{m,l}<0} w_m\left|\vec{\nabla}b_{il}^{sp}\right| \sum_{k=\text{type1}}^{\#\text{tracks on } l} \left\{\begin{array}{l}\partial A_{kle,m} 2\left(\dfrac{t_{0,k} - t^{n+1/2}}{\Delta t^n}\right)3\mu_{m,l}^2 \ldots \\ \left[\psi_{m,g,k,l,e,inc} - \dfrac{B_{g,M,bdry,k}^{(0)}}{4\pi}\right]\end{array}\right\} \tag{A.250}$$

The boundary terms in Eq. (A.249) and Eq. (A.250) are not written in a form that allows for interpretation. Some manipulation is required to write the boundary terms in a more usual form.

First, Eq. (A.249) is rewritten with the term that the stiffness matrix came from written in its original form.

$$J_{tracks}^{(1),avg,i} = -\sum_{g=1}^{G}\frac{1}{\sigma_{a,g}^{(0)}}\sum_{m=1}^{M} w_m J_m \sum_{k=\text{type1}} A_{k,m}\int_0^{\Delta s_{k,m}} ds\vec{\Omega}_m \bullet \left(\vec{\nabla}b_i^{top} + \vec{\nabla}b_i^{bot}\right)\frac{\partial \psi_{m,g,k}^{(0)}(s)}{\partial s} + \sum_{g=1}^{G}\frac{1}{3\sigma_{a,g}^{(0)}}\sum_{l\in bdry}\sum_{\mu_m<0} w_m\left|\vec{\nabla}b_{il}^{sp}\right| \sum_{k=\text{type1}}^{\#\text{tracks on e}_{l,\text{bdry}}} \partial A_{kle,m} 3\mu_{m,l}^2\left[\psi_{m,g,k,l,e,inc} - \frac{B_{g,M,bdry,k}^{(0)}}{4\pi}\right] \tag{A.251}$$

The leading-order intensity in the *s* integral is isotropic and is written in terms of the leading-order Planckian.

$$
\begin{aligned}
&J_{tracks}^{(1),avg,i} = \\
&-\sum_{g=1}^{G}\frac{1}{\sigma_{a,g}^{(0)}}\sum_{m=1}^{M}w_m J_m \sum_{k=\text{type1}} A_{k,m}\int_0^{\Delta s_{k,m}} ds\vec{\Omega}_m\bullet\left(\vec{\nabla}b_i^{top}+\vec{\nabla}b_i^{bot}\right)\frac{1}{4\pi}\frac{\partial B_g^{(0)}(s)}{\partial s}+ \\
&\sum_{g=1}^{G}\frac{1}{3\sigma_{a,g}^{(0)}}\sum_{l\in bdry}\sum_{\mu_{m,l}<0} w_m\left|\vec{\nabla}b_{il}^{sp}\right|\sum_{k=\text{type1}}^{\#\text{tracks on e}_{\text{bdry}}}\partial A_{kle,m}3\mu_{m,l}^2\left[\psi_{m,g,k,l,e,inc}-\frac{B_{g,M,bdry,k}^{(0)}}{4\pi}\right]
\end{aligned}
\tag{A.252}
$$

This leading-order intensity is then integrated by parts to yield the following expression.

$$
\begin{aligned}
&J_{tracks}^{(1),avg,i} = \\
&-\sum_{g=1}^{G}\frac{1}{\sigma_{a,g}^{(0)}}\sum_{m=1}^{M}w_m J_m \sum_{k=\text{type1}} A_{k,m}\left\{\begin{array}{l}\left[\vec{\Omega}_m\bullet\left(\vec{\nabla}b_i^{top}+\vec{\nabla}b_i^{bot}\right)\frac{B_g^{(0)}}{4\pi}\right]_0^{\Delta s_{k,m}}- \\ \int_0^{\Delta s_{k,m}} ds\,\frac{B_g^{(0)}}{4\pi}\frac{\partial^2}{\partial s^2}\left(b_i^{top}+b_i^{bot}\right)\end{array}\right\}+ \\
&\sum_{g=1}^{G}\frac{1}{3\sigma_{a,g}^{(0)}}\sum_{l\in bdry}\sum_{\mu_{m,l}<0} w_m\left|\vec{\nabla}b_{il}^{sp}\right|\sum_{k=\text{type1}}^{\#\text{tracks on e}_{\text{bdry}}}\partial A_{kle,m}3\mu_{m,l}^2\left[\psi_{m,g,k,l,e,inc}-\frac{B_{g,M,bdry,k}^{(0)}}{4\pi}\right]
\end{aligned}
\tag{A.253}
$$

The first term in the second line of Eq. (A.253) turns into a summation over the edges of the side so Eq. (4.55) can be applied along with the fact that the gradient of the basis function is a constant on this side (the time part of the top and bottom basis functions adds to unity). The same notation used in section 4.3 for interior and boundary unknowns is used here. The summation over angles is split into incoming and outgoing angles on this side.

$$
\begin{aligned}
&J_{tracks}^{(1),avg,i} = \\
&-\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=1}^{\#\text{tracks}\notin e_{bdry}} A_{k,m} \frac{1}{\sigma_{a,g}^{(0)}} \left\{ \left[\vec{\Omega}_m \bullet \vec{\nabla} b_i \frac{B_{g,\text{int},k}^{(0)}}{4\pi} \right]_0^{\Delta s_{k,m}} - \int_0^{\Delta s_{k,m}} ds \frac{\partial^2 b_i}{\partial s^2} \frac{B_{g,\text{int},k}^{(0)}}{4\pi} \right\} \\
&-\sum_{g=1}^{G} \frac{1}{\sigma_{a,g}^{(0)}} \left| \vec{\nabla} b_{il}^{sp} \right| \left[\begin{array}{l} \sum_{\mu_{m,l}<0} w_m \sum_{k=1}^{\#\text{tracks on } e_{bdry}} \partial A_{kle,m} \left|\mu_{m,l}\right| \mu_{m,l} \left[\frac{B_{g,\text{int},k}^{(0)}}{4\pi} - \frac{B_{g,M,bdry,k}^{(0)}}{4\pi} \right] - \\ \sum_{\mu_{m,l}>0} w_m \sum_{k=1}^{\#\text{tracks on } e_{bdry}} \partial A_{kle,m} \left|\mu_{m,l}\right| \mu_{m,l} \left[\frac{B_{g,M,bdry,k}^{(0)}}{4\pi} - \frac{B_{g,\text{int},k}^{(0)}}{4\pi} \right] \end{array} \right] \\
&+\sum_{g=1}^{G} \frac{1}{3\sigma_{a,g}^{(0)}} \sum_{l\in bdry} \sum_{\mu_{m,l}<0} w_m \left| \vec{\nabla} b_{il}^{sp} \right| \sum_{k=\text{type1}}^{\#\text{tracks on } e_{bdry}} \partial A_{kle,m} 3\mu_{m,l}^2 \left[\begin{array}{l} \psi_{m,g,k,l,e,inc} - \\ \frac{B_{g,M,bdry,k}^{(0)}}{4\pi} \end{array} \right]
\end{aligned}
\tag{A.254}
$$

With a symmetric quadrature set and fine track spacing, assuming the same tracks are used for opposing angles, the incoming and outgoing angles on the boundary edge are combined to give the expression in Eq. (A.255).

$$
\begin{aligned}
&J_{tracks}^{(1),avg,i} = \\
&-\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=1}^{\#\text{tracks}\notin e_{bdry}} A_{k,m} \frac{1}{\sigma_{a,g}^{(0)}} \left\{ \left[\vec{\Omega}_m \bullet \vec{\nabla} b_i \frac{B_{g,\text{int},k}^{(0)}}{4\pi} \right]_0^{\Delta s_{k,m}} - \int_0^{\Delta s_{k,m}} ds \frac{\partial^2 b_i}{\partial s^2} \frac{B_{g,\text{int},k}^{(0)}}{4\pi} \right\} \\
&-\sum_{g=1}^{G} \frac{1}{3\sigma_{a,g}^{(0)}} \sum_{l\in bdry} \left| \vec{\nabla} b_{il}^{sp} \right| \left[\sum_{\mu_{m,l}<0} w_m \sum_{k=1}^{\#\text{tracks on } e_{bdry}} \partial A_{kle,m} 3\mu_{m,l}^2 \left[\begin{array}{l} \frac{2B_{g,\text{int},k}^{(0)}}{4\pi} - \\ \frac{2B_{g,M,bdry,k}^{(0)}}{4\pi} \end{array} \right] \right] \\
&+\sum_{g=1}^{G} \frac{1}{3\sigma_{a,g}^{(0)}} \sum_{l\in bdry} \sum_{\mu_{m,l}<0} w_m \left| \vec{\nabla} b_{il}^{sp} \right| \sum_{k=\text{type1}}^{\#\text{tracks on } e_{bdry}} \partial A_{kle,m} 3\mu_{m,l}^2 \left[\begin{array}{l} \psi_{m,g,k,e,inc} - \\ \frac{B_{g,M,bdry,k}^{(0)}}{4\pi} \end{array} \right]
\end{aligned}
\tag{A.255}
$$

Terms are combined and simplified in this expression to give Eq. (A.256).

$$
\begin{aligned}
&J_{tracks}^{(1),avg,i} = \\
&-\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=1}^{\#\text{tracks}\notin e_{\text{bdry}}} A_{k,m}\frac{1}{\sigma_{a,g}^{(0)}}\left\{\left[\vec{\Omega}_m\bullet\vec{\nabla}b_i\frac{B_{g,\text{int},k}^{(0)}}{4\pi}\right]_0^{\Delta s_{k,m}} - \int_0^{\Delta s_{k,m}} ds\,\frac{\partial^2 b_i}{\partial s^2}\frac{B_{g,\text{int},k}^{(0)}}{4\pi}\right\} \\
&-\sum_{g=1}^{G}\frac{1}{3\sigma_{a,g}^{(0)}}\sum_{l\in bdry}\sum_{\mu_{m,l}<0} w_m\left|\vec{\nabla}b_{il}^{sp}\right| \sum_{k=1}^{\#\text{tracks on } e_{\text{bdry}}} \partial A_{kle,m}\,3\mu_{m,l}^2\frac{2B_{g,\text{int},k}^{(0)}}{4\pi} \\
&+\sum_{g=1}^{G}\frac{1}{3\sigma_{a,g}^{(0)}}\sum_{l\in bdry}\sum_{\mu_{m,l}<0} w_m\left|\vec{\nabla}b_{il}^{sp}\right| \sum_{k=\text{type1}}^{\#\text{tracks on } e_{\text{bdry}}} \partial A_{kle,m}\,3\mu_{m,l}^2\left[\psi_{m,g,k,e,inc}+\frac{B_{g,M,bdry,k}^{(0)}}{4\pi}\right]
\end{aligned}
\tag{A.256}
$$

The same procedure following in section 4.3 to define a "variational" boundary Planckian term can then be followed. This analysis begins with Eq. (4.116). The results of this analysis lead to an expression for the weighted first-order current on a boundary cell as given in Eq. (A.257).

$$
J_{tracks,bdrycell}^{(1),avg,i} = -\frac{1}{3}\left\{\sum_{l=1}^{\#\text{sides}}\frac{ac}{\sigma_{R,l}^{(0)}}\left[K\right]_l\left(\vec{\tilde{T}}_{cell,V}^{(0),n+1}\right)^{4,avg}\right\}_i \tag{A.257}
$$

From the expression in Eq. (A.257), the boundary condition seen by interior cells is different from the usual Marshak boundary condition defined previously.

Returning to Eq. (A.250), the analysis of this term is very similar to the previous analysis for Eq. (A.249). The main difference is that the summation over type 1 tracks can be split into a summation over the sets of tracks in time. Performing the summation of one set of tracks, the time term in Eq. (A.250) will give the first-moment or slope of the intensity. Therefore a weighted slope current term for a boundary cell can be defined similar to Eq. (A.257) as given in Eq. (A.258).

$$J^{(1),slope,i}_{tracks,bdrycell} = -\frac{1}{3}\left\{\sum_{l=1}^{\#sides}\frac{1}{3}[K]_l\left\{\begin{array}{l}\frac{ac}{\sigma^{(0),n+1}_{R,l}}\left(\tilde{\vec{T}}^{(0),n+1}_{cell,V}\right)^{4,slope}+\\ \left(\frac{ac}{\sigma^{(0),n+1}_{R,l}}-\frac{ac}{\sigma^{(0)}_{R,l}}\right)\left(\tilde{\vec{T}}^{(0),n+1}_{cell,V}\right)^{4,avg}\end{array}\right\}\right\}_i \tag{A.258}$$

Overall for a cell on the boundary, the weighted first-order current must be split among sides on the cell. For a side edge that touches a boundary vertex but does not include the problem boundary edge, the Marshak boundary condition must be included. For a side edge that is on the problem boundary, the incident angular intensity and slope of the incident angular intensity must be included. With this first-order current defined for boundary cells, the balance equations given in Eq. (A.235) and Eq. (A.236) hold for all cells in the problem.

Finally, the balance equations given in Eqs. (A.235) and (A.236) must be analyzed for the initial layer of the problem. The analog of Eq. (A.235) for the first time step is given in Eq. (A.259), where it is assumed the initial condition can be written in terms of the basis functions. The superscript *i* denotes the initial condition for the original transport problem.

$$\sum_{cell\in v_i}\left[\begin{array}{l}-\frac{1}{3}\sum_{l=1}^{\#sides}\frac{ac}{\sigma^{(0)}_{R,l}}[K]_l\left(\vec{T}^{(0),n+1}_{cell}\right)^{4,avg}+\\ \frac{a}{\Delta t^n}[A_0]\left(\begin{array}{l}\left(\left(\vec{T}^{(0),n+1}_{cell}\right)^{4,avg}+\left(\vec{T}^{(0),n+1}_{cell}\right)^{4,slope}\right)-\\ \left(\vec{T}^{(0),i}_{cell}\right)^4\end{array}\right)+\\ \sum_{g=1}^{G}\frac{C_v}{\Delta t^n}[A_0]2\left(\Delta\vec{T}^{(0),avg}\right)-\sum_{g=1}^{G}[A_0]\left(\vec{Q}^{avg}_{cell}\right)\end{array}\right]=0 \tag{A.259}$$

Similarly, the analog of Eq. (A.236) for the initial time step is given in Eq. (A.260).

$$\sum_{cells\in v_i}\left[\begin{array}{l}
-\frac{1}{3}\sum_{l=1}^{\#sides}\frac{1}{3}[K]_l\left\{\begin{array}{l}\frac{ac}{\sigma_{R,l}^{(0),n+1}}\left(\vec{T}_{cell}^{(0),n+1}\right)^{4,slope}+\\ \left(\frac{ac}{\sigma_{R,l}^{(0),n+1}}-\frac{ac}{\sigma_{R,l}^{(0)}}\right)\left(\vec{T}_{cell}^{(0),n+1}\right)^{4,avg}\end{array}\right\}+\\
\frac{a}{\Delta t^n}[A_0]\left(\begin{array}{l}\left(\left(\vec{T}_{cell}^{(0),n+1}\right)^{4,avg}+\left(\vec{T}_{cell}^{(0),n+1}\right)^{4,slope}\right)+\\ \left(\vec{T}_{cell}^{(0),i}\right)^4\end{array}\right)-\\
\frac{2a}{\Delta t^n}[A_0]\left(\vec{T}_{cell}^{(0)}\right)^{4,avg}+\\
\sum_{g=1}^{G}\frac{C_v}{\Delta t^n}[A_0]\Delta\vec{T}^{(0),slope}+\sum_{g=1}^{G}[A_0]\frac{1}{3}\vec{Q}_{cell}^{slope}
\end{array}\right]=0 \qquad \text{(A.260)}$$

The general initial condition on each cell must be able to reproduce these two previous equations. If the initial intensity is Planckian at the initial temperature, Eqs. (A.259) and (A.260) would produce the balance equations of Eqs. (A.235) and (A.236). In order for a general initial condition to reproduce these balance equations, the initial condition must satisfy the following expression.

$$\left\{\frac{a}{\Delta t^n}[A_0]\left(\left(\vec{T}_{cell}^{(0),n}\right)^{4,avg}+\left(\vec{T}_{cell}^{(0),n}\right)^{4,slope}\right)\right\}_{n=0}=\frac{a}{\Delta t^n}[A_0]\left(\vec{T}_{cell}^{(0),i}\right)^4 \qquad \text{(A.261)}$$

Overall, it has been found that this PPWL-STLC method produces a set of balance equations given in Eqs. (A.235) and (A.236) for i = 1:2N, that hold for all cells given the boundary current terms defined in Eqs. (A.257) and (A.258) and the initial condition given by Eq. (A.261). These balance equations form a set of equations consistent with an analogous DFEM discretized diffusion equation that can be solved for the 2N temperature unknowns. Treating the time variable as linear for each spatial position should provide an accurate solution for most time-dependent problems in the

thick diffusion limit on arbitrary polygons and polyhedra. This statement does not account for the issue found regarding the continuity of unknowns without fine track spacing.

APPENDIX B

This appendix details the implementation of the LC and STLC methods in the Parallel Deterministic Transport (PDT) code at Texas A&M University as of October 2012. Refer to section 5 of this dissertation for the overview of this implementation. These details assume some knowledge of the PDT code, including the definitions of a buffer, graph edge, and cell edge, which can be found in the PDT manual.

Setting tags in the input file

Starting from the beginning of running a LC problem in PDT, several special parameters are needed in the input file. First, the spatial discretization method (SDM) must be set to the chosen method. This tag is called <spatial_method> and must be set to SDM_LC for using LC only in the spatial dimension or SDM_STLC for using STLC in space and time. This tag is then followed by an <lc_type> which specifies the source approximation. The PWC-LC and PWC-STLC methods are specified by the lc_type called STEP_LC. The PWL-LC and PPWL-STLC methods are specified by the lc_type called LINEAR_LC. Once other source approximations are implemented, these lc_types will need to be updated. The other special parameters needed are in the <quad_info> section of the input file which is in each <energy_set>. This section needs to include the offset of the coordinate systems and the track spacing, all specified by floating point numbers. For a 2D LC problem the following tags need to be specified: <quad_xoffset.fp>, <quad_yoffset.fp>, and <quad_omegaspacing.fp>. These values correspond to the spacing and offset values described in section 2.1. For a 3D LC problem, the <quad_zoffset.fp> and <quad_uspacing.fp> tags must be added to

correspond to the values used in the section 2.4 equations. The STLC method is currently only implemented in 2D and it should include all of the same tags used for the 3D LC method, however the _zoffset tag actually refers to the offset of the *vt* coordinate as outlined in section 3.2.

The boundary conditions for an LC and STLC problem use the same input as all other methods including being specified by major divisions. The details of how the boundary condition is set from this input is outlined in one of the following sections. An STLC problem requires an initial condition in the input. This section begins with the <initial_condition> tag. Next, the type of initial condition must be set using the <initial_type> tag to one of the following: ZERO, ISOTROPIC, LINEAR. If isotropic or linear initial conditions are specified, the <flux_for_e_group> tag is needed followed by the <group_id> for the energy group this flux is associated with. For the isotropic initial condition, the angular flux value is specified for this group by the <flux> flag. For the linear initial condition, the coefficients for this linear initial condition must be given by the following tags, much like the linear boundary condition: A_mu, A_eta, A_x, A_y, A. The A coefficient can take into account if the problem does not begin at an initial time of t = 0. The details of how this input is used to set the initial angular flux for a cell are given in one of the following sections. Also, the initial condition for STLC problems can be set by major division like the boundary condition.

Setting global track min and max values

After the input is read and values are set for the material, cell, and quadrature, the edges and tracks must be initialized. This setup is performed through function calls in

SweepProblem. Before the edges can be setup, the global minimum and maximum track indices across all processors must be determined. This is done by finding the local minimum and maximum indices on each processor and then performing an all-reduce. This method is unique to LC methods and is performed in SweepProblem after the spatial grid and pArray of vertices is determined. Since the boundary of the problem must remain fixed, it is not affected by vertex motion, so this global index calculation can be performed before this vertex motion occurs. A description of the local calculation of global minimum and maximum track indices is presented next. This method, called setLC_global_index_values( ) or setSTLC_global_index_values( ) can be found in LogicallyRectangularGrid or LogicallyHexahedralGrid. Note that this global min and max calculation is not needed in 2D (x, y).

The local global min and max ray index calculation begins with a loop over all cells on this processor followed by a loop over edges in each cell. Next, since this calculation is looking for global minimum and maximum track indices, only global boundary edges are considered. If a processor does not have any global boundary edges, the index calculation is performed on the first cell and first cell edge on the processor because this information will be replaced once the all-reduce across processors is done. Next, groupsets are looped over followed by the anglesets in the groupset and the angles in the angleset. For the STLC method, a loop over groups in the set is also needed since the track index values depend on speed as evident in the equations given in above sections. The vertex positions on this edge are then converted to the rotated coordinate system values and then to track indices given by i and j. Then looping over these

vertices the min and max vertex indices are found. The closest integer track index to these min and max vertex indices is then found and compared to the local min and max stored on this processor. The local min and max *i* and *j* indices are stored for each angle with the quadrature variables. After all processors have found their local min and max track indices, an all-reduce is performed to find the global values and store them on each processor in the quadrature class in the variables called: IMIN, JMIN, and JMAX.

After these values have been set, vertex motion is performed if it is requested from the input file. Note that vertex motion must be performed before the edges are initialized for the LC methods. This is because the edge initialization depends on the position of the vertices of a cell to determine the number of tracks entering this cell for each angle. After performing vertex motion, edges are initialized by calling a routine in ArbitraryPGrid.h called initialize_edges_lc( ) or initialize_edges_stlc( ). These methods are comparable to the usual call to initialize_edges( ) used with other spatial discretization methods. The edge initialization routines for the LC methods set up the track information on each edge of the cell by calling other methods. For a PWC-LC or PWL-LC problem in 2D (*x*, *y*) and a STLC problem (*x*, *y*, *t*) these methods are found in LogicallyRectangularGrid. For a PWC-LC or PWL-LC problem in 3D (*x*, *y*, *z*), these methods are found in LogicallyHexahedralGrid.

Finding track intersections in 2D (x, y)

This section gives the details of the geometric track and edge calculations in 2D (*x*, *y*). First, the findLC_rayintersections() routine for 2D STEP-LC or PWL-LC problems is reviewed. This calculation begins on a given edge with a loop over

groupsets, then a loop over anglesets, and a loop over angles in this angleset. The layout of tracks is dependent on the angle as evident by the equations given in the above sections. This code is written for a four-sided cell (hence it being in LogicallyRectangularGrid), so it is assumed each edge has two vertices. Using the edge numbering of PDT, the *x* and *y* coordinates of the north and south vertices are stored based upon the comments in this section of the code. Next the track index, denoted by *i*, of the north and south vertex is found using Eq. (B.1), which is from Eq. (5.18).

$$i = \frac{|\mu_m|}{\Delta\omega_m \sqrt{\mu_m^2 + \eta_m^2}} \left(y + y_0 - (x - x_0) \frac{\eta_m}{\mu_m} \right) \tag{B.1}$$

In this equation, it has been assumed the y_offset, y_0, has the opposite sign than what is used in Eq. (5.18). For an edge on the global boundary of the problem, the track ID's incoming on this edge must be stored. Therefore, inside the global boundary edge loop, a separate loop is needed for incident angles. This incident angle loop must account for the convention for moving tracks that will take place in the sweepchunk discussed later. This convention is outlined for vertex intersections in the comments in this loop. For angles with μ and η values that have the same sign, the track is moved along the ω-axis in the negative direction. The opposite is done for the other angles, where the positive and negative direction of the ω-axis is based upon both μ and η values being positive. Keeping this movement in mind, the upper and lower integer track indices between the two vertex index values are found. This number of tracks, also known as the degrees of freedom and the number of elements, on this edge for this angle is stored in the total_tracks vector. The lowest track ID for this angle on this boundary edge is stored in

the track_start_id vector and all of the track ID's on this edge for this angle are stored in the tracks_found vector. The non-incident angle loop on a boundary edge is simpler because it does not need to move tracks near vertices and does not need to store the list of track ID's on the edge. This loop is also the same as the internal edge loop. Finally, the maximum number of tracks on an edge over all angles and groups is returned to set the size of the edge buffer.

Finding track intersections in 3D (x, y, z)

Next, the ray intersection calculation for 3D LC methods is reviewed. This calculation is performed in the findLC_rayintersections() routine in LogicallyHexahedralGrid. The convention for determining if a track is associated with a boundary edge is based upon how the track will be moved in the sweepchunk. If a track intersects a vertex of the "edge" of a face, it is moved toward the face center (see the comments in PDT). Therefore, for a track intersecting an "edge" or vertex, the track is given to the cell with the lowest ID and the first face of this cell that finds this track. This routine again begins with a loop over groupsets, loop over anglesets, and loop over angles. Inside this loop over angles, the coordinates of the vertices of this edge are converted to rotated coordinates, u and ω, and the corresponding index values, i and j, using Eq. (5.21). Next, the minimum and maximum integer index values over all vertices on the edge are found in order to loop over these integers to find the total number of tracks on an edge.

The global boundary loop has two separate routines for incoming and outgoing angles on this edge. For an incident angle, the i-values are looped over to find the j-

values on each of these *i* "rows", and hence the number of tracks. For each integer *i*-value, a routine called find_bdry_edge_tracks() is called. The details of this routine will be given in the next paragraph, but it ultimately finds the total number of tracks and the track ID's on this edge for this angle. If the track flag is set in this routine, there are no tracks on this edge for this *i*-value, so zero is added to the totaltracks counter. If a horizontal edge flag is set, the track ID's have already been put in the correct vector, so only the totaltracks counter needs to be incremented. Otherwise, the totaltracks vector is incremented and the track ID's are put into the tracks_found vector by their global ID calculated using Eq. (5.20). For a non-incident angle on a boundary edge, the tracks are found in a routine called find_interior_edge_tracks(), which is called inside a loop over integer *i*-values. These track ID's do not need to be saved so only the totaltracks vector needs to be incremented. The loop for interior edges for all angles is the same as for non-incident angles on a boundary edge. Next, the total number of tracks on the edge for this angle is saved in the track_n_values vector and the maximum number of tracks over all angles and groups is returned to set the size of the buffer on this edge. Now the particulars of the routines called in this findLC_rayintersections() method are outlined.

The find_bdry_edge_tracks() routine is where the tracks intersecting this boundary edge are found. Remember that this routine is called from inside a loop over integer *i*-values, so it is trying to find the integer *j*-values associated with this *i*-value on this edge. It begins with a loop over the vertices on this edge, where *k* is the current vertex and *vertCW* is the neighboring vertex, therefore they form an "edge" of this face. This first loop over vertices is checking if there is a horizontal edge on this face that the

given *i*-value is near. If this *i*-value is near both of the vertex index values, the loop proceeds. Next, the edge's sister ID is found along with the neighboring cell. This is done by each vertex knowing which cells touch it. If the edge's cell ID is larger than its neighbor, there are no tracks on this edge for this *i*-value so the track_flag is set and the loop over vertices is exited. If the edge's cell ID is the lowest, the vertices of this "edge" are checked for an intersection with an integer *j*-value by checking the ceiling and floor values of the vertex *j*-values. If there is a track within a tolerance of a vertex, the vertex_intersect() method is called which is detailed in the next paragraph. If this method does not find any tracks near this vertex, the track_flag is set and the vertex loop is exited.

The variables denoted by *jCW* and *jCW_other* denote the bounding j-values for this *i*-value on the edge. They are set if a track is near a vertex by the vertex_intersect call. If there is no track near a vertex, these variables are set by checking if the ceiling or floor of the vertex *j*-index is within the face by using cross products through the find_cross_products( ) method. Only one or none of these values will be within the face and the *jCW* or *jCW_other* integer can be set accordingly. Once these values have been set, it must be determined if these tracks have already been taken or they can be put in the tracks_done and mytracks vectors. The variables trackstart and trackend now take over the *j*-values and *jCW* and *jCW_other* become counting variables to keep track of the tracks on this edge for this *i*-value. Looping over the *j*-values, if this track is near a vertex it has already been added to the tracks_done vector via the vertex_intersect() method and its ID is found and put into the mytracks vector. If not near a vertex, the ID

of the track is checked with the tracks_done vector. If it has not been taken its ID is put in the tracks_done and mytracks vectors. Finally, if all of the tracks have been taken, there are no tracks for this *i*-value and the track_flag is set. At the end of the horizontal check, the horiz_flag is set so the next loops are broken. The rest of this find_bdry_edge_tracks() routine will be returned to after a few paragraphs detailing other methods.

The find_cross_products() method is a simple method that again loops over vertices of the edge to form face edges and determine if the given track is within the face by calculating the cross product of the vectors making up the face edges and the track position. This routine loops over the vertices on the face and gets the neighbor vertices on either side of the current vertex. These three points form two vectors; the vector pointing from vertex *k* to vertex *vertCW* and the vector pointing from vertex *k* to vertex *othervert*. The cross product of these two vectors with the vector pointing from vertex *k* to the given track position is calculated, yielding the values *cp1* and *cp2*. If one of these cross products is zero the track could be on the edge, so the usual cases are checked and if the track does not satisfy any of them the onface flag is set to false and the vertex loop is broken. If the signs of the cross products are the same, the track is most likely outside of the face but several cases must be checked to ensure this is true due to numerical precision. If the signs of the cross products are different, the track is most likely inside of the face. After looping over all vertices, all edges of the face have been checked to determine what side the track lies and the value of the onface flag is returned.

The vertex_intersect() method is a simple routine that determines if the given track is owned by this cell, once it has already been determined to be near a vertex of the face. First, the cell ID's for all cells touching this vertex are looped over to find the minimum cell ID. If the current cell is not the minimum, this track is not owned by this cell, but the *j*-value that is on this face near this vertex must be determined. If this method is being called from the second loop over vertices in the find_bdry_edge_tracks routine, and the *jCW* value is the same as *jCW_other*, there are no tracks on this face for this *i*-value because this cell doesn't own the track near the vertex. To find the *j*-value if this is not true, the integer values above and below the vertex are checked by calling the find_cross_products() routine to determine which track is on the face. If neither track is on the face, the track_flag is set because there are no tracks for this *i*-value on this face. Otherwise, if this current cell is the minimum, the global track ID is calculated and the tracks_done vector is checked to see if it has already been taken by another face of the cell. If it has been taken, the same routine is followed for if this cell was not the minimum. If the track has not been taken, its ID is put in the track_done vector if this face is on the boundary of the problem and the *j*-value is set.

Returning to the find_brdy_edge_tracks() method, after the first loop over vertices to check for horizontal "edges", a second loop over vertices is entered. If the horiz_flag or the track_flag has been set by the previous loop, this loop is exited. Otherwise, this loop sets the neighboring vertex ID and begins to check for vertex or edge intersections for the given *i*-value. If the given *i*-value is near a vertex and lies within the edge made up of the two vertices, the *i*-value is checked to see if it is equal to

the *i*_min or *i*_max calculated for this face. If so, the ceiling and floor integer values of the *j*-value for this vertex are checked and if a track is within a tolerance of the vertex, the min_max_vertex_intersect() routine is called and the vertex loop is broken. If neither the ceiling nor floor of the vertex *j*-value is on the face, there are no tracks on this face for this *i*-value and the track_flag is set. If the *i*-value is not equal to the *i*_min or *i*_max for this face, the integer *j*-values near the vertex are checked and, if within a tolerance, the vertex_intersect() routine is called and the loop is broken. Finally, for an *i*-value near a vertex and within this edge, if no integer *j*-value is near the vertex, the *j* along the edge is calculated using the following equation, where *k* and *CW* refer to vertex *k* and vertex *CW*.

$$j = \frac{\left(i\Delta\omega - u_k\right)\left(\omega_k - \omega_{CW}\right) + \omega_k\left(u_k - u_{CW}\right)}{\Delta u\left(u_k - u_{CW}\right)} \tag{B.2}$$

The integers above and below this *j*-value are checked for an intersection with this edge, and if found the edge_intersect() routine is called. If no intersection, these integers are checked to see which is on the face by calling the find_cross_products() method and the *j*-value for this loop is set accordingly. If neither of these integers is on the face, the track_flag is set. For an *i*-value that is near a vertex but not within this edge, this *i*-value could be outside of the face but still near a vertex. If this is true, the *i*-value must fall within the *i*_min and *i*_max for this face and therefore the min_max_vertex_intersect() method can be called to determine the *j*-value near the vertex. After both vertices are checked for an intersection with *i*-value, the edge must be checked for an intersection. If

the *i*-value lies within this edge, the *j* along this edge is calculated using Eq. (B.2). The same edge intersection routine is followed afterward.

Once this loop over vertices has been completed or broken, a final loop over vertices begins starting where the previous loop ended to determine the value of *jCW_other*. Therefore, the *jCW* and *jCW_other* values determine the total tracks along this *i*-value on this face. If track_flag, horiz_flag, or extreme_flag (set in min_max_vertex_intersect()) is set, this loop over vertices is broken because all of the tracks have already been found for this *i*-value. The loop proceeds to find a neighboring vertex, set the secondloop flag, and then check for vertex and edge intersections in a similar manner as the previous vertex loop. The only difference from the first loop is a check in the two vertex intersection sections to set *jCW_other* equal to *jCW* if this vertex has already been done by the previous loop. Once the end of the find_bdry_edge_tracks() method is reached, the tracks on this face for the given *i*-value have been found.

One of the methods called in find_bdry_edge_tracks() is the min_max_vertex_intersect() method. This routine is very similar to the vertex_intersect() method. The only difference is that if the current track, which is near a vertex, is not on this cell for whatever reason, the closest track on the face does not need to be found because there are no tracks for this *i*-value on this face since the *i*-value is near one of the extreme vertices of this face and outside of the face.

Another simple method used in the boundary edge tracks determination is the edge_intersect() method. This method is also very similar to the vertex_intersect()

method. In order to determine if this cell is the lowest cell ID, the cells that each vertex of this edge of the face touches must be checked. Once this current cell ID and the sister ID that shares this edge is determined, the procedure to check the track and determine if it is taken or a new one needs to be found is the same as the vertex_intersect() method.

Now the routine for finding tracks on an interior edge or a non-incident boundary edge is outlined. Again, the find_interior_edge_tracks() method is called within a loop over *i*-values on this face from the findLC_rayintersections() method. It is simpler than the boundary edge method because it only cares about finding an estimate for the maximum number of tracks on this edge for a given angle without needing to determine if tracks will be moved. The loops over vertices are the same as the boundary edge routine, beginning with a loop over vertices to check for an intersection with a horizontal edge of the face. If the *i*-value is near both vertex *i*-values on this edge, this edge is horizontal and the *j*-values on this face are found by checking the ceiling and floor of the *j*-value of each vertex and picking the one closest to the vertex and potentially on the face. The next loop over vertices checks the usual flags and breaks from the loop if they are set. Otherwise, the neighboring vertex is found and the *i*-value is checked for a potential intersection with either vertex or an edge. The rest of this loop is very similar to the boundary edge loop. The same is true for the final loop over vertices to set the value of *jCW_other* if it has not already been set; it is very similar to the last boundary edge vertex loop. Since this method does not care about whether a track needs to be moved it can directly set the *j*-value to the closest integer value of a vertex or edge.

Once this find_interior_edge_tracks() method is completed the number of tracks for this *i*-value on this face for the given angle has been found.

Finding track intersections in 3D (x, y, t)

Finally, the STLC ray intersection calculation in three dimensions, (*x*, *y*, *t*), is reviewed. These calculations are combinations of the 2D and 3D geometric calculations already outlined. The first method called in initialize_edges_stlc() is the set_STLC_time_bdry() routine which finds the tracks incoming on the starting time boundary of the (*x*, *y*, *t*) cell. This routine is found in LogicallyRectangularGrid.h and is similar to the 3D calculation of findLC_rayintersections() in LogicallyHexahedralGrid. After getting the coordinates of the vertices for this cell, a loop over groupsets is entered followed by a loop over anglesets and a loop over all angles in the angleset. The coordinates of the vertices are converted to *i*-values and *s*-values using Eq. (B.3) and Eq. (B.4), which are related through the geometry of the rotated coordinate systems from section 3.2 and Eq. (5.22).

$$i = \frac{\left(y - y_0\right)\frac{\mu_m}{\sqrt{\mu_m^2 + \eta_m^2}} - \left(x - x_0\right)\frac{\eta_m}{\sqrt{\mu_m^2 + \eta_m^2}}}{\Delta\omega_{k,m}} \tag{B.3}$$

$$s = \frac{\sqrt{\left(x - x_0 + i\Delta\omega_{k,m}\frac{\eta_m}{\sqrt{\mu_m^2 + \eta_m^2}}\right)^2 + \left(y - y_0 - i\Delta\omega_{k,m}\frac{\mu_m}{\sqrt{\mu_m^2 + \eta_m^2}}\right)^2}}{\sqrt{\mu_m^2 + \eta_m^2}} \tag{B.4}$$

Next the minimum and maximum *i*-values and integer *i*-vals over these vertices are found as was done in the *x*-*y*-*z* calculation.

A loop over groups in the groupset is then entered. Inside this loop the *j*-values of the vertices can be calculated, since this index depends on group speed as given in Eq. (B.5), where *v* is the speed of the particle, *t* is the initial start time (usually t = 0), and *s* is the previously calculated *s*-value for the vertex.

$$j = \frac{\sqrt{\mu_m^2 + \eta_m^2}\left(v\left(t - t_0\right) - s\right)}{\Delta u_{k,m}} \tag{B.5}$$

Then, the minimum and maximum *j*-values over the vertices are found. Next, the integer *i*-vals are looped over to find the *j*-values on this time boundary, similar to what was done in 3D (*x*, *y*, *z*). This calculation is done by calling the find_tracks() method. After these *j*-values are found for one *i*-val, the totaltracks vector is incremented and the track ID's on this time boundary are put into the mytracks vector. Once the *i*-val loop is finished, the psi and oldpsi vectors for this angle and group are sized for the total number of tracks plus one value. These vectors hold pairs for each angle and group to represent the track ID and the incoming angular flux. The first pair only stores the total number of tracks for this angle and group, with the second element of the pair being a dummy. After these psi and oldpsi vectors are sized, they are initialized with track ID's and initial angular flux values of zero.

After all groupsets have been looped over, the initial angular flux values are set through a call to set_STLC_initial_condition(). This method sets the angular flux for each track in the psi vector on the cell according to the input read from the initial condition section of the input file. For a ZERO initial condition, nothing is done because the psi vector is already initialized to zero. For an ISOTROPIC initial condition, the

tracks angular fluxes for the specified energy groups are set to the given flux value that is normalized over the groupset. For a LINEAR initial condition, the position that each track intersects this time boundary is found by the usual intersection procedure. This includes moving tracks that are near a vertex, edge, or face center (which is outlined in the sweepchunk section). A different routine is needed for the PWC-STLC method and PPWL-STLC method in order to account for moving tracks on sides with the PPWL-STLC method. After the track intersection position is found, the flux is calculated using the coefficients from the input and put into the psi vector. At the end of the set_STLC_time_bdry() method, all of the incoming tracks and their associated initial angular flux on the cell's initial time boundary have been set.

The find_tracks() method called in the set_STLC_time_bdry() routine is very similar to the find_bdry_edge_tracks() method in LogicallyHexahedralGrid therefore only the main differences will be mentioned. Both of these methods seek to find the tracks on a face of a cell for a given *i*-value, but the find_tracks() routine is specialized for the time boundary of a cell. The main difference occurs if an integer *j*-value is not near a vertex or edge. The *j*-value on this edge for the given *i*-val is calculated by first finding the *s*-value on this edge by calling a method called find_track_intersect(). Then, the *j*-value on this edge is calculated using Eq. (B.5). The vertex_intersect(), min_max_vertex_intersect(), and edge_intersect() methods called in this routine also have one main difference from their counterparts in LogicallyHexahedralGrid. If a track has not already been taken and it is associated with this cell, it is put in the tracks_done vector regardless of whether this cell is on the global boundary. The rest of this

find_tracks() routine is the same as the find_bdry_edge_tracks() routine that has been outlined previously.

The find_track_intersect() method is a simple calculation of two lines that will also be used in the 2D STLC sweepchunk outlined below. Given an *i*-val and two vertices of an edge, the (*x*, *y*) intersection point of these two lines can be determined. The equation of the track when only considering the *i*-val is given by Eq. (5.18) where *k* = *i*. And the equation of the edge can be written in terms of the vertex coordinates as given in Eq. (B.6).

$$y = \left(x - x_1\right)\frac{y_2 - y_1}{x_2 - x_1} + y_1 \tag{B.6}$$

These two equations can be solved to find their intersection point. With this (*x*, *y*) point, the *s* value corresponding to this *i*-val is calculated using Eq. (B.4).

After setting the tracks on the initial time boundary of the cell, the tracks on each spatial "edge" of the cell are found through the findSTLC_rayintersections() method. This routine is very similar to the findLC_rayintersections() method already outlined in LogicallyHexahedralGrid. For a given "edge", this method begins with a loop over groupsets, followed by a loop over anglesets, and a loop over angles in the angleset. The *i*-values and *s*-values corresponding to the (*x*, *y*) position of the two vertices of this edge are then found using Eqs. (B.3) and (B.4). Keep in mind that each of these vertices has two *j*-values associated with it to correspond to the initial and final time boundaries of the cell in this first time step. Next, the minimum and maximum integer *i*-vals on this face are found by using the calculated vertex values and a loop over groups in the

groupset is entered. The layout within this group loop is much like the *x*-*y*-*z* ray intersection calculation. If the face is on a global problem boundary and the angle is incident on this face, a loop over integer *i*-vals is entered and the *s*-value on this edge for the given *i*-val is calculated using the find_track_intersect() method and passed to the find_bdry_edge_tracks() method to find the tracks on this face. If the angle is not incident on this boundary face, the *s*-value is calculated the same way inside a loop over *i*-values, the find_interior_edge_tracks() method is called to find an upper bound on the number of tracks for this *i*-val, and the totaltracks vector is updated accordingly. For a non-boundary face, the tracks are calculated for each *i*-val using the same find_interior_edge_tracks() method. Then, the total tracks on the face for this angle and group is set in the total_tracks_STLC vector. Also, just like the findLC_rayintersections() method, this method finds the maximum number of tracks on this face over all angles and groups to set the size of the edge buffer.

The rule for moving tracks and determining which face a track is associated with is the same as outlined for *x*-*y*-*z*. The find_bdry_edge_tracks() method used for spatial boundary edges of *x*-*y*-*vt* cells is similar to the analogous method for *x*-*y*-*z* cells that has already been outlined. First a check for if the *i*-val is along the edge is done, and if true, it is assumed these tracks will be/have been taken by other faces of the cell so no tracks are on this face for this *i*-val. If the *i*-val is near a vertex and inside of the edge, the *j*-values at the beginning and ending times of the time step are calculated using Eq. (B.5) with the *s*-value for the intersection with the edge previously calculated. The closest integer *j*-values, which correspond to tracks, to these upper and lower *j*-values are found

and then compared, via *j*_upper and *j*_lower. If the upper value is less than the lower value, there are no tracks on this face for this time step. Otherwise, the *j*-values from *j*_lower to *j*_upper are looped over to find the global track ID and check the tracks_done vector. If the track has not been taken its ID is put in the tracks_found and tracks_done vectors and totaltracks is incremented. If the *i*-val is near a vertex but outside of the edge, it is assumed these tracks have been/will be taken by another face or time step so there are no tracks for this *i*-val on this face. If the i-val is within the edge but not near a vertex the *j*-values are calculated for the beginning and end of the time step as before. The integer values above and below the *j*_start value are checked, and if within a tolerance, the tracks_done vector is checked with the track ID. If the track ID is not taken it is put in the tracks_done vector, otherwise the *j*_lower value is increased by one. The *j*_upper value is then set by checking the nearest integer values to *j*_stop, but the tracks_done vector does not need to be checked because these tracks could not have been taken already. Once the *j*_lower and *j*_upper values have been found for this *i*-val within the edge, the totaltracks are incremented and the track ID's within these values are put into the tracks_found vector for this edge. After the find_bdry_edge_tracks() method has been concluded, all of the tracks for a given *i*-val that are associated with this face have been found and the totaltracks counter has been incremented accordingly.

Now the find_interior_edge_tracks() method used for the STLC method in LogicallyRectangularGrid is reviewed. This method begins with a check for if the *i*-val is along the edge or parallel with the edge and in the face plane. If this is true, it is assumed these tracks will be/have been taken by other spatial faces or the time boundary

of the cell, therefore the track_flag is set. Next, if the *i*-value is near a vertex, it may be inside of the edge or outside of it. If inside of the edge, the *j*-value at the start time and stop time of this initial time step are calculated using Eq. (B.5) with the *s*-value found for this *i*-value on the edge. The nearest integer values above and below these *j*-values are checked and the *j*_lower and *j*_upper values are set. If the i-val is near a vertex but outside of the edge, the j-values of the vertex at the beginning and end of the time step are calculated and the nearest integer is found and saved for *j*_lower and *j*_upper. For an *i*-val that is not near either vertex, it must be within the edge and the *j*-values are calculated for the beginning and end of the time step using the *s*-value of the edge intersection. After finding the *j*_upper and *j*_lower values for this *i*-val, if the upper value is lower, the track_flag is set. Remember, this method only needs to determine a number of tracks on this interior edge for a given *i*-val to contribute to setting the buffer to a maximum size.

There is also a method called at the end of every time step to find the tracks incoming on the boundary edges for the next time step. This method is in ArbitraryPGrid and is called Set_STLC_new_ts_Boundary(), which is very similar to initialize_edges_stlc(). It loops over all cells and edges of the cells. For edges on the global boundary, it first puts the tracks that were already found on this edge in the previous time step into the tracks_found vector so they are not duplicated. Next, the Find_bdry_tracks_for_ts() method is called to find the tracks on this edge for this time step. This method is the same as the findSTLC_rayintersections() method, excluding the section for non-boundary edges. Therefore, it finds the maximum size of the buffer

needed for this time step as well as the incoming track ID's on each boundary edge in this time step. Next the number of elements on this edge is calculated (same as in initialize_edges_stlc) and the maximum between this number and the number of elements over all anglesets from the previous time step must be determined due to resizing the buffer (which will be discussed later). Once this number of elements has been set accordingly, the tracks are put into the boundary buffer by calling the Set_bdry_tracks_for_ts() routine and the boundary condition is set as before by calling the set_boundary_from_input() method. These methods will be detailed below.

Putting tracks in the buffer

The final method called in the initialize_edges LC routines sets the previously found tracks on the boundary edges. This method is called from inside the edge loop, only for boundary edges, and can be found in LogicallyRectangularGrid or LogicallyHexahedralGrid. For the 2D and 3D LC methods, this function is named set_tracks_from_input() and for the STLC methods it is named set_STLCtracks_from_input(). The boundary buffer is passed to these methods in order to put the total tracks and incoming track ID's into this buffer. It also serves to initialize this boundary buffer. In the set_tracks_from_input() method for the LC methods, the tracks stored on the edge are gotten via get_track_n_values() or get_total_tracks() as well as the number of elements, which is the maximum over all angles and groups. Next, the boundary buffer is initialized by looping over anglesets and then angles and groups and setting all of the elements to zero. Then, a separate loop over angles in the angleset determines if this angle is incident on this edge and gets the totaltracks for this

angle. Finally, inside a loop over groups, the number of tracks is placed in the first elements position on the buffer and the track ID's are put into the following element positions using the tracks_found vector. The set_STLCtracks_from_input() method is the same except the total number of tracks and track ID's are dependent on group and angle.

To set the boundary tracks at the beginning of each time step after the initial time step, the Set_bdry_tracks_for_ts() method is used, as called from Set_STLC_new_ts_Boundary(). This method is the same as the set_STLCtracks_from_input() method except the buffer vector does not need to be resized because the number of elements has been set accordingly and passed to this method.

To set the incoming boundary fluxes on the buffer for these LC methods is also slightly different than other methods via the set_boundary_from_input() method. As mentioned previously, the boundary conditions are set based upon the input in BoundaryInputs.h. Unlike other methods, the boundary buffer has already been initialized and the track ID's have been put into it for these LC and STLC methods, so it must not be overwritten when the angular flux values are put into this buffer. Also, the position at which these angular flux values start in the buffer is dependent on the number of elements on the edge. Remember this number is equal to twice the maximum number of tracks plus one. Therefore the position at which the angular flux values start on the buffer is the ceiling of the number of elements divided by two. Also, each angular flux value in this boundary buffer should correspond to a track incident on this boundary

edge. However, this is not enforced for blanket conditions that do not depend on position or angle. The Linear boundary condition is the main boundary condition that must take into account each track already in the buffer by finding the position at which this track intersects the edge. This intersection calculation accounts for moving a track on the edge according to how it is done in each sweepchunk. There are separate routines for the LC and STLC linear boundary conditions called set_linear_LC and set_linear_STLC. One thing to note is that the STLC linear boundary condition can depend on time so the A_vt coefficient is needed in the input of a linear boundary condition for STLC. Each track has a different incident angular flux depending on where it intersects the edge for this linear boundary condition.

Long Characteristic SweepChunks

The rest of the setup for a problem is the same as with other methods and will not be presented here. The next difference with these LC methods occurs in the actual solving of the transport equation that occurs in what PDT calls the sweepchunk. A "sweepchunk" is defined by dividing a problem into cellsets, groupsets, and anglesets. The solution procedures for these chunks are defined in either Discretizations_2D or Discretizations_3D and they are called from the operator of the solver routine. The LongChar_2D.cc file is the basic solving routine for the PWC-LC and PWL-LC methods in (x, y). LongChar_3D.cc is the corresponding routine for these methods in 3D (x, y, z). STLC_2D.cc is the solving routine for the PWC-STLC and PPWL-STLC methods. Each of these methods calls routines that are specific to the source approximation specified. The XY_SLC_Method is called when running the PWC-LC method in 2D

with the corresponding XYZ_SLC_Method called for 3D problems. The XY_LLC_Method is called when running the PWL-LC method in 2D and the XYZ_LLC_Method is called in 3D. For the STLC method, the XY_STEPSTLC_Method is called when running the PWC-STLC method and XY_PWLSTLC_Method is called when running the PPWL-STLC method. At the time of this paper, these are the only methods implemented in PDT-PTTL.

2D LC SweepChunk

This detailed review begins with the LongChar_2D() method. This method is performed on one chunk of the problem, which consists of one cellset and one groupset. The speed for each group in this set is calculated based on the theta scheme if this is a time-dependent problem before entering a loop over cells. Once inside a loop over the ready cells in this chunk, the local contiguous vector to store the angular flux values is initialized. This is done by looping over edges on this cell and getting the number of elements for an edge. The contiguous vector psi_face is sized and initialized to zero for each element, angle, and group in this set. The buffer for each angleset must then be retrieved. The buffer on each edge owns each angleset that is incident upon it, which is determined by the primary angle in the angleset. If the edge owns the angleset in this chunk, it is retrieved and put into the local psi_face vector. If not, due to vertex motion, some angles in the angle set may be incident on this edge, but these values are stored on the sister edge so a loop over angles must be entered. If one angle is incident on this edge, the buffer information for the whole angleset is retrieved from the sister and put into the local psi_face vector and the angle loop is broken.

This method has a separate loop for a time-dependent problem and a steady-state problem. For a time-dependent problem, this solve begins with a loop over each angle in the set. The exit_face vector keeps track of which faces are exiting for this angle, and is set in the find_exit_face() method. This simple method determines the exit faces of the given cell by looping over edges and testing if the given angle is incident through a dot product with the edge normal. If the angle is not incident on an edge, the edge index in the exit_face vector is set with the elements on this face. After this vector has been set, a loop over energy groups is entered. Inside this loop, the total cross section is adjusted based upon the theta scheme as well as the angular source on each corner of the cell. Then the psi_face vector for exiting faces is initialized with a temporary number of total tracks from the total_tracks vector saved on the edge. Also, the incoming faces are checked to output a warning if no tracks are incident on this edge for the given angle to make sure the user is aware. After this, the temporary matrices, Mlctemp and Tlctemp, that correspond to the matrices M and T that were detailed in the algorithms section are sized and initialized. Finally the compute_cell_fluxes() solving routine is called to follow the tracks through the cell and contribute to these matrices and find the exiting angular flux for each track. This routine will be detailed later for the different source approximations.

After the temporary matrices are contributed to for this angle, these values are put into the cell matrices, Mlc and Tlc, after being multiplied by the appropriate weight functions. For this time-dependent loop, the angular flux at the end of the time step is found by using the algorithm previously outlined. This is done by using the temporary

matrices Mlctemp and Tlctemp through a call to the Gauss Elimination solver with no pivoting, GE_no_pivoting(). The angular fluxes at the corners of the cell are stored in the right hand side vector, given by b, after this solve. This flux contributes to the angular flux on each corner of the cell through the theta scheme which uses this current flux and the old flux (stored in oldpsi).

At the end of this loop over angles and groups, all of the angular fluxes for this groupset have been found and the Mlc and Tlc matrix values have been set. The steady-state loop is the same as the time-dependent loop without the adjusted speed, angular source, and solving for the angular flux at the end of the time step. After solving for all of the angles in the set, the angular flux values stored in the local psi_face vector must be put into the buffer for the neighboring cells. Looping over the edges of the cell, if this edge owns the angleset, the psi_face values are written to its buffer, otherwise they are written to its sister's buffer.

3D LC SweepChunk

Before discussing the 3D LC sweepchunk, the notion of "sides" on a 3D cell is briefly reviewed. Based upon previous research into the PWLD method[21,22,27,28], the cell is divided into sides on which each chosen basis function is linear. These sides are tetrahedra on each face of the cell, so a hexahedral cell has twenty four sides with each face of the hexahedral having four sides touching it. The sides on a face can each have different outward normal vectors, therefore a face is faceted by these different planes. These faceted faces and the use of sides for flux calculations are the need for some of the procedures detailed in the 3D LC sweepchunks.

The sweepchunk for 3D problems using an LC method is very similar to the 2D LC sweepchunk. The differences are detailed here. When setting up the variables needed for a cell in the chunk, the average external face normals are set through the call to init_cell_info(). These face normals are set for each edge by looping over all of the sides on each cell and contributing the side's area normal to the average face normal the side is associated with. These face normals are only computed once for each cell, the very first time a sweep is performed in the whole problem, by setting a flag to false after they are computed. The average face normals are then normalized.

Another difference in this 3D sweep is the use of the edge_flag which is set for each face and angle on a cell and is gotten through a call to get_LCedge_flag(). This flag is set to true if any side on a face is incident for a given angle. This is needed because, although a face may not be incident based upon its average face normal, some tracks may be incident on one of the sides on the face and the buffer must be retrieved for this angle to get these incident tracks. Besides these two changes, this chunk follows the same solution procedure as the 2D LC sweepchunk.

2D (x, y, vt) STLC SweepChunk

This section details the main sweepchunk for the 3D STLC methods. Since this method is only used for time-dependent problems, it does not have a separate loop for steady-state problems. The layout of this routine is very similar to the 2D LC steady-state loop sweepchunk. First, the vectors and the local psi_face vector are initialized in the same manner as the other LC sweepchunks. Additionally, two local vectors for each face, called my_global_tracks and my_global_fluxes, are used to accumulate the tracks

and fluxes exiting on each face for each angle in this set during the solution process in order to account for the possible need to resize the local psi_face vector for this angleset. After initializing these variables, the tracks and associated angular fluxes incident on the time boundary of the cell are retrieved and stored in the vector called psi_time_inc. Next, inside the loop over angles, the my_global_tracks and my_global_fluxes vectors must be sized for the number of groups because each group can have a different particle speed and therefore a different number of tracks. The procedure inside the loop over groups is the same as the 2D LC routine, up to putting the local matrix values into the Mlc and Tlc matrices. The tracks and fluxes found exiting the cell on the time boundary for this angle and group, which are stored in the time_tracks vector, are then put into the psiLC vector stored on the cell. Before exiting the loop over groups, the number of tracks on each exiting face is checked in order to find a maximum over all angles and groups, stored in the maxtracks vector. Before exiting the loop over angles, the exiting faces for each angle are stored in the my_exit_faces vector.

After solving for all angles in the set, the local psi_face vector for each face is resized if needed. This is done by checking the maximum tracks for each face with the number of elements in the buffer for the face. If the number of elements needed for this face is larger than the number of elements currently in the buffer, the psi_face vector is resized for the entire angleset and initialized to zero. Then looping over angles and groups, the track ID's and exiting fluxes for exiting faces stored in the my_global_tracks and my_global_fluxes vectors are put into the local psi_face vector in the appropriate position. Finally, the values in the psi_face vector are written to the edge if it owns the

angleset, or written to the sister edge if not owned by this edge. This call to set_psiface_STLC(), which is located in BaseEdge.h, appropriately resizes the buffer based upon the size of the psi_face vector that is passed to it. After completing this step, this sweep has solved for the exiting angular fluxes and the contribution to the matrix values for all of the cells in this cellset.

For the STLC method, the buffer for each edge may need to be resized for each angleset in a time step. The set_psiface_STLC() method in BaseEdge.h is where this resizing is done. If the size of the vector passed to this method is larger than the current angleset's buffer size, the current buffer is reset and the number of elements on this edge for the angleset is reset via the LC_n_elements vector. A similar routine is used to set the buffer for an entire groupset, which is done for the boundary edges of the problem. These buffers usually need to be resized for the STLC method in each time step after the tracks incoming on these boundary edges are found.

"Side" Information

Before proceeding to the specific details of the compute_cell_fluxes() routines for each LC and STLC method (which find the exiting angular fluxes along each track and contributes values to the Mlc and Tlc matrices) some details about the "sides" of a cell are discussed. Based on PWL-DFEM research, the PWL basis implementation means dividing polygons and polyhedra into sides because each basis is linear on a side as mentioned previously. Therefore, for a 2D polygonal cell, the cell is divided into N triangular sides that correspond to the number of vertices that make up the cell. For a 3D polyhedral cell, the number of tetrahedral sides is related to the number of vertices

and faces of the cell, for example 24 sides for a hexahedral cell. For the PPWL-STLC method involving (*x*, *y*, *vt*) cells, the cell is divided into N triangular prism sides. The number of sides for a regular 3D cell are not needed for the PPWL-STLC method because the basis function is only linear in time, not PWL. The full PWL-STLC would need to solve on the same number of tetrahedral sides as in the equivalent 3D polyhedral cell.

Each cell contains information about its sides that is needed for the LC and STLC methods. This side information is found during the problem setup that occurs in Rectangular_cell_corner_creator and Hex_cell_corner_creator particulary in create_cell() and init_parray_vertices(). In (*x*, *y*) geometry, the side information is stored in vectors on the cell. The ID of a side is the ID of the vertex that is the north_corner of the face it is associated with and is stored in the sp vector. The face ID a side is associated with is stored in the sf vector. The ID of the side counter clock-wise from the current side is stored in the ssccw vector and the clockwise neighbor is stored in the sscw vector. The volume of a side is stored in the svol vector.

In (*x*, *y*, *z*) geometry, a side class is used to store side information for each cell. The LC information needed for a side is which sides share the face of each tetrahedral side. The ID of the side that shares the side-face made up of vertex 1, the face center, and the cell center is stored in the ssv1fz of the side class. The side that shares the side-face made up of vertex 2, the face center, and the cell center is stored in ssv2fz. The side that shares the side-face made up of vertex 1, vertex 2, and the cell center is stored in

ssv1v2z. An extra array, called side_order, is also needed in 3D in the cell class that stores the side ID's on each face of the cell.

2D PWC-LC Angular Flux Solver

The angular flux solver for the 2D PWC-LC method is found in XY_SLC_Method.cc and is called compute_cell_fluxes(). This routine finds the exiting angular flux for each track entering this cell as well as contributes to the Mlc and Tlc matrices for the given angle and group. Remember, the contribution to the Mlc and Tlc matrices is only done for convenience of using the same scalar flux solving routine as the PWL-LC method. This PWC-LC method only needs to solve for an average value in a cell.

The compute_cell_fluxes() method begins by looping over the cell edges and determining if the given angle is incident on an edge. If incident, the incident tracks for this face are looped over by using the local psi_face vector for this angle and group. For each track, first the incoming *x-y* position on this face is found via the find_track_intersect() method located in LongChar_2D.cc. This is a simple method that determines the intersection between the equation for a track and the equation representing the cell edge. This method was outlined previously in the *Finding track intersections in 3D (x, y, t)* section using Eq. (5.18) and Eq. (B.6). Note that this method has special cases for vertical and horizontal edges (which are the usual cases with orthogonal grids). Next, this incoming position for the track is checked for a potential vertex intersection on this edge by calling the check_start_intersect() method located in LongChar_2D.cc. If near a vertex of the edge, this method moves the track along the

edge away from the vertex by a fraction of the edgelength. It also saves the amount the track has been moved in *x* and *y* to use later.

After the incoming track position has been determined, the exiting position and distance to this exit on the cell is determined. This is done by looping over the edges of the cell and checking the exit faces according to the previously calculated exit_face vector. For a potential exit face, the exiting position of this track is calculated using the find_track_intersect() method and the distance between the starting position and this exit position is stored in the distance vector as well as the slope of this exiting edge. After all potential exiting edges have been looped over, the shortest distance stored in the distance vector needs to be found. This is done by looping over the values in the distance vector and performing the following procedure.

If the direction of the track is parallel to the slope of the exiting edge, the track cannot intersect it, so this loop over exiting faces proceeds only if the track is not parallel to the edge. Inside this loop, first the exiting edge is checked for a potential vertex intersection using the check_start_intersect() method where the track_move vector is used to account for if the track was moved initially. If the exiting track position is not moved, the distance does not need to be adjusted. If the exiting track position is moved, the incoming position is adjusted by adding to the track_move vector and calling the find_track_intersect() method again using the incoming edge. Then the distance between this new incoming position and exiting position is recalculated. Finally, if this distance between enter and exit positions is positive and is less than the value stored in the distance variable, this distance is saved along with the exiting face.

Once the exiting face and distance between incoming and exiting position is found for this track, the path length traveled in the cell is calculated by the following equation (which incorporates the Jacobian for 2D x, y).

$$\Delta s_k = \frac{\sqrt{\left(x_{k,in} - x_{k,out}\right)^2 + \left(y_{k,in} - y_{k,out}\right)^2}}{\sqrt{\mu_m^2 + \eta_m^2}} \tag{B.7}$$

Using this track length, the exiting angular flux and the contribution to the Mlc and Tlc matrices is calculated. From the development of this PWC-LC method previously, the exiting angular flux is determined by the following equation.

$$\begin{aligned} &\psi_{m,k}^{out}\left(s = \Delta s_k\right) = \psi_{k,inc} e^{-s\sigma_t} + \int_0^s ds' q_{tot,k}\left(s'\right) e^{-\sigma_t\left(s-s'\right)} = \\ &\psi_{k,inc} e^{-s\sigma_t} + q_{k,B} \int_0^s ds' \left(\frac{\Delta s_k - s'}{\Delta s_k}\right) e^{-\sigma_t\left(s-s'\right)} + \\ &q_{k,E} \int_0^s ds' \left(\frac{s'}{\Delta s_k}\right) e^{-\sigma_t\left(s-s'\right)} \end{aligned} \tag{B.8}$$

The conservative flux is called psi_cell and is defined as the following based upon the previous development section.

$$\begin{aligned} &\psi_m^{cons} = \sqrt{\mu_m^2 + \eta_m^2} \sum_{k=1}^{\#tracks} \Delta\omega_k \int_0^{\Delta s_k} ds \left(\psi_{k,inc} e^{-s\sigma_t} + \int_0^s ds' q_{tot,k}\left(s'\right) e^{-\sigma_t\left(s-s'\right)} \right) = \\ &\sqrt{\mu_m^2 + \eta_m^2} \sum_{k=1}^{\#tracks} \Delta\omega_k \int_0^{\Delta s_k} ds \left(\begin{array}{l} \psi_{k,inc} e^{-s\sigma_t} + q_{k,B} \int_0^s ds' \left(\frac{\Delta s_k - s'}{\Delta s_k}\right) e^{-\sigma_t\left(s-s'\right)} + \\ q_{k,E} \int_0^s ds' \left(\frac{s'}{\Delta s_k}\right) e^{-\sigma_t\left(s-s'\right)} \end{array} \right) \end{aligned} \tag{B.9}$$

The Jacobian term that is present here is not included in this track calculation; it is multiplied by the fully accumulated term in LongChar_2D.cc before it is put into the Tlc matrix. The source at the starting position, or beginning, of the track is called

avgsourceB with the source at the exiting position called avgsourceE. It is assumed the fixed source that is from the input file is a constant in each cell. If some other representation for the fixed source is input, the value of the avgsourceB and avgsourceE would need to be interpolated between the corner values at which the angular source is stored. These average source values at the beginning and ending point of the track on the cell are gotten from the angular source vector which was populated in Long Char_2D.cc and uses the flux from the previous iteration for the scattering source.

The variable tau is used to store the attenuation factor which is defined as the following for a track.

$$\tau_k = \sigma_t \Delta s_k \tag{B.10}$$

A separate loop is used to find the exiting angular flux and conservative flux contribution if the value of tau is small in order to use an approximation for the exponential function. For small tau, the exponential function is approximated by a Taylor series expansion given in Eq. (B.11).

$$e^{-\tau} \cong 1 - \tau + \frac{\tau^2}{2} - \frac{\tau^3}{6} + \frac{\tau^4}{24} - \frac{\tau^5}{120} + \ldots \tag{B.11}$$

For large tau, the exponential function is directly evaluated. After finding the exiting angular flux for this track, the track ID and angular flux value, stored in temp, are put into the local mytrack and myfluxes vectors for this exiting face. Since the scattering source is approximated as a constant in the cell, the contribution to the Mlc matrix is just the area associated with each track in the cell. This area is found by multiplying the

track width, given by omega_spacing, by the track length, dels, and this value is accumulated in the volumesum variable.

Finally after looping over all tracks incident on the cell, the values stored in the mytracks and myfluxes vectors for each exiting face are put into the local psi_face vector for the current angle and group. Remember the first element in this vector is the total number of tracks on this edge so it is set to the size of the mytracks vector for each exiting face. The startflux variable is needed in order to determine at which element position the angular flux values should be put in the psi_face vector. The track ID's and the associated exiting angular flux on each exiting face is then put in the appropriate position of this psi_face vector. The last loop in this compute_cell_fluxes() routine puts the conservative flux and track volume sum into the "element" positions of the Tlc and Mlc matrices. As mentioned previously, this PWC-LC method only solves for one value in a cell since it uses a constant source approximation. But in order to use the same matrices and solving routine as for the PWL-LC method, the conservative flux and the accumulated track volume is put into the appropriate positions in the Mlc and Tlc matrices. These matrices are defined in terms of the corner values of the cell so the number of elements/corners is equal to the number of vertices in the cell. Once this compute_cell_fluxes() method is completed, all of the angular fluxes for the tracks for this angle and group have been solved.

3D PWC-LC Angular Flux Solver

The angular flux solver for the 3D PWC-LC method in (x, y, z) is found in XYZ_SLC_Method.cc in the compute_cell_fluxes() routine. The first step in this

method finds the beta and source value at the center of cell, where beta is used in the calculation of the basis functions and is equal to the inverse of the number of vertices in the cell. Next, each face of the 3D cell is looped over and its vertices are converted to *u*, ω, *s* coordinates as well as the associated *i* and *j* index values. This conversion is done by using Eqs. (5.19) and (5.21). The reverse of these equations are given in Eq. (B.12).

$$\begin{aligned} u &= \left(x - x_0\right)\frac{\mu_m}{\sqrt{1-\xi_m^2}}\xi_m + \left(y - y_0\right)\frac{\eta_m}{\sqrt{1-\xi_m^2}}\xi_m - \left(z - z_0\right)\sqrt{1-\xi_m^2} \\ \omega &= \left(y - y_0\right)\frac{\mu_m}{\sqrt{1-\xi_m^2}} - \left(x - x_0\right)\frac{\eta_m}{\sqrt{1-\xi_m^2}} \\ s &= \left(x - x_0\right)\mu_m + \left(y - y_0\right)\eta_m + \left(z - z_0\right)\xi_m \end{aligned} \tag{B.12}$$

The coordinates of the face center is also converted in this loop. After converting, the face must be checked to determine if this conversion has caused the overlapping of sides that can occur with vertex motion. This overlap is referred to as the number of "twists" on this face in the rotated coordinate system and is stored in the variable numtwists for each face. The check_face_twists() method (located in LongChar_3D.cc) determines the number of twists of this face in the rotated coordinate system. This routine loops over the vertices of this face to form the edges of the face and checks the cross product of adjacent edges. If adjacent edges have cross products of opposite signs, then a twist has occurred on the face and the numtwists vector is incremented. If there is at least one twist on a face, the sides must be checked to determine if there are incoming and outgoing sides on the face. The variables called numincident and numexit store the number of incoming and outgoing sides on this face which are determined by the dot product between the outward side normal and the given direction vector.

After this setup for each face has been done for the given cell, another loop over faces is entered. This loop is continued if the given direction is incident on the face (determined by the face normal) or the numincident variable is greater than zero. If incident, the tracks on this face are looped over. The track ID and incident angular flux are gotten from the local psi_face vector and the ID is converted to an *i* and *j* index. The incoming position of the track on this face is determined by checking the sides on the face. Before entering this loop over sides, the following variables are defined to contribute to the track position.

$$\begin{aligned} x_{0ij} &= x_0 + j\Delta u_{k,m}\frac{\mu_m}{\sqrt{1-\xi_m^2}}\xi_m - i\Delta\omega_{k,m}\frac{\eta_m}{\sqrt{1-\xi_m^2}} \\ y_{0ij} &= y_0 + j\Delta u_{k,m}\frac{\eta_m}{\sqrt{1-\xi_m^2}}\xi_m + i\Delta\omega_{k,m}\frac{\mu_m}{\sqrt{1-\xi_m^2}} \\ z_{0ij} &= z_0 - j\Delta u_{k,m}\sqrt{1-\xi_m^2} \end{aligned} \tag{B.13}$$

Inside the loop over sides, the side must only be checked if the numincident variable is greater than zero. If this is true, the incidence of the side is determined by using its outward normal vector and the loop proceeds if this angle is incident on the side.

The incoming track intersection with the side is determined by the equation for an intersection of a line with a plane using the three vertices of the side on the face. The plane the side is in can be determined by the following simple equation.

$$Ax + By + Cz + D = 0 \tag{B.14}$$

Using the three vertices of this side, these coefficients are determined as given in Eq. (B.15).

$$
\begin{aligned}
A &= y_1\left(z_2 - z_3\right) + y_2\left(z_3 - z_1\right) + y_3\left(z_1 - z_2\right) \\
B &= z_1\left(x_2 - x_3\right) + z_2\left(x_3 - x_1\right) + z_3\left(x_1 - x_2\right) \\
C &= x_1\left(y_2 - y_3\right) + x_2\left(y_3 - y_1\right) + x_3\left(y_1 - y_2\right) \\
D &= -\left[x_1\left(y_2 z_3 - z_2 y_3\right) + x_2\left(y_3 z_1 - z_3 y_1\right) + x_3\left(y_1 z_2 - z_1 y_2\right)\right]
\end{aligned}
\tag{B.15}
$$

The intersection of a track with this plane is determined by using Eq. (B.14) and Eq. (B.16).

$$
\begin{aligned}
x &= x_{0ij} + \mu_m s \\
y &= y_{0ij} + \eta_m s \\
z &= z_{0ij} + \xi_m s
\end{aligned}
\tag{B.16}
$$

Solving this system, the intersection point is given in Eq. (B.17).

$$
s = -\frac{D + Cz_{0ij} + By_{0ij} + Ax_{0ij}}{A\mu_m + B\eta_m + C\xi_m}
\tag{B.17}
$$

Plugging this intersection value back into Eq. (B.16) gives the potential *x*-*y*-*z* incoming position of this track. Next, this incoming position is checked to determine if it occurs on this side through the check_face_side() method. This method is the same as the check_face_side() method outlined in 2D from the previous section which uses cross products of the edges of the side to determine if the given point is within the side.

If the track is on this side, the incoming position is checked for potential intersection with a vertex or edge of the side and moved toward the face center if needed. This track moving routine is inside of a while loop in order to ensure the track has been moved sufficiently away from an intersection point. The check_intersect() method (located in LongChar_3D.cc) loops only over the vertices of the side on the face edge. First, the edge is checked if it is horizontal which is determined by the *i*-values of

the vertices of the side-edge. If the track is near a vertex or on this horizontal edge, the track is moved toward the face center position by a fraction of the distance to the face center from the current incoming position. If the side-edge is not horizontal, each vertex is checked separately for an intersection as well as the side-edge and the track is moved toward the face center if needed in the same manner. Note that the center point of the face (which is one of the vertices of every side) is not checked for intersection and neither are the two side-edges that touch this face center point. After potentially moving the incoming track position, this moved position is checked through the check_face_side() method again to determine if it is still on the face. If the incoming position is on this side, the loop over sides is broken, otherwise the initial values are reset and the loop over sides continues.

After determining the incoming position of the track, the track's exiting position on the cell is determined. This is done by looping over the faces of the cell again. If a face is an exit face as was determined in the exit_face vector or the numexit variable for this face is greater than zero, the sides on this exit face are looped over. If this face contains outgoing sides based upon the numexit vector, the side is checked to determine if it is an exiting side for the given direction. If the side is outgoing, the potential exiting position of the track is determined. If the track is parallel to the plane of the side, the track cannot be on this side and the track length for this side is set to zero, otherwise the intersection of this track and the plane of the side is calculated (as was done for the incoming position). This exiting position is then checked to determine if it is on the side. If on the side, the track is checked for potential intersection with vertices or the

face edge and moved if needed. If moved, the exiting position and incoming position are recalculated in order to preserve the track length across the cell and the exiting position is checked to make sure it is still on this side. Next, the track length on the cell is calculated using the differences between the beginning and exiting *s*-values. If the track has been found to be on the current side, was moved due to an intersection, and the tracklength is positive, the track length and exiting side are saved and the loop over sides is broken. If the track length is positive but hasn't been moved, this tracklength is checked with the previous track length from other side calculations and the lesser of the two is saved. This tracklength is saved on the current exiting face. Once the loop over the face's sides is exited, the tracklength on this face is compared to the overall tracklength variable, dels, which accounts for all of the exiting faces. If this face's tracklength is the smallest it is saved in dels along with saving the exiting face.

Once the shortest track length and associated exiting face have been found, the exiting angular flux and contribution to the Mlc and Tlc matrices is calculated. This section is the same as the 2D calculation done in XY_SLC_Method.cc to find the exiting angular flux and conservative flux contributions. The track ID and exiting angular flux are then put into the vectors called trackid and angflux for this exiting face and the counter for this exiting face is incremented. The volumesum that contributes to the Mlc matrix is incremented by the track volume which includes the track length on the cell and the two track spacing variables.

Once all of the incident faces and associated incoming tracks have been looped over, the values stored in the trackid and angflux vectors must be put into the psi_face

vector. This section uses the same startflux notion as in 2D to determine where to put the angular flux values in this psi_face vector. However, since some incident faces have twists, the numexit vector and mycounter vectors are checked to determine if a face has exiting tracks on it for this angle. After the values in psi_face are set, the edge_flag for the incident faces as determined by the exit_face vector must be set. This edge_flag is used in LongChar_3D to set the sister edges properly. If an incoming face has outgoing tracks on it the edge_flag is set to true so these values will be set on the buffer. Finally, the track volume and conservative flux values are put into the temporary Tlc and Mlc matrices as was done in 2D.

2D PWL-LC Angular Flux Solver

This section outlines the angular flux solving routine for the 2D PWL-LC method found in compute_cell_fluxes() located in XY_LLC_Method.cc. Since the basis functions used in this method are linear on sides of the cell, this compute_cell_fluxes() method solves for the angular flux and matrix contribution of each track along each side of the 2D (x, y) cell. This method begins by defining the beta values and source at the center of the cell, similar to the 3D PWC-LC flux solver. Next, a loop over the edges of the cell is entered. If the given angle is incident on the current edge, this loop proceeds to loop over the tracks incident on this edge as gotten from the psi_face vector. For a track, the incoming intersection point with this incident edge is found via the find_track_intersect() method which is the same method used for the 2D PWC-LC method. Next, this incoming position is checked for a vertex intersection on this edge through the check_start_intersect() method which is also the same method used in the

2D PWC-LC method. This method moves the incoming position of the track if it is within a tolerance of a vertex on the edge. Once this check is complete, the incident flux is retrieved from the psi_face vector and the starting side is set based upon the north vertex of the incoming edge.

Next, a while loop is entered that follows the track through sides of the cell until it exits the cell. A counter variable is used, called *it*, to keep track of the number of sides the track has traversed. Inside this while loop, first the vertices of this side are set based upon which side-edge the track is entering the side. Remember the sides of a cell in 2D (x, y) are triangles. Next, the potential side exit points on the other two side-edges of the side are calculated via the find_track_intersect() method and stored in point1 and point2. The shortest distance from the incoming position on this side is then found using these two exiting points. If the slope of the track is parallel to a side-edge, the track cannot intersect it and therefore must intersect the other edge. For a track that is parallel to one side-edge, the center and vertices of the other side-edge must be checked for a potential intersection. If near the center, the check_center() method is called to move the track. If near a vertex of the side-edge, the check_vertex() method is called to move the track. If the potential exit point is not near a vertex, the exit position is saved along with the side-edge ID and the track length on this side (found using Eq. (B.7)). This side-edge ID is used to determine the incoming side-edge for the next side the track passes through.

The check_center() method moves the track along the given side-edge away from the center point using a fraction of the side-edge length. A variable called track_move stores the amount the track has been moved in x and y. Next, this method recalculates

the incoming position on the side in order to preserve the track length on the side using the find_track_intersect() method. Finally, the new distance between the moved exiting and incoming points on the side is calculated along with the track length. The check_vertex() method moves the exiting position of the track on the side-edge away from the given vertex using a fraction of the side-edge length. The track_move variable is updated by the amount moved and the new incoming position is found using the find_track_intersect() method. Finally, the new distance between points and the track length on the side is calculated and stored. Both the check_center() and check_vertex() methods also save the ID of the side-edge the track exits the side.

Returning to determining the exiting position of the track on a side, if the track is not parallel to either side-edge the distances using the two exiting points could be almost equal. There are several cases that can cause this to occur, but again the center and vertices must be checked for potential intersection. If the track exit position is not near any vertex, one of the exiting points is picked and checked to determine if it is on the side by calling the find_track_length() method. This method determines if the exit point is within the given side-edge and if not, picks the other exit point as the exit point that occurs on the side. It first checks if the given side-edge is a vertical line. If so, the *y*-coordinate of the exiting point is checked to determine if it is outside the side-edge. If it is outside of the edge, the track length, exiting position, and edge ID of the other exiting point are saved. If not a vertical edge, the *x*-coordinate of the given exiting point is checked to determine if it is outside of the side-edge. If it is outside of the edge, the other exiting point is saved.

Next, if one exiting distance is less than the other, the closer exiting position is checked to determine if it occurs on the side-edge by calling the check_on_edge() method. This method checks whether the given exiting point is outside of the given side-edge, including the special cases of a horizontal or vertical side-edge. If the exit point is outside of the side-edge, the onedge flag is set to false and this method returns this flag. If the track exits on this side-edge, the center and vertices are again checked for a potential intersection and the track length and exiting position on the side are saved. If not near a center or vertex, the exiting position is also checked again via the find_track_length() method. Finally, if the track exit position does not occur on the side-edge, the other exiting point is chosen as the shortest distance and the same intersection checking procedure follows.

After finding the track length on the side, the source at the beginning and exiting points of the track on the side is found and stored in the variables called avgsourceB and avgsourceE. These source values are found by interpolating between the source values at the vertices of the incoming and exiting edges of the side using the incoming and exiting positions of the track. Therefore, this section uses the edge ID's that have been stored for the incoming and exiting track position on the side in order to know which vertex or center source values to use. Next, the attenuation factor, tau, is calculated as was done in the 2D PWC-LC method and the exiting angular flux and contribution to the conservative flux and least-squares matrices are calculated for this side.

As was done in the 2D PWC-LC method there is a separate loop for small tau to find the exiting angular flux and matrix contributions from this side which uses the same

approximation for the exponential function as given in Eq. (B.11). The loop for a regular tau is reviewed here because the loop for a small tau is similar, with different approximations for expressions involving the exponential function. In this section, the least-squares set of equations given in the previous sections for this 2D PWL-LC method is used to calculate the appropriate contributions to the Tlc and Mlc matrices. Before entering a loop over the elements for this cell, which is also the number of PWL basis functions on the cell, some common terms are calculated.

The exiting angular flux along a track is expressed using Eq. (B.8). This expression can be written in terms of the following variables when multiplied by a basis function:

$$b_i \Psi = f_{k,inc}^{i} \psi_{k,m,inc} + f_{k,B}^{i} q_{k,B} + f_{k,E}^{i} q_{k,E}, \tag{B.18}$$

where,

$$\begin{aligned} f_{k,inc}^{i} &= \int_0^{\Delta s_k} ds b_i(s) e^{-s\sigma_t} \\ f_{k,B}^{i} &= \int_0^{\Delta s_k} ds b_i(s) \int_0^{s} ds' \frac{\Delta s_k - s'}{\Delta s_k} e^{-(s-s')\sigma_t} \\ f_{k,E}^{i} &= \int_0^{\Delta s_k} ds b_i(s) \int_0^{s} ds' \frac{s'}{\Delta s_k} e^{-(s-s')\sigma_t} \end{aligned} . \tag{B.19}$$

The PWL basis function can be written in terms of the *s* variable as given in Eq. (B.20), where $b_i(s=0)$ refers to the value of the basis function at the incoming track position on a side and $b_i(s=\Delta s_k)$ refers to the value of the basis function at the exiting track position on a side.

$$b_i(s) = b_i(s=0) + \frac{s}{\Delta s_k}\left(b_i(s=\Delta s_k) - b_i(s=0)\right) \tag{B.20}$$

Using this expression for a basis function, the integrals in Eq. (B.19) can be performed and yield the following expressions for a large value of tau. Equation (B.10) has been applied and the basis function values at the incoming and exiting positions on a side have been shortened to b_i^0 and $b_i^{\Delta s_k}$.

$$f_{inc}^i = \frac{b_i^{\Delta s_k} + b_i^0\left(\tau_k - 1\right) - e^{-\tau_k}\left(b_i^{\Delta s_k}\left(1+\tau_k\right) - b_i^0\right)}{\tau_k^2} \tag{B.21}$$

$$f_B^i = b_i^0 \frac{e^{-\tau_k}\left(1+\tau_k\right) + \frac{\tau_k}{2} - 1}{\sigma_t \tau_k^2} + \left(b_i^{\Delta s_k} - b_i^0\right)\frac{\frac{\tau_k}{2} - \tau_k + \frac{\tau_k^3}{6} - 1 + e^{-\tau_k}\left(1 + 2\tau_k + \tau_k^2\right)}{\sigma_t \tau_k^3} \tag{B.22}$$

$$f_E^i = b_i^0 \frac{1 - \tau_k + \frac{\tau_k}{2} - e^{-\tau_k}}{\sigma_t \tau_k^2} + \left(b_i^{\Delta s_k} - b_i^0\right)\frac{1 - \frac{\tau_k}{2} + \frac{\tau_k^3}{3} - e^{-\tau_k}\left(1+\tau_k\right)}{\sigma_t \tau_k^3} \tag{B.23}$$

There are similar expressions for small tau with the same approximation for the exponential function as was given in Eq. (B.11). From the expressions in Eq. (B.21)-(B.23), several variables are calculated before the loop over elements on the cell: att, fB1, fB2, fE1, fE2. Rewriting these expressions using these variables yields the following equations.

$$f_{inc}^i = \frac{b_i^{\Delta s_k} + b_i^0\left(\tau_k - 1\right) - \mathrm{att}\left(b_i^{\Delta s_k}\left(1+\tau_k\right) - b_i^0\right)}{\tau_k^2} \tag{B.24}$$

$$f_B^i = b_i^0 \mathrm{fB1} + \left(b_i^{\Delta s_k} - b_i^0\right)\mathrm{fB2} \tag{B.25}$$

$$f_E^i = b_i^0 \text{fE1} + \left(b_i^{\Delta s_k} - b_i^0 \right) \text{fE2} \tag{B.26}$$

After defining these variables, a loop over elements in the cell is entered to contribute values to the Tlc and Mlc matrices.

The value of a basis function at the incoming and exiting positions of the track on the side is found by interpolating between vertex values, similar to the source interpolation. These basis function values are found through a call to the find_basis_values() method. Based upon the given basis function ID and the side-edge the tracks exits or enters, this method interpolates between the values of the basis function at the vertices of the side-edge and the incoming or exiting position. Remember the value of every basis function at the center of the cell is equal to beta, and the value at a vertex of the cell is equal to unity or zero. The value of the basis function at the incoming track position on the side is stored in the variable b_0 and the value at the exiting track position on the side is stored in the variable b_1. Using these basis function values, the function values defined in Eq. (B.24)-(B.26) are found and the conservative flux values stored in the Tlc matrix are contributed to for this element using Eq. (B.18) and the equations from the implementation algorithm section.

Next another loop over elements is entered, and the contribution to the Mlc matrix values are found and stored. The find_basis_values( ) method is called again to find the value of the current element's basis function at the incoming and exiting position of the track on this side. These basis function values are stored in the variables *b0j* and *b1j*. Using these basis values, the Mlc matrix value for the (*i*, *j*) position is

found. These matrix values come from the 2D version of Eq. (5.7). Evaluating the integral in this equation for the PWL-LC method in 2D leads to the following equation.

$$\Delta\omega_{k,m}\int_0^{\Delta s_k} ds b_i b_h = \Delta\omega_{k,m}\int_0^{\Delta s_k} ds\left[b_{i0}+\frac{s}{\Delta s_k}\left(b_{i1}-b_{i0}\right)\right]\left[b_{j0}+\frac{s}{\Delta s_k}\left(b_{j1}-b_{j0}\right)\right]= \\ \Delta\omega_{k,m}\frac{\Delta s_k}{6}\left[2b_{i1}b_{j1}+2b_{i0}b_{j0}+b_{i0}b_{j1}+b_{i1}b_{j0}\right] \tag{B.27}$$

After calculating and storing this value in the cell Mlc matrix for all element combinations, the exiting angular flux is calculated as given in Eq. (B.28). This equation is similar to Eq. (B.18) but not multiplied by a basis function. The exiting angular flux for this track on this side is stored in the psi_inc variable to use on the next side the track traverses in the cell.

$$\psi_{k,m,exit}=\psi_{k,m,inc}e^{-\tau_k}+q_{k,B}\frac{\Delta s_k}{\tau_k^2}\left(1-e^{-\tau_k}\left(\tau_k+1\right)\right)+q_{k,E}\frac{\Delta s_k}{\tau_k^2}\left(e^{-\tau_k}+\tau_k-1\right) \tag{B.28}$$

Finally, the next side the track enters on the cell is determined and the track starting position on the new side is set to the exiting position of the previous side. If it exits the cell on this side, the current while loop over sides is exited. Upon exiting the cell, the track ID is put into the mytracks vector for the exiting face and the exiting angular flux is put into the myfluxes vector for the exiting face associated with this track.

Once all of the incoming tracks on each incoming face have been looped over and checked, the mytracks and myfluxes vectors need to be put into the local psi_face vector for each edge. This is done by looping over the edges of the cell and putting the mytracks and myfluxes vectors for each edge into the psi_face vector. Remember the psi_face vector includes one value for the total number of tracks for a given angle

followed by the track ID's on the edge and then a list of the associated angular flux values for each track. If no tracks exited on an edge, the first two values in the psi_face vector are set to zero. This concludes the detail of the compute_cell_fluxes( ) routine for the 2D PWL-LC method.

3D PWL-LC Angular Flux Solver

This section outlines the angular flux solving routine for the 3D PWL-LC method found in compute_cell_fluxes() located in XYZ_LLC_Method.cc. Since the basis functions used in this method are linear on sides of the cell, this compute_cell_fluxes() method solves tracks along each side of the 3D (x, y, z) cell. The details for this computing method are similar to the 2D PWL-LC and 3D PWC-LC routines but slightly more complicated due to the 3D side geometry. This method begins by getting the alpha value from the cell and storing it in a vector for each element. This alpha value is based upon the expression for the PWL basis function in 3D. Then the vertices of each face are converted to the rotated coordinates (u, ω, s) as well as i and j index values. This conversion is done by using Eq. (B.12). The face center coordinates are converted and stored as well in the rotated_fc and index_fc vectors. For each face, the number of elements (tracks) is stored in the num_elements vector. Finally, after exiting the loop over faces, the cell center coordinates are converted and stored in the rotated_zc and index_zc vectors.

To begin retrieving the incoming tracks on this cell, another loop over faces is entered. The current face is checked for "twists" as was done with the 3D PWC-LC method via the check_face_twists( ) method. If there are twists present on this face for

the current angle, the outward edge normals of each side on the face are checked to determine the number of incoming and outgoing side-faces occurring on this cell face. The next loop is only entered if the given angle is incident on the current cell face or the number of incident side-faces is greater than zero. The number of tracks on this face is gotten from the local psi_face vector and the tracks on this face are then looped over.

For each track incident on this face, the track ID and incoming angular flux is gotten from the local psi_face vector. The track ID is then converted to an (i, j) index as done in previous methods. Then the variables used to define the track position as given in Eq. (B.13) are found. The incoming position of the track on this face is found by checking the sides on this face. If a side is determined to be incident based upon its edge normal and the given angle, the coordinates of this side-face are saved and the potential incoming position of the track on this side-face is found by using Eqs. (B.15)-(B.17). This potential incoming position is then checked to determine if it occurs on this side-face by calling the check_face_side( ) method. This method has already been outlined previously and uses cross products of the edges of the side to determine if a given point lies within the side. If the incoming position for this track occurs on this side-face, the coordinates of the side-face center are calculated and stored in the side_face_fc variable. The incoming position is then checked to determine if it needs to be moved due to an intersection with a side-face edge or side-face vertex. Refer to the outline from the 3D PWC-LC method for the detail of the rest of the side loop to determine the incoming track position on the cell. Note that a difference with the PWC-LC method is that the track is moved toward the side-face center instead of the cell face center if necessary.

After determining the incoming position of the track on the cell, a loop over sides is entered to follow the track trajectory though the cell's sides. A while loop over traversed sides is entered and breaks when the track exits the cell. This loop assumes the current side ID the track is in is known. The faces of the current side are given an ordering as follows:

SF0 contains vertex 1, vertex 2, cell center,

SF1 contains vertex 1, face center, cell center,

SF2 contains vertex 2, face center, cell center,

SF3 contains vertex 1, vertex 2, face center.

These ID's are used to determine the exiting face of the side for the track. Next, the (*x*, *y*, *z*) coordinates as well as the rotated coordinates and indices of the vertices that make up the side are saved in vectors to easily loop through (they have previously been calculated). Finally, a loop over side-faces is entered to determine the distances from the incoming track position to the other faces of the side. The coefficients to make the equation for the plane of the each face are calculated using Eq. (B.16). If the track is parallel to a side-face, it cannot intersect, so the track length to this side face is set to zero via the *delta_s* vector. Also, the track cannot exit out of the side face it entered, so the track length to this side face is also set to zero. If neither of these cases is true, the potential outgoing position of the track on the side face is calculated using Eqs. (B.15)-(B.17) and stored in the variables x1, y1, and z1.

Next, the center coordinates of the side face are calculated and saved and the exiting position is checked to determine if it occurs on the side face by calling the

check_face_side( ) method. If the outgoing position occurs on the current side face, the track is checked to determine whether it needs to be moved via the check_intersect( ) method as was done for the incoming track position. If the track is moved, the incoming position on the side is recalculated using the same equations. Note that the movement of the track occurs inside a while loop to ensure the track has been moved sufficiently away from an intersection with a vertex or edge. The final exiting position is checked to make sure it still occurs on the side face.

If the exiting position occurs on the current side face the track length is calculated and stored in the delta_s vector, otherwise this track length is set to zero. Next, the track length needs to be checked against the lengths to the other side faces to determine if the side face loop needs to continue. If the exiting track position was moved due to an intersection the *trackdone* variable was set to true. If this exiting position remained on the face and the track length on the side is greater than zero, the side face loop is broken and variables are saved for the remaining calculations. However, if the exiting position is not on the side face or the track length is negative, variables are reset to old values to continue the side face loop. For tracks that did not need to be moved due to an intersection, the value in the delta_s vector for the current side face is compared with the value stored in the dels variable and is saved if it is smaller. The side face loop then continues until the smallest positive track length on the side is found.

Once the shortest track length and associated exiting face are found, the exiting angular flux and contributions to the Mlc and Tlc matrices are calculated. This begins

with the calculation of the tau variable. Next, the source values at the beginning and ending positions of the track on the side are found by calling the interpolate_source( ) method. The interpolate_source( ) method uses the actual form of the basis function to calculate its value at the given (*x*, *y*, *z*) point. The form of a PWL basis function in 3D is given in Eq. (B.29):

$$b_j(x,y,z) = t_j(x,y,z) + \alpha_{C,j} t_C(x,y,z) + \sum_{\text{faces at j}} \beta_{f,j} t_f(x,y,z), \tag{B.29}$$

where,

$$\begin{aligned} \alpha_{C,j} &= \frac{1}{N_{cell}} \\ \beta_{f,j} &= \frac{1}{N_{face}} \end{aligned}. \tag{B.30}$$

The value of *N* in Eq. (B.30) is the number of vertices on a cell or face. Each function in Eq. (B.29) is linear as given in Eq. (B.31).

$$t_*(x,y,z) = a_* + h_* x + d_* y + g_* z \tag{B.31}$$

The coefficients in Eq. (B.31) are found by using the definition of the PWL bases. On a given side, the standard function, t_j, has a value of one at the j^{th} vertex of the side and decreases linearly to zero on all other vertices of each side that touches vertex *j*. The t_c function is one at the cell midpoint and zero at each face midpoint and each cell vertex. The t_f function is one at the face midpoint and zero at the cell midpoint and each of the face vertices. With this information, a set of equations can be solved for all of the coefficients needed to fully form the basis given in Eq. (B.29). These coefficients are

calculated in the interpolate_source( ) method and the local *t* functions are formed at the given interpolation point.

Remember the source for this method is written in terms of these PWL bases given above. Inside the interpolate_source() method, a loop over elements is entered to add the contribution to the interpolated source value from each *N* source value. Note that if the interpolated point occurs on SF3, which is on a face of the cell, the contribution from the t_c function is zero and only the elements on the face need to be looped over. After this source method is complete, the intvalue variable contains the value of the source at the interpolated point (the incoming or exiting track position).

Next, a loop over elements is entered and the value of the basis function at the track's incoming and exiting positions is calculated by calling the find_basis_values( ) method. This routine is very similar to the interpolate_source( ) routine. After calculating the local basis functions, a loop over elements is entered to add the contribution from all basis functions in the cell at the interpolated point. The only difference with the source routine is the value of the source at each vertex is not included because only the value of the basis function is needed. These methods should be combined in the future.

Returning to the compute_cell_fluxes( ) method, a separate loop is used for small tau to determine the same coefficients for the Tlc matrix contribution as was done in the 2D PWL-LC method. The same variables defined in Eqs. (B.18)-(B.26) are calculated in this method as well. After determining the expression for these variables, the contribution to the Tlc matrix is saved based upon Eq. (B.18). Another loop over

elements in then entered to find the contribution to the Mlc matrix. The find_basis_values( ) method is called again to find the current element's basis function values at the incoming and exiting track positions on the side. Then the contribution to the Mlc(i, j) matrix position is added using Eq. (B.27) with the extra track spacing for 3D. Finally, the exiting angular flux for this track on the side is calculated using Eq. (B.28).

The next side the track enters is determined based upon the side face ID the track exited (still inside the while loop for the track exit). If the track exited out of SF3, the track has exited the cell and the trackexit_flag is set accordingly to break the while loop. If the track is not exiting the cell, the next sideID the track enters is found based upon the face it is exiting by using the data saved for each side about its neighboring sides. After setting values needed for a continuing track in the cell, the while loop over sides follows the track and contributes to the Mlc and Tlc matrices until it exits the cell.

After a track has exited the cell, the outgoing angular flux is stored in the angflux vector for the exiting face and the track ID is saved in the trackid vector. The counter for the exiting face is also incremented for use in putting values in the psi_face vector. Once all of the incoming tracks have been traversed through the cell, the angflux and trackid values must be put into the local psi_face vector for each exiting face. This is done by looping over the faces of the cell. If the mycounter vector for a face is greater than zero, the value in the mycounter vector is put into the first value of the psi_face vector for the face to set the number of tracks exiting on this face. Then the number of tracks is looped over using this counter and the track ID in the trackid vector and the

associated exiting angular flux in the angflux vector are put into the appropriate positions in the psi_face vector. If the given face is an exit face but no tracks exit on it, the first value in the psi_face vector is set to zero. Also, if this face has some exiting sides on it (as given by the numexit vector), the tracks exiting on this face are set in the psi_face vector as well. Finally, this method must set the edge_flag for the exiting face so the sister edge can be set appropriately in the rest of the sweepchunk. If the face is not an exit face based upon its face normal and the given angle but the mycounter vector is greater than zero, the edge_flag for this face and angle is set to true. If the mycounter vector has zero tracks in it for this face, the edge_flag is set to false.

After completing this compute_cell_fluxes( ) routine for the 3D PWL-LC method for a given groupset and angleset, the exiting angular fluxes for each track on the cell and the contributions to the Mlc and Tlc matrices have been calculated and saved.

2D PWC-STLC Angular Flux Solver

This section outlines the angular flux solving routine for the 2D PWC-STLC method found in compute_cell_fluxes() located in XY_STEPSTLC_Method.cc. This routine is different from the 2D LC flux routines because it must also calculate fluxes and matrix contributions for tracks incoming and exiting on the time boundary of the space-time cell. It begins by looping over the spatial faces of the cell and proceeds only if the face is incoming for the given angle. For an incident face, the tracks incoming on this face are looped over by using the values stored in the psi_face vector. The track ID gotten from this vector is converted to an *i* and *j* index using the equations defined in Eq.

(5.20). Next, the incoming (*x*, *y*) position of this track on the face is calculated by calling the find_track_intersect( ) method.

The find_track_intersect( ) method is located in STLC_2D.cc. It is very similar to the same method in LongChar_2D.cc with some additions. The details of the 2D LC find_track_intersect( ) method can be found in the ***2D PWC-STLC Angular Flux Solver*** section to explain how the spatial intersection is found. After calculating the spatial position of the track intersecting an edge, the *s*-value of this intersection is also calculated in this method by using Eq. (B.4).

After finding the starting *s*-value of the track, the starting position in time is calculated using Eq. (5.22). In order to check if the track needs to be moved due to an intersection, the vertices of this face are converted to an *i*-index and an *s*-value using Eq. (B.3) and Eq. (B.4). In doing so, it is necessary to check that the correct sign of the square root used to find the *s*-value has been taken. Next, the check_intersect( ) method is called to determine if the track intersects a vertex or edge of the face and move the track if needed.

The check_intersect( ) method is also similar to the check_intersect( ) method located in LongChar_3D.cc and outlined in the ***3D PWC-LC Angular Flux Solver*** section. It begins by calculating a face center point for the cell's spatial face. This face is rectangular with the *vt* coordinate of the center equal to the midpoint of the time step. Next, the *i*-index of the track is checked for an intersection with one of the vertices of this edge. If this is true, the *j*-value of this vertex must be found at the beginning and end of the time step. If the track is near either of these points it is moved toward the

center of the face by a fraction of the distance from the incoming position to the face center. If the track is not near one of these *j*-values but it is near the *i*-index of the vertex it must be along the edge between the *j*-values and is therefore moved away from this edge in the same manner. The same check is performed for the other spatial vertex of this face. Finally, the *j*-values on the time boundaries for the given *i* index of the track are calculated. If the *j* index of the track is near these *j*-values, it is moved toward the face center in the same manner. The moved variable is set to true if the track is moved in this method. Returning to the main compute_cell_fluxes( ) method, if the track has been moved, the index values for the track are updated using Eq. (B.3) and Eq. (B.5).

Next, the distances from the incoming position to the exiting faces of the cell are calculated and compared to find the shortest distance. First, the spatial edges of the cell are looped over. If the spatial edge is an exit face for the given angle, the loop proceeds to calculate the intersection of this track with this edge by using the find_track_intersect() method. The distance between this exiting position and the incoming track position is then calculated and put into the distances vector. The slope of the spatial coordinates of this exiting face is also saved along with the *i*-indices of the vertices of the exit face.

After all exiting spatial faces have been looped over, the shortest distance in the distances vector must be determined for this track. A loop over the size of this distances vector is entered which essentially loops over the potential exiting faces for this track. This loop proceeds only if the track is not parallel to this exiting face. The s-values of

this exiting face's vertices are then calculated along with the exiting *vt* coordinate on the face. Then the check_intersect( ) method is called to move the track if necessary as was done for the incoming position. If the track has been moved, the track indices are recalculated and the incoming position is recalculated using these new indices by calling the find_track_intersect( ) method. The new distance between the moved incoming and exiting positions is then calculated and stored in the mydistance vector. If the track is not moved, the previously calculated distance from the distances vector is stored in the mydistance variable. This mydistance value is compared with the value stored in the distance variable and if it is smaller, this distance is saved along with the exiting face.

After determining the shortest distance to a spatial edge of the cell, this distance must be compared to the distance to the time boundary. This distance to the time boundary is calculated by using the starting position and the time for the end of the time step as given in Eq. (B.32).

$$length = \frac{vt^{n+1} - vt_{k0}}{\sqrt{\mu_m^2 + \eta_m^2}} \tag{B.32}$$

Next, the timeflag boolean is set to false to assume the track does not exit the cell out of the time boundary. If the shortest spatial distance is greater than the distance to the time boundary from the incoming position, which is stored in the distance_time variable, the timeflag is set to true and the track length is calculated as the difference in vt0 and the end of the time step as given in Eq. (B.33).

$$\Delta s_{k,m} = vt^{n+1} - vt_{k0} \tag{B.33}$$

If the spatial distance is shorter than the distance_time variable, the track length is calculated as the usual spatial distance as given in Eq. (B.34) which is the same as Eq. (B.7).

$$\Delta s_{k,m} = \frac{dist}{\sqrt{\mu_m^2 + \eta_m^2}} \tag{B.34}$$

The next step is to calculate the contribution to the average angular flux in the cell and the exiting angular flux for this track. First the average source at the incoming and exiting track positions is found. At the time of this work, PDT only allowed constant values for the fixed source and the scattering source for this PWC-STLC method is assumed to be a constant, therefore there is only one value for the angular source in the cell for a given angle. This calculation of the source values will need to be changed to an interpolation routine to allow for other types of fixed sources in the future. The tau variable is then calculated as the attenuation factor and the incoming angular flux is gotten for this track from the psi_face vector. Again as with previous LC methods, there is a separate loop for a small value of tau in order to use an approximation for the exponential function as given in Eq. (B.11). The same equations as were used for the 2D PWC-LC method given in Eqs. (B.8) and (B.9) are used here to find the exiting angular flux and contribute to the average conservative angular flux as stored in the temp and psi_cell variables. Note that the Jacobian term is not present in the calculation done here because it is included in the STLC_2D sweepchunk when values are put into the Tlc matrix on the cell.

After determining the exiting angular flux, the track ID and exiting flux must be put into the appropriate vectors. If the track exits on the time boundary, these values are put in as a pair into the time_tracks vector. If the track exits on a spatial boundary, these values are put into the mytracks and myfluxes vector for the appropriate exiting face. Finally, the contribution to the track volume summation in the cell (stored in the volumesum variable) is added using this track length.

Once all incoming spatial edges have been looped over, the tracks incoming on the time boundary are looped over. The first step is getting the number of tracks incoming on this boundary as stored in the first element of the psi_time_inc vector. This vector stores track ID's and incoming angular flux values as pairs. Next, the spatial coordinates of the cell are gotten and converted to *i* and *j* indices as stored in the index_values vector by using Eqs. (B.3)-(B.5). Then, the tracks incoming on the time boundary are looped over. The track ID is gotten from the psi_time_inc vector and converted to an *i* and *j* index using Eq. (5.20). Some variables are then defined to contribute to finding the incoming track position on the time boundary as given in Eq. (B.35).

$$\begin{aligned} x_{0ij} &= x_{offset} - i\Delta\omega_{k,m}\frac{\eta_m}{\sqrt{\mu_m^2+\eta_m^2}} \\ y_{0ij} &= y_{offset} + i\Delta\omega_{k,m}\frac{\mu_m}{\sqrt{\mu_m^2+\eta_m^2}} \\ vt_{0ij} &= vt_{offset} + \frac{j\Delta u_{k,m}}{\sqrt{\mu_m^2+\eta_m^2}} \end{aligned} \tag{B.35}$$

The incoming position in time is known and set to be the value at the incoming time step as stored in the *vt0* variable.

Next, the incoming spatial position of the track on the time boundary is calculated by forming the equation of the time boundary plane and finding the intersection with the track. This is done by using a modification of Eqs. (B.14)-(B.17) where the *z* variable is replaced by the *vt* variable. After finding the *s*-value of intersection, the incoming spatial position of the track is then defined using Eq. (B.35) and Eq. (5.22). The next step is to determine if this track needs to be moved due to an intersection with a vertex or edge. This potential track movement is done by calling the check_time_intersect( ) method.

The check_time_intersect( ) method is very similar to the check_intersect( ) method that is in the LongChar_3D.cc sweepchunk and is detailed in the outline for the ***3D PWC-LC Angular Flux Solver***. This method first calculates the (*x*, *y*) coordinates of the center of the time boundary of the cell. Then it loops over the vertices of the time boundary to form edges of this face. First, the edge is checked if it is horizontal by the *i*-values of the vertices of the edge. If the track is near a vertex or on this horizontal edge, the track is moved toward this face center by a fraction of the distance to the face center from the current incoming position. This distance must be checked with the size of the time step ($v\Delta t$). If the distance is too large, it is reset to a fraction of the size of the time step. The new track position is then calculated and the loop over vertices on this face is broken. If the edge is not horizontal, both vertices are checked for an intersection. If an intersection occurs, the track is moved in the same manner. Finally, to check if the track

intersects the edge, the find_track_intersect( ) method is called to find the *s*-position this intersection may occur and a potential *j*-index on the edge is calculated to compare to the track index. If an intersection occurs on this edge, the track is moved in the same manner and the loop over edges is broken. Once leaving this method, the spatial coordinates of the track have been updated if needed and the moved Boolean has been set accordingly.

If the track has been moved, the updated track indices are calculated as well as the x_{0ij} and y_{0ij} track values. The next step is to determine the exiting position of this track on the cell. This part of the loop follows the same procedure as was done in the spatial edge loop, with the only difference being if the track is moved and incoming position needs to be recalculated. This recalculation uses the equation of the time boundary plane as was done to find the incoming position on the time boundary. After determining the exiting position of the track, the same procedure as in the previous spatial edge loop is used to find the exiting angular flux and the contribution to the average angular flux. This is followed by putting the tracks in the appropriate vectors depending on where they exit the cell and contributing to the track volume summation of the cell as was done previously.

Finally after looping over all incoming tracks on the cell, the values stored in the local mytracks and myfluxes vectors are put into the my_global_tracks and my_global_fluxes vectors for the given angle and energy group. This is done by looping over the exiting spatial edges of the cell and putting these values into these vectors for all exiting faces that have exiting tracks on them as previously calculated. Lastly, this

method loops over the number of elements on the cell, which remember is $2N$ if N is the number of spatial elements. This loop puts the conservative average angular flux value into each element of the Tlctemp matrix and the track-volume summation into each diagonal element of the Mlctemp matrix. This PWC-STLC method only solves for one average value in the cell, but in order to use the same scalar flux solver for all STLC methods, these matrices are used for convenience. At the end of this compute_cell_fluxes( ) method, all of the exiting angular fluxes for the tracks on the cell for this angle and energy group have been computed and stored.

2D PPWL-STLC Angular Flux Solver

This section outlines the angular flux solving routine for the 2D PPWL-STLC method that is found in compute_cell_fluxes( ) in XY_PWLSTLC_Method.cc. This routine is similar to a combination of the 2D PWC-STLC fluxes routine, and the 2D and 3D PWL-LC fluxes routines. It begins by defining the beta values for each spatial element of the cell and the source value at the center of the cell at the beginning and end of the time step. These average source values, stored in the source_z and source_zn1variables, are an average of the source values at the spatial elements of the cell at the beginning and ending of the time step. Next, a loop over the spatial edges of the cell is entered. If the given angle is incident on this face, this loop proceeds for a given edge. The number of tracks incident on this face are gotten from the first entry of the psi_face vector and are then looped over. The track ID is retrieved from this psi_face vector as well and is converted to an i and j index as defined by Eq. (5.20).

The incoming position of this track is determined next. This is done by calling the find_track_intersect( ) method which was detailed in the ***2D PWC-STLC Angular Flux Solver*** section. The incoming position in time is again determined by using Eq. (5.22). The vertices of the face are next converted to an *i*-index and an *s*-value using Eq. (B.3) and Eq. (B.4) in order to check for intersection with this track. The check_intersect( ) method is then called to check for intersection with a vertex or edge of the face and move the track toward the face center if needed. This method was also outlined in the previous section. If the track has been moved, the index values of the track are recalculated using Eq. (B.3) and Eq. (B.4).

Once the incoming position is determined, the track must be traversed across the sides of the cell. The incident angular flux for this track is retrieved from the psi_face vector and saved in the psi_inc variable. Also, the starting side is set based upon the ID of the north vertex of the face. Next, a while loop over the sides the track traverses is entered. This while loop exits when the track exits the space-time cell. The vertices of the other edges of the side the track enters on are retrieved and stored based upon where the track enters the side. This logic is similar to the logic used in the 2D PWL-LC flux method. Next, these vertices are converted to *i*-indices and *s*-values. Remember that each side of the space-time cell is a triangular prism. The intersection points with the two other spatial edges of the side are calculated next by calling the find_track_ intersect( ) method. The spatial intersection points are stored in the variables point1 and point2 with the exiting positions in time being stored in the vtexit1 and vtexit2 variables. Also, the distance between these spatial points and the incoming track position (stored in

the trackstartpos variable) is calculated and stored in the variables distance1 and distance2.

The next step is to determine which of these two spatial distances is shorter. First, if the given angle is parallel to one edge, the shortest distance must be given by the other edge. Therefore, this other edge is checked for an intersection with the given point by calling the check_intersect( ) method. Again, this method is the same as the one in XY_STEPSTLC_Method.cc and is detailed in the previous section. If the track is moved in this routine, the incoming position on the side must be recalculated using the usual equations and the new distance between incoming and exiting positions is also recalculated. The track length is calculated using the distance to this edge and saved in the variable dels. The exiting spatial edge ID and the exiting position in time on this edge are also saved in the intersect and vt1 variables respectively. Second, if the track is not parallel to an edge, distance1 and distance2 are checked to determine if they are almost equal. If they are almost equal, one of them is chosen as the exiting edge and the usual check for intersection and moving of the track is performed. If the track is not moved on this current side edge, the find_track_length( ) method is called to determine if the distance to the other side edge should be used.

The detail of the find_track_length( ) method is the same as the method used in the 2D PWL-LC method and can be found in that section above. Finally, if one distance is shorter than the other distance, the associated point is checked to determine if it occurs within the spatial edge by calling the check_on_edge( ) routine. This check_on_edge( ) routine is also the same as the method that is detailed in the 2D PWL-LC method

compute_cell_fluxes( ) section above. If the exiting position does occur on the edge, the track is checked for an intersection and moved as usual. If the exiting position does not occur on the edge with the shorter distance, the other spatial edge is chosen and the track is checked for an intersection and moved as usual. After the shortest track length has been determined based upon the exiting spatial edges of the side, this length must be compared with the track length to exit out of the time boundary. This track length to the time boundary is calculated by using Eq. (B.33) and is stored in the variable dels_time.

If the shortest distance based on spatial edges stored in the dels variable is greater than the distance to the time boundary, dels_time, the track has exited out of the time boundary and the exiting position must be checked for potential intersection with a vertex. First, the track length in dels_time is put into the dels variable and the timeflag is set to true. Next, the *i* and *j* index values of the side vertices on the ending time boundary of the cell are calculated. Then the exiting position on this time boundary is calculated using the equation of the time boundary plane and the equation for the track similar to Eqs. (B.14)-(B.17) where the *z* variable is replaced by the *vt* variable. After finding the *s*-value of intersection, the exiting spatial position of the track is then defined using Eq. (B.35) and Eq. (5.22). This exiting position is then checked for intersections with vertices and the track is moved if needed by calling the check_time_intersect( ) method. This method is the same as used for the PWC-STLC method and the details are outlined in the previous section. Note that the track is moved, if needed, toward the center of the side on this time boundary in this method. If the track has been moved, the

incoming position of the track is recalculated using the usual intersection equations and the new track length on the side is calculated and stored in the dels variable.

Next, the source values at the beginning and exiting positions of the track on this side are calculated. This is done by an interpolation between the source values at each element of the cell. The representation of the total source for the track can be written in terms of the *s* variable as given in Eqs. (B.36)-(B.37),

$$q_{tot}(s) = \left[\frac{t^{n+1} - t_b}{\Delta t^n} + \frac{s}{\Delta s}\left(\frac{t_b - t_e}{\Delta t^n}\right)\right]\left[\Delta q_b + \frac{s}{\Delta s}\left(\Delta q_e - \Delta q_b\right)\right], \\ + \left[q_b^{n+1} + \frac{s}{\Delta s}\left(q_e^{n+1} - q_b^{n+1}\right)\right] \tag{B.36}$$

$$\Delta q_b = q_b^n - q_b^{n+1} \\ \Delta q_e = q_e^n - q_e^{n+1}, \tag{B.37}$$

where *b* represents the value at the incoming or beginning track position on the side and *e* represents the value at the exiting track position on the side. From these equations, the source value must be found at the beginning and exiting time boundaries of the time step for the beginning and exiting track spatial positions on the side. First, the source at the beginning and end of the time step is found at the incoming track spatial position on the side. These source values are stored in the sourcen_b and sourcen1_b variables to correspond to the *n* and *n*+1 time values of the time step. If the track did not enter on the starting time boundary, the beginning source values are simply an interpolation between the source values at the vertices of the side edge the track entered the side. Note that the angular source has 2*N* components to account for different values at the starting and ending time boundaries in the cell.

If the track entered the side on the time step boundary (which is never true in this spatial edge loop but is true in the next loop over tracks incoming on the time boundary), a more complicated interpolation using the source values at the three vertices of the side must be performed. The equations to perform this interpolation are now outlined. The three vertices of the side on the time step boundary are denoted by the variables *c*, *j*, and *k* and the interpolation point is not denoted by an index. Three equations are defined to give the source at specific *x* and *y* positions on the time boundary as given in Eq. (B.38).

$$\begin{aligned} q_k &= q_j + \left(x_k - x_j\right)\frac{dq}{dx} + \left(y_k - y_j\right)\frac{dq}{dy} \\ q_c &= q_j + \left(x_c - x_j\right)\frac{dq}{dx} + \left(y_c - y_j\right)\frac{dq}{dy} \\ q &= q_j + \left(x - x_j\right)\frac{dq}{dx} + \left(y - y_j\right)\frac{dq}{dy} \end{aligned} \tag{B.38}$$

Using these three equations, $\frac{dq}{dx}$ and $\frac{dq}{dy}$ can be found as given in Eq. (B.39).

$$\begin{aligned} \frac{dq}{dy} &= \begin{cases} \dfrac{q_k - q_j}{\left(y_k - y_j\right)}, & x_k = x_j, \\ \dfrac{\left(x_k - x_j\right)\left(q_c - q_j\right) - \left(x_c - x_j\right)\left(q_k - q_j\right)}{\left(y_c - y_j\right)\left(x_k - x_j\right) - \left(y_k - y_j\right)\left(x_c - x_j\right)}, & x_k \neq x_j \end{cases} \\ \frac{dq}{dx} &= \begin{cases} \dfrac{q_c - q_j}{\left(x_c - x_j\right)} - \dfrac{\left(y_c - y_j\right)\left(q_k - q_j\right)}{\left(y_k - y_j\right)\left(x_c - x_j\right)}, & x_k = x_j \\ \dfrac{\left(q_k - q_j\right)}{\left(x_k - x_j\right)} - \dfrac{\left(y_k - y_j\right)}{\left(x_k - x_j\right)}\left(\dfrac{dq}{dy}\right), & x_k \neq x_j \end{cases} \end{aligned} \tag{B.39}$$

Finally, these expressions can be substituted back into the third expression of Eq. (B.38) to calculate the source at the interpolated point. The same procedure as outlined above

is followed to find the exiting source values for the track that are stored in the variables sourcen_e and sourcen1_e depending on where the track exited the side.

Next, the track attenuation factor is calculated and stored in the tau variable and the contributions to the Tlc and Mlc matrices are found. As with previous methods, a separate loop for small tau is needed to use an approximation for the exponential function as given in Eq. (B.11). The equations used to contribute to the Tlc and Mlc matrices are now outlined. Remember these matrices arose from performing a least-squares fit as given in Eq. (3.25) and is implemented using similar equations to Eqs. (5.3)-(5.9) where $i = 1,\ldots,2N$. Each basis function for the PPWL-STLC method can be written as given in Eq. (B.40), where *b* and *e* refer to values at the beginning and exiting positions of the track on a side.

$$b^i(s) = b^i_{xy} b^i_t = \left[b^i_{xy}(s_b) + \frac{s}{\Delta s}\left(b^i_{xy}(s_e) - b^i_{xy}(s_b) \right) \right] \left[b^i_t(s_b) + \frac{s}{\Delta s}\left(b^i_t(s_e) - b^i_t(s_b) \right) \right] \quad \text{(B.40)}$$

Putting this basis expression into Eq. (5.7), the following integral needs to be performed over the track length on the side.

$$\int_0^{\Delta s_k} ds \left\{ \begin{array}{l} \left[b^i_{xy}(s_b) + \frac{s}{\Delta s_k}\left(b^i_{xy}(s_e) - b^i_{xy}(s_b) \right) \right] \left[b^i_t(s_b) + \frac{s}{\Delta s_k}\left(b^i_t(s_e) - b^i_t(s_b) \right) \right] * \\ \left[b^h_{xy}(s_b) + \frac{s}{\Delta s}\left(b^h_{xy}(s_e) - b^h_{xy}(s_b) \right) \right] \left[b^h_t(s_b) + \frac{s}{\Delta s}\left(b^h_t(s_e) - b^h_t(s_b) \right) \right] \end{array} \right\} \quad \text{(B.41)}$$

Performing this integral, leads to the following expression to be put into the Mlc matrix, where the notation for the values of the basis at the beginning and ending points of the track has been shortened.

$$
\left[M_{LC}\right]_{ih,k} = \sum_{sides \in k} \Delta u_{k,m} \Delta \omega_{k,m} \Delta s_k \left[\begin{array}{l}
b^i_{xyb} b^i_{tb} b^h_{xyb} b^h_{tb} + \\
\frac{1}{2} b^i_{xyb} b^i_{tb} \left(\begin{array}{l} b^h_{xyb}\left(b^h_{te} - b^h_{tb}\right) + \\ b^h_{tb}\left(b^h_{xye} - b^h_{xyb}\right) \end{array} \right) + \\
\frac{1}{2} b^h_{xyb} b^h_{tb} \left(\begin{array}{l} b^i_{xyb}\left(b^i_{te} - b^i_{tb}\right) + \\ b^i_{tb}\left(b^i_{xye} - b^i_{xyb}\right) \end{array} \right) + \\
\frac{1}{3} b^i_{xyb} b^i_{tb} \left(b^h_{te} - b^h_{tb}\right)\left(b^h_{xye} - b^h_{xyb}\right) + \\
\frac{1}{3} b^h_{xyb} b^h_{tb} \left(b^i_{te} - b^i_{tb}\right)\left(b^i_{xye} - b^i_{xyb}\right) + \\
\frac{1}{3} \left(\begin{array}{l} b^i_{xyb}\left(b^i_{te} - b^i_{tb}\right) + \\ b^i_{tb}\left(b^i_{xye} - b^i_{xyb}\right) \end{array} \right) \left(\begin{array}{l} b^h_{xyb}\left(b^h_{te} - b^h_{tb}\right) + \\ b^h_{tb}\left(b^h_{xye} - b^h_{xyb}\right) \end{array} \right) + \\
\frac{1}{4}\left(b^h_{xye} - b^h_{xyb}\right)\left(b^h_{te} - b^h_{tb}\right) \Rightarrow \\
\left(\begin{array}{l} b^i_{xyb}\left(b^i_{te} - b^i_{tb}\right) + \\ b^i_{tb}\left(b^i_{xye} - b^i_{xyb}\right) \end{array} \right) + \\
\frac{1}{4}\left(b^i_{xye} - b^i_{xyb}\right)\left(b^i_{te} - b^i_{tb}\right) \Rightarrow \\
\left(\begin{array}{l} b^h_{xyb}\left(b^h_{te} - b^h_{tb}\right) + \\ b^h_{tb}\left(b^h_{xye} - b^h_{xyb}\right) \end{array} \right) + \\
\frac{1}{5}\left(b^i_{xye} - b^i_{xyb}\right)\left(b^i_{te} - b^i_{tb}\right) \Rightarrow \\
\left(b^h_{xye} - b^h_{xyb}\right)\left(b^h_{te} - b^h_{tb}\right)
\end{array}\right] \qquad \text{(B.42)}
$$

The values to contribute to the conservative flux values in the Tlc matrix and the exiting angular flux are outlined next. Remember the expression for the angular flux along a track is given in Eq. (B.43), where the total source expression is given in Eq. (B.36).

$$\psi_k(s) = \psi_k^{inc} e^{-\sigma_t s} + \int_0^s ds' q_{tot}(s') e^{-\sigma_t (s-s')} \tag{B.43}$$

Using this expression, the exiting angular flux for a track on a side is easily found via Eq. (B.44).

$$\psi_{k,exit}(s = \Delta s_k) = \psi_k^{inc} e^{-\sigma_t \Delta s_k} + \int_0^s ds' q_{tot}(s') e^{-\sigma_t (s-s')} \tag{B.44}$$

The integral over the total source using Eq. (B.36) can be performed to yield the following expression for the total source with attenuation as a function of *s*.

$$\begin{aligned} Q(s) \equiv \int_0^s ds' q_{tot}(s') e^{-\sigma_t (s-s')} &= \left[\frac{t^{n+1} - t_b}{\Delta t^n} \Delta q_b + q_b^{n+1} \right] \frac{1 - e^{-\sigma_t s}}{\sigma_t} \\ &+ \left[\begin{array}{l} \frac{t^{n+1} - t_b}{\Delta t^n} (\Delta q_e - \Delta q_b) + \\ \left(\frac{t_b - t_e}{\Delta t^n} \right) \Delta q_b + \left(q_e^{n+1} - q_b^{n+1} \right) \end{array} \right] \left(\frac{1}{\tau \sigma_t} \left(\sigma_t s - 1 + e^{-\sigma_t s} \right) \right) \\ &+ \left(\frac{t_b - t_e}{\Delta t^n} \right) (\Delta q_e - \Delta q_b) \left(\frac{1}{\tau^2 \sigma_t} \left(2 - 2\sigma_t s + \sigma_t^2 s^2 - 2e^{-\sigma_t s} \right) \right) \end{aligned} \tag{B.45}$$

Finally, the basis weighted integral of Eq. (5.8) needed to contribute to the Tlc matrix is performed. Using the expression for the angular flux, the contribution to the Tlc matrix values can be written as given in Eq. (B.46).

$$\left[T_{LC} \right]_i = \sum_{sides \in k} \Delta u_{k,m} \Delta \omega_{k,m} \int_0^{\Delta s_k} ds \left[\begin{array}{l} \left(b_{xyb}^i + \frac{s}{\Delta s_k} \left(b_{xye}^i - b_{xyb}^i \right) \right) * \\ \left(b_{tb}^i + \frac{s}{\Delta s_k} \left(b_{te}^i - b_{tb}^i \right) \right) * \\ \left(\psi_k^{inc} e^{-\sigma_t s} + Q(s) \right) \end{array} \right] \tag{B.46}$$

Performing this integral over the track on a side leads to the following expression involving the incident flux.

$$\begin{aligned}
&\int_0^{\Delta s_k} ds b^i \psi_k^{inc} e^{-\sigma_t s} \equiv \psi_k^{inc} f_k^{i,inc} \equiv \psi_k^{inc} b_{xyb}^i b_{tb}^i \left(\frac{1-e^{-\tau}}{\sigma_t} \right) \\
&+\psi_k^{inc} \left(b_{xyb}^i \left(b_{te}^i - b_{tb}^i \right) + b_{tb}^i \left(b_{xye}^i - b_{xyb}^i \right) \right) \left(\frac{1-e^{-\tau}\left(\tau+1\right)}{\tau\sigma_t} \right) \\
&+\psi_k^{inc} \left(b_{xye}^i - b_{xyb}^i \right) \left(b_{te}^i - b_{tb}^i \right) \left(\frac{1}{\tau^2\sigma_t} \left(2 - e^{-\tau} \left(2 + \tau^2 + 2\tau \right) \right) \right)
\end{aligned} \tag{B.47}$$

The integration over the basis weighted source term over the track will be split into three terms as given in Eqs. (B.48)-(B.50).

$$\begin{aligned}
&\int_0^{\Delta s_k} ds \left(b_{xyb}^i b_{tb}^i \right) Q(s) = \left(b_{xyb}^i b_{tb}^i \right) \left[\frac{t^{n+1} - t_b}{\Delta t^n} \Delta q_b + q_b^{n+1} \right] \left(\frac{\tau - \left(1 - e^{-\tau}\right)}{\sigma_t^2} \right) \\
&+\left(b_{xyb}^i b_{tb}^i \right) \left[\begin{array}{l} \frac{t^{n+1} - t_b}{\Delta t^n} \left(\Delta q_e - \Delta q_b \right) + \\ \left(\frac{t_b - t_e}{\Delta t^n} \right) \Delta q_b + \left(q_e^{n+1} - q_b^{n+1} \right) \end{array} \right] \left(\frac{1}{\tau\sigma_t^2} \left(1 + \frac{\tau^2}{2} - \tau - e^{-\tau} \right) \right) \\
&+\left(b_{xyb}^i b_{tb}^i \right) \left(\frac{t_b - t_e}{\Delta t^n} \right) \left(\Delta q_e - \Delta q_b \right) \left(\frac{1}{\tau^2\sigma_t^2} \left(2\tau + \frac{\tau^3}{3} - \tau^2 + 2e^{-\tau} - 2 \right) \right)
\end{aligned} \tag{B.48}$$

$$\int_0^{\Delta s_k} ds\left(b_{xyb}^i\left(b_{te}^i-b_{tb}^i\right)+b_{tb}^i\left(b_{xye}^i-b_{xyb}^i\right)\right)\left[\frac{s}{\Delta s_k}Q(s)\right]=\begin{pmatrix} b_{xyb}^i\left(b_{te}^i-b_{tb}^i\right)+ \\ b_{tb}^i\left(b_{xye}^i-b_{xyb}^i\right)\end{pmatrix}*$$

$$\left[\begin{array}{l}\left[\frac{t^{n+1}-t_b}{\Delta t^n}\Delta q_b+q_b^{n+1}\right]\frac{1}{\tau\sigma_t^2}\left(e^{-\tau}\left(1+\tau\right)-1+\frac{\tau^2}{2}\right) \\ +\left[\begin{array}{l}\frac{t^{n+1}-t_b}{\Delta t^n}\left(\Delta q_e-\Delta q_b\right)+ \\ \left(\frac{t_b-t_e}{\Delta t^n}\right)\Delta q_b+\left(q_e^{n+1}-q_b^{n+1}\right)\end{array}\right]\left(\frac{1}{\tau^2\sigma_t^2}\left(1+\frac{\tau^3}{3}-\frac{\tau^2}{2}-e^{-\tau}\left(1+\tau\right)\right)\right) \\ +\left(\frac{t_b-t_e}{\Delta t^n}\right)\left(\Delta q_e-\Delta q_b\right)\left(\frac{1}{\tau^3\sigma_t^2}\left(e^{-\tau}\left(1+\tau\right)-1+\frac{\tau^2}{2}-\frac{\tau^3}{3}+\frac{\tau^4}{8}\right)\right)\end{array}\right] \tag{B.49}$$

$$\int_0^{\Delta s_k} ds\left(b_{xye}^i-b_{xyb}^i\right)\left(b_{te}^i-b_{tb}^i\right)\left[\left(\frac{s}{\Delta s_k}\right)^2Q(s)\right]=\left(b_{xye}^i-b_{xyb}^i\right)\left(b_{te}^i-b_{tb}^i\right)*$$

$$\left[\begin{array}{l}\left[\frac{t^{n+1}-t_b}{\Delta t^n}\Delta q_b+q_b^{n+1}\right]\left(\frac{1}{\tau^2\sigma_t^2}\left(e^{-\tau}\left(\tau^2+2\tau+2\right)+\frac{\tau^3}{3}-2\right)\right) \\ +\left[\begin{array}{l}\frac{t^{n+1}-t_b}{\Delta t^n}\left(\Delta q_e-\Delta q_b\right)+ \\ \left(\frac{t_b-t_e}{\Delta t^n}\right)\Delta q_b+\left(q_e^{n+1}-q_b^{n+1}\right)\end{array}\right]\left(\frac{1}{\tau^3\sigma_t^2}\left(\begin{array}{l}2-\frac{\tau^3}{3}+\frac{\tau^4}{4}- \\ e^{-\tau}\left(\tau^2+2\tau+2\right)\end{array}\right)\right) \\ +\left(\frac{t_b-t_e}{\Delta t^n}\right)\left(\Delta q_e-\Delta q_b\right)\left(\frac{1}{\tau^4\sigma_t^2}\left(\begin{array}{l}e^{-\tau}\left(4+4\tau+2\tau^2\right)-4+ \\ \frac{2\tau^3}{3}-\frac{\tau^4}{2}+\frac{\tau^5}{5}\end{array}\right)\right)\end{array}\right] \tag{B.50}$$

Putting together the expressions in Eqs. (B.47)-(B.50) leads to the value added to the Tlc matrix position for a given basis.

Returning to the PPWL-STLC compute_cell_fluxes( ) routine, a loop over the 2*N* elements on the space-time cell is entered to make contributions to the Tlc matrix for this track on a side using the equations given above. To find the values of the i^{th} basis

function needed for the equations above for the beginning and ending of the track, the find_basis_values( ) method is called. Remember these basis functions are PWL spatially with a value of beta at the center of the cell and linear in time with a value of unity or zero at the beginning or end of the time step. Therefore, the spatial part of the basis function is found by the same interpolation procedure used to find the value of the source at a given point and the time part of the basis function is found through a simple linear interpolation between values at the incoming and exiting time values. Next, a second loop over elements is entered to contribute to the Mlc matrix for this track. Again the find_basis_values( ) method is called to find the values of the j^{th} basis at the beginning and exiting positions of the track on the side. Finally, the exiting angular flux is calculated. After calculating these values for this side, the ID of the next side the track enters is found or the exiting track flags are set according to if the track exits out of the time boundary or a spatial edge. The exiting position of the track is set to the starting position for the next side and the while loop over sides continues until one of the flags is set to true. To end the loop over tracks incoming on a spatial edge, the track ID and exiting angular flux are put in the appropriate vectors depending on where the track exits. If the track exits the cell on the time boundary the track ID and angular flux are put into the time_tracks vector as a pair. If the track exits on a spatial edge of the cell, the track ID is put into the mytracks vector for the exiting face and the angular flux is put into the myfluxes vector for the exiting face.

Once all of the incoming tracks on all incoming spatial faces of the cell have been looped over, the tracks incoming on the time boundary for the given angle and

energy group are looped over. The number of tracks incoming on this boundary is retrieved from the psi_time_inc vector and the spatial coordinates of the vertices of the cell are converted to *i* and *j* indices using Eqs. (B.3)-(B.5). The coordinates of the cell center at the beginning of the time step are also converted to index values and stored in the index_z vector. Then a loop over the tracks incoming on the time boundary is entered. The track ID is gotten from the psi_time_inc vector and converted to an *i* and *j* index using Eq. (5.20). Some variables are then defined to contribute to finding the incoming track position on the time boundary as defined in Eq. (B.35). The incoming position in time is known and set to be the value at the incoming time step as stored in the *vt0* variable.

Next, the incoming spatial position of the track on the time boundary is calculated using a modification of Eqs. (B.14)-(B.17) where the *z* variable is replaced by the *vt* variable and then using Eq. (B.35) and Eq. (5.22) to find the incoming spatial position. With this incoming position, the incoming side on this time boundary face is determined. This is done by looping over the edges of the cell and checking if the incoming position is on the current side by calling the check_face_side( ) method. The check_face_side( ) method has been outlined in the previous compute cell fluxes routines in 2D and 3D for the LC method. It is a simple routine that uses the cross products of the edges of the given side to determine if a given point lies within the side.

If the track is on the current side, the check_time_intersect( ) method is called to move the track toward the side-face center if near an intersection with a vertex or edge of the side. If the track is moved, the updated track indices and other variables are

calculated. Finally, the incoming position is checked again to make sure it still occurs on the current side and the side ID is set in the nextside variable.

After determining the incoming side for the track on the time boundary, a while loop is entered to follow the track through the sides it traverses on the cell. This loop is almost the same as the while loop followed in the spatial edge loop previously outlined. However, when determining the exiting position of the track on the first side it enters the cell, the three spatial faces of the side must be checked instead of only two that are checked for subsequent sides (due to the track incoming on the time boundary face). This is denoted by a separate loop for if the while loop is in iteration zero. After determining the shortest track length when considering the spatial edges of the side, the track length to exit out of the time boundary is calculated and compared to the value stored in the dels variable. If the track length to the exiting time boundary is shorter than the current track length, the same procedure as outlined previously is followed for determining the exiting position on this boundary and moving the track if necessary.

Next, the source at the beginning and exiting points of the track on the side are calculated in the same manner as outlined previously. The contributions to the Tlc and Mlc matrices as well as the calculation of the exiting angular flux also follow the same procedure as outlined previously. To end the traversal of this track on this side, the ID of the next side the track enters or an exit flag is set depending on where the track exited the current side. Finally, the exiting position of the track is set to the starting position for the next side and the while loop continues in the cell. Once, the track has exited the cell

the track ID and exiting angular flux are put into the appropriate vectors depending on where the track exited the cell as was outlined previously.

Finally, after finishing all tracks entering the cell on the time boundary for the given angle, all of the tracks entering the cell have been traversed across the cell and values need to be put into the local vectors. This is done by looping over the spatial edges of the cell. If a spatial edge is an exiting face, the values stored in the local mytracks and myfluxes vectors are put into the my_global_tracks and my_global_fluxes vectors for the exiting face to be put into the psi_face vector in the STLC_2D sweepchunk. This concludes the calculation of the compute_cell_fluxes( ) method for the PPWL-STLC method as called from the STLC_2D sweepchunk.

Finding the Least-Squares Scalar Flux

After all of the anglesets have been swept for each cellset in the problem of the current sweep for a groupset, the routine to find the least-squares scalar flux is called. This call is found in the transport_step located in SweepProblem.h. The LC methods call a routine named LC_flux( ) and the STLC methods call a routine named STLC_flux( ) for the current groupset. These routines are also located in SweepProblem.h. Both of these methods loop over each cell in the problem followed by the groups in the groupset. For each element of the cell, the right hand side of the system to be solved is gotten from the Tlc matrix for the current group. Next, the least-squares flux is found by using Gauss Elimination with no pivoting on the system which uses the Mlc matrix for the current group as outlined in the algorithm section of the implementation section 5. This routine stores the solution in the right hand side of the

system, referred to as the b vector, when it is completed. A loop over elements is then performed to set the scalar flux (phi) on each element to its value stored in the b vector for this group and to zero out the appropriate elements in the Tlc and Mlc matrices for the next sweep. The difference between the LC and STLC routines is that the scalar flux at the beginning and end of the time step is solved and set with the STLC methods. These values are stored in the phi_n_plus and phi_n_minus variables on each spatial element of the space-time cell for the STLC methods. The average scalar flux variable is found as the average of these phi_n_plus and phi_n_minus values for each element and stored in the phi variable. Note again, for the LC and STLC methods that use a piece-wise constant approximation for the source, this matrix solve is not necessary because each cell only has one element. However, it is done for convenience in using the same solving routine for all methods. Also, note that this solving routine for the least-squares scalar flux is only appropriate for isotropic scattering. This routine needs to be modified to accommodate anisotropic scattering, which would also bring in the need for modifications elsewhere with these long characteristic methods.

STLC in Td_solve()- the time-dependent solver

As mentioned previously, the sweeping of an STLC problem or time-dependent LC problem is first handled through the td_solve() routine in the NeutronicsSolver, RadSolver, or RadSolverSTLC. There are several special calls in this method when solving an STLC problem. First, the differences for a time-dependent neutronics problem are reviewed. After a transport solve has been performed for the current time step, the angular flux values stored in the oldpsi vector in each cell are updated for the

STLC method. This is done by calling a routine for each cell called set_oldpsiLC() and sending it the current angular flux values found exiting the time boundary stored in the psiLC vector. This routine takes care of any resizing of this oldpsiLC vector that is needed. Also, as mentioned previously, the tracks incoming on the boundary edges of the next time step for this STLC method are found and set by calling the Set_STLC_new_ts_Boundary() routine after the time step size for the next time step has been determined.

There are also extra variables that need to be set when calculating the inscattering and total source before each sweep. For the STLC methods, each element of the cell includes source values at the beginning and end of the time step denoted by n_minus and n_plus respectively. Considering a neutronics problem, the fixed source at n_plus and n_minus is set through a call to set_fixed_source( ) located in NeutronicsSolver.h in ss_solve( ). The extra inscattering variables for each group for STLC are called qinscat_n_plus and qinscat_n_minus. These values are set in the file called Setinscat_Method.h similar to the average qinscat value but they use the values of the scalar flux at the beginning (phi_n_minus) and end (phi_n_plus) of the time step. Finally, the total source at n_plus and n_minus is set in SetTotsource_Method.h by adding the contributions from qfixed_*, qinscat_*, and a locally calculated qwithscat_* to account for within group-set scattering, where * is either n_plus or n_minus.

Another thing to point out when solving an STLC neutronics problem is that the STLC methods require using a separate iterative operator for the within_groupset and across_groupset solvers. This was outlined in the implementation section of section 5

and accounts for needing to using the flux values at n_plus and n_minus for each element.

PWC-STLC with RadSolver.h

Next, the implementation differences for a time-dependent radiative transfer problem using the STLC method are discussed. When running a PWC-STLC radiative transfer problem, the usual RadSolver is used because this long characteristic discretization in time looks like a fully implicit time discretization. Therefore, the same equations used to solve for temperature at each element for the other methods in PDT can be used for the PWC-STLC method. There are some slight modifications to this RadSolver to account for the PWC-STLC method. First, in the initialize( ) routine, the angular flux values incoming on the time boundary must be initialized to the Planckian at the initial temperature. This is done for the PWC-STLC method by setting each track's angular flux in psiLC and oldpsiLC on the cell to the normalized Planckian.

The next difference with the usual solver occurs in the radTsolver( ) routine. This routine calls the calculate_q_fixed_rad( ) method. As in the neutronics solver to calculate the fixed source, the fixed source at n_plus and n_minus must be set for this STLC method. Currently these values are set to the same as the average fixed source in the cell. The same source values for q_tot and q_inscat at n_plus and n_minus as outlined in the NeutronicsSolver.h must also be set for the RadSolver.h with the PWC-STLC method. These values are set in Setinscat_Method.h and SetTotsource_Method.h in the appropriate method for a Radiative Transfer problem. Note for the PWC-STLC method the qinscat_n_plus and qinscat_n_minus values are set to the same as the

average qinscat value. And again, the qtot_n_plus and qtot_n_minus values are just a summation of the previously defined source values at n_plus and n_minus.

The rad_T_solver( ) routine also calls the ss_solve after setting source values and material values. In this routine, the material source and radiation source in each cell is added to the qfixed_n_plus and qfixed_n_minus values for the PWC-STLC method, as well as the average qfixed value.

Back to td_solve( ), after a transport solve has been performed for the current time step, the angular flux values stored in the oldpsi vector in each cell are updated for the PWC-STLC method. This is done by calling a routine for each cell called set_oldpsiLC() and sending it the current values found exiting the time boundary stored in the psiLC vector. This routine takes care of any resizing of this oldpsiLC vector that is needed. Also, as mentioned previously, the tracks incoming on the boundary edges of the next time step for this PWC-STLC method are found and set by calling the Set_STLC_new_ts_Boundary() routine after the time step size for the next time step has been determined.

Also, as with a neutronics problem, a radiative transfer problem that uses any STLC method also uses the separate STLC iterative operator defined in IterativeSolveOperations.h to find the ARD for each element of a cell.

PPWL-STLC with RadSolverSTLC.h

Since the PPWL-STLC method treats temperature as linear and discontinuous and solves for element values at the beginning and end of the time step, it requires different equations to solve a radiative transfer problem than the usual finite differenced

radiative transfer equations. Therefore, a separate radiation solver is implemented in PDT for the PPWL-STLC method called RadSolverSTLC.h. The routines and flow of this solver follow the same as the usual RadSolver.h. The main differences come from implementing the solving of the PPWL-STLC radiative transfer equations.

Review of PPWL-STLC Radiative Transfer Equations

Each spatial element of a cell needs to solve for two values: the temperature at the beginning (n_minus) and end (n_plus) of the time step. Since these temperatures are coupled, this solving procedure involves solving a set of two equations (one for the beginning and one for the end of the time step) for each element. For a given element, the following two equations define the two temperature equations that need to be solved. A value at the beginning of the time step is referred to as *bot* and a value at the end of the time step is referred to as *top*.

$$\frac{C_v}{\Delta t}\left(T^{top}-T^{bot}\right)=\frac{2}{3}\left\{\sum_{g=1}^{G}\sigma_g\phi_g^{top}-4\pi\sum_{g=1}^{G}\sigma_g B_g\left(T^{top}\right)+Q_m^{top}\right\}+ \frac{1}{3}\left\{\sum_{g=1}^{G}\sigma_g\phi_g^{bot}-4\pi\sum_{g=1}^{G}\sigma_g B_g\left(T^{bot}\right)+Q_m^{bot}\right\} \tag{B.51}$$

$$\frac{C_v}{\Delta t}\left(T^{top}-T^{bot}\right)+2\frac{C_v}{\Delta t}\left(T^{bot}-T^{n-}\right)= \frac{2}{3}\left\{\sum_{g=1}^{G}\sigma_g\phi_g^{bot}-4\pi\sum_{g=1}^{G}\sigma_g B_g\left(T^{bot}\right)+Q_m^{bot}\right\}+ \frac{1}{3}\left\{\sum_{g=1}^{G}\sigma_g\phi_g^{top}-4\pi\sum_{g=1}^{G}\sigma_g B_g\left(T^{top}\right)+Q_m^{top}\right\} \tag{B.52}$$

Next, the Planckian is linearized around some temperature as given in Eq. (B.53).

$$B_g\left(T^{top}\right) \approx B_g\left(T^{top*}\right) + \left(T^{top} - T^{top*}\right)\left.\frac{\partial B}{\partial T}\right|_{T^{top*}}$$
$$B_g\left(T^{bot}\right) \approx B_g\left(T^{bot*}\right) + \left(T^{bot} - T^{bot*}\right)\left.\frac{\partial B}{\partial T}\right|_{T^{bot*}} \tag{B.53}$$

These approximations can then be substituted into Eqs. (B.51) and (B.52) leading to the following expressions.

$$\begin{aligned}&\frac{C_v}{\Delta t}\left(T^{top} - T^{bot}\right) = \\ &\frac{2}{3}\left\{\sum_{g=1}^{G}\sigma_g\phi_g^{top} - 4\pi\sum_{g=1}^{G}\sigma_g\left[B_g\left(T^{top*}\right) + \left(T^{top} - T^{top*}\right)\left.\frac{\partial B}{\partial T}\right|_{T^{top*}}\right] + Q_m^{top}\right\} + \\ &\frac{1}{3}\left\{\sum_{g=1}^{G}\sigma_g\phi_g^{bot} - 4\pi\sum_{g=1}^{G}\sigma_g\left[B_g\left(T^{bot*}\right) + \left(T^{bot} - T^{bot*}\right)\left.\frac{\partial B}{\partial T}\right|_{T^{bot*}}\right] + Q_m^{bot}\right\}\end{aligned} \tag{B.54}$$

$$\begin{aligned}&\frac{C_v}{\Delta t}\left(T^{top} - T^{bot}\right) + 2\frac{C_v}{\Delta t}\left(T^{bot} - T^{n-}\right) = \\ &\frac{2}{3}\left\{\begin{array}{l}\displaystyle\sum_{g=1}^{G}\sigma_g\phi_g^{bot} - 4\pi\sum_{g=1}^{G}\sigma_g\left[B_g\left(T^{bot*}\right) + \left(T^{bot} - T^{bot*}\right)\left.\frac{\partial B}{\partial T}\right|_{T^{bot*}}\right] + \\ Q_m^{bot}\end{array}\right\} + \\ &\frac{1}{3}\left\{\sum_{g=1}^{G}\sigma_g\phi_g^{top} - 4\pi\sum_{g=1}^{G}\sigma_g\left[B_g\left(T^{top*}\right) + \left(T^{top} - T^{top*}\right)\left.\frac{\partial B}{\partial T}\right|_{T^{top*}}\right] + Q_m^{top}\right\}\end{aligned} \tag{B.55}$$

The difference between top and bottom temperatures can be written differently as given in Eq. (B.56).

$$T^{top} - T^{bot} = \left(T^{top} - T^{top*}\right) + T^{top*} - \left(T^{bot} - T^{bot*}\right) - T^{bot*} \tag{B.56}$$

Substituting this expression into Eq. (B.54) and Eq. (B.55) leads to the following two equations.

$$
\begin{aligned}
&\left(T^{top}-T^{top*}\right)\left[\frac{C_v}{\Delta t}+\frac{2}{3}4\pi\sum_{g=1}^{G}\sigma_g\left.\frac{\partial B}{\partial T}\right|_{T^{top*}}\right]+\\
&\left(T^{bot}-T^{bot*}\right)\left[\frac{1}{3}4\pi\sum_{g=1}^{G}\sigma_g\left.\frac{\partial B}{\partial T}\right|_{T^{bot*}}-\frac{C_v}{\Delta t}\right]=\frac{C_v}{\Delta t}\left(T^{bot*}-T^{top*}\right)+\\
&\frac{2}{3}\left\{\sum_{g=1}^{G}\sigma_g\phi_g^{top}-4\pi\sum_{g=1}^{G}\sigma_g B_g\left(T^{top*}\right)+Q_m^{top}\right\}+\\
&\frac{1}{3}\left\{\sum_{g=1}^{G}\sigma_g\phi_g^{bot}-4\pi\sum_{g=1}^{G}\sigma_g B_g\left(T^{bot*}\right)+Q_m^{bot}\right\}
\end{aligned}
\tag{B.57}
$$

$$
\begin{aligned}
&\left(T^{top}-T^{top*}\right)\left[\frac{C_v}{\Delta t}+\frac{1}{3}4\pi\sum_{g=1}^{G}\sigma_g\left.\frac{\partial B}{\partial T}\right|_{T^{top*}}\right]+\\
&\left(T^{bot}-T^{bot*}\right)\left[\frac{C_v}{\Delta t}+\frac{2}{3}4\pi\sum_{g=1}^{G}\sigma_g\left.\frac{\partial B}{\partial T}\right|_{T^{bot*}}\right]=\\
&\frac{C_v}{\Delta t}\left(T^{bot*}-T^{top*}\right)-2\frac{C_v}{\Delta t}\left(T^{bot*}-T^{n-}\right)+\\
&\frac{2}{3}\left\{\sum_{g=1}^{G}\sigma_g\phi_g^{bot}-4\pi\sum_{g=1}^{G}\sigma_g B_g\left(T^{bot*}\right)+Q_m^{bot}\right\}+\\
&\frac{1}{3}\left\{\sum_{g=1}^{G}\sigma_g\phi_g^{top}-4\pi\sum_{g=1}^{G}\sigma_g B_g\left(T^{top*}\right)+Q_m^{top}\right\}
\end{aligned}
\tag{B.58}
$$

The previous two equations can be written in matrix form as given in Eq. (B.59).

$$
\begin{bmatrix}
\left(\frac{C_v}{\Delta t}+\frac{2}{3}4\pi\sum_{g=1}^{G}\sigma_g\left.\frac{\partial B}{\partial T}\right|_{T^{top*}}\right) & \left(\frac{1}{3}4\pi\sum_{g=1}^{G}\sigma_g\left.\frac{\partial B}{\partial T}\right|_{T^{bot*}}-\frac{C_v}{\Delta t}\right)\\
\left(\frac{C_v}{\Delta t}+\frac{1}{3}4\pi\sum_{g=1}^{G}\sigma_g\left.\frac{\partial B}{\partial T}\right|_{T^{top*}}\right) & \left(\frac{C_v}{\Delta t}+\frac{2}{3}4\pi\sum_{g=1}^{G}\sigma_g\left.\frac{\partial B}{\partial T}\right|_{T^{bot*}}\right)
\end{bmatrix}
\begin{bmatrix}
\left(T^{top}-T^{top*}\right)\\
\left(T^{bot}-T^{bot*}\right)
\end{bmatrix}=
$$

$$
\begin{bmatrix}
\frac{2}{3}\sum_{g=1}^{G}\sigma_g\phi_g^{top}+\frac{1}{3}\sum_{g=1}^{G}\sigma_g\phi_g^{bot}\\
\frac{2}{3}\sum_{g=1}^{G}\sigma_g\phi_g^{bot}+\frac{1}{3}\sum_{g=1}^{G}\sigma_g\phi_g^{top}
\end{bmatrix}+
$$

$$
\begin{bmatrix}
-\frac{2}{3}4\pi\sum_{g=1}^{G}\sigma_g B_g\left(T^{top*}\right)-\frac{1}{3}4\pi\sum_{g=1}^{G}\sigma_g B_g\left(T^{bot*}\right)\\
-\frac{2}{3}4\pi\sum_{g=1}^{G}\sigma_g B_g\left(T^{bot*}\right)-\frac{1}{3}4\pi\sum_{g=1}^{G}\sigma_g B_g\left(T^{top*}\right)
\end{bmatrix}+
$$

$$
\begin{bmatrix}
\frac{2}{3}Q_m^{top}+\frac{1}{3}Q_m^{bot}+\frac{C_v}{\Delta t}\left(T^{bot*}-T^{top*}\right)\\
\frac{2}{3}Q_m^{bot}+\frac{1}{3}Q_m^{top}+\frac{C_v}{\Delta t}\left(T^{bot*}-T^{top*}\right)-2\frac{C_v}{\Delta t}\left(T^{bot*}-T^{n-}\right)
\end{bmatrix}
\tag{B.59}
$$

The following matrix can be defined as given in Eq. (B.60).

$$
[C]\equiv\begin{bmatrix}
\left(\frac{C_v}{\Delta t}+\frac{2}{3}4\pi\sum_{g=1}^{G}\sigma_g\left.\frac{\partial B}{\partial T}\right|_{T^{top*}}\right) & \left(\frac{1}{3}4\pi\sum_{g=1}^{G}\sigma_g\left.\frac{\partial B}{\partial T}\right|_{T^{bot*}}-\frac{C_v}{\Delta t}\right)\\
\left(\frac{C_v}{\Delta t}+\frac{1}{3}4\pi\sum_{g=1}^{G}\sigma_g\left.\frac{\partial B}{\partial T}\right|_{T^{top*}}\right) & \left(\frac{C_v}{\Delta t}+\frac{2}{3}4\pi\sum_{g=1}^{G}\sigma_g\left.\frac{\partial B}{\partial T}\right|_{T^{bot*}}\right)
\end{bmatrix}
\tag{B.60}
$$

This 2x2 system of equations given in Eq. (B.59) can be solved by finding the inverse of the [C] matrix. This inverse is defined in Eq. (B.61) with the determinant of the matrix defined in Eq. (B.62).

$$\left[C\right]^{-1}=\frac{1}{\left\|\left[C\right]\right\|}\begin{bmatrix}\left(\frac{C_v}{\Delta t}+\frac{2}{3}4\pi\sum_{g=1}^{G}\sigma_g\left.\frac{\partial B}{\partial T}\right|_{T^{bot*}}\right) & -\left(\frac{1}{3}4\pi\sum_{g=1}^{G}\sigma_g\left.\frac{\partial B}{\partial T}\right|_{T^{bot*}}-\frac{C_v}{\Delta t}\right)\\ -\left(\frac{C_v}{\Delta t}+\frac{1}{3}4\pi\sum_{g=1}^{G}\sigma_g\left.\frac{\partial B}{\partial T}\right|_{T^{top*}}\right) & \left(\frac{C_v}{\Delta t}+\frac{2}{3}4\pi\sum_{g=1}^{G}\sigma_g\left.\frac{\partial B}{\partial T}\right|_{T^{top*}}\right)\end{bmatrix} \tag{B.61}$$

$$\begin{aligned}\left\|\left[C\right]\right\|=&\left(\frac{C_v}{\Delta t}+\frac{2}{3}4\pi\sum_{g=1}^{G}\sigma_g\left.\frac{\partial B}{\partial T}\right|_{T^{bot*}}\right)\left(\frac{C_v}{\Delta t}+\frac{2}{3}4\pi\sum_{g=1}^{G}\sigma_g\left.\frac{\partial B}{\partial T}\right|_{T^{top*}}\right)-\\&\left(\frac{C_v}{\Delta t}+\frac{1}{3}4\pi\sum_{g=1}^{G}\sigma_g\left.\frac{\partial B}{\partial T}\right|_{T^{top*}}\right)\left(\frac{1}{3}4\pi\sum_{g=1}^{G}\sigma_g\left.\frac{\partial B}{\partial T}\right|_{T^{bot*}}-\frac{C_v}{\Delta t}\right)\end{aligned} \tag{B.62}$$

Finally, solving Eq. (B.59) for the temperature unknowns leads to the following expression.

$$\begin{aligned}\begin{bmatrix}\left(T^{top}-T^{top*}\right)\\ \left(T^{bot}-T^{bot*}\right)\end{bmatrix}=&\left[C\right]^{-1}\begin{bmatrix}\frac{2}{3}\sum_{g=1}^{G}\sigma_g\phi_g^{top}+\frac{1}{3}\sum_{g=1}^{G}\sigma_g\left(\phi_g^{bot}\right)\\ \frac{2}{3}\sum_{g=1}^{G}\sigma_g\phi_g^{bot}+\frac{1}{3}\sum_{g=1}^{G}\sigma_g\phi_g^{top}\end{bmatrix}-\\&\left[C\right]^{-1}\begin{bmatrix}\frac{2}{3}4\pi\sum_{g=1}^{G}\sigma_g B_g\left(T^{top*}\right)+\frac{1}{3}4\pi\sum_{g=1}^{G}\sigma_g B_g\left(T^{bot*}\right)\\ \frac{2}{3}4\pi\sum_{g=1}^{G}\sigma_g B_g\left(T^{bot*}\right)+\frac{1}{3}4\pi\sum_{g=1}^{G}\sigma_g B_g\left(T^{top*}\right)\end{bmatrix}+\\&\left[C\right]^{-1}\begin{bmatrix}\frac{2}{3}Q_m^{top}+\frac{1}{3}Q_m^{bot}+\frac{C_v}{\Delta t}\left(T^{bot*}-T^{top*}\right)\\ \frac{2}{3}Q_m^{bot}+\frac{1}{3}Q_m^{top}+\frac{C_v}{\Delta t}\left(T^{bot*}-T^{top*}\right)-2\frac{C_v}{\Delta t}\left(T^{bot*}-T^{n-}\right)\end{bmatrix}\end{aligned} \tag{B.63}$$

The PPWL-STLC transport equation for the i^{th} element is written in terms of the least-squares procedure as given in Eq. (B.64).

$$\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=1}^{\#\text{ tracks}} A_{k,m} \int_0^{\Delta s_{k,m}} b_i(s) \sum_{j=1}^{N} b_j^{sp}(\vec{r})\vec{b}(t) \begin{bmatrix} \dfrac{\partial \psi_{g,j}}{\partial s} + \sigma_g \psi_{g,j} = \\ \sigma_g B_{g,j} + Q_{r,j} \end{bmatrix} \tag{B.64}$$

The basis function with respect to time can be written in terms of top and bottom values as well as the unknowns in this equation. Also, the Planckian can be linearized as was given in Eq. (B.53). Rewriting this transport equation leads to the following expression.

$$\sum_{g=1}^{G}\sum_{m=1}^{M} w_m J_m \sum_{k=1}^{\#\text{ tracks}} A_{k,m} \int_0^{\Delta s_{k,m}} b_i(s) \sum_{j=1}^{N} b_j^{sp}(\vec{r}) \begin{bmatrix} b^{top} & b^{bot} \end{bmatrix} \Rightarrow$$

$$\left\{ \begin{array}{l} \dfrac{\partial \psi_{g,j}}{\partial s} + \begin{bmatrix} \sigma_g \psi_{m,g}^{top} \\ \sigma_g \psi_{m,g}^{bot} \end{bmatrix}_j = \\ \begin{bmatrix} \sigma_g \left.\dfrac{\partial B}{\partial T}\right|_{T^{top*}} & 0 \\ 0 & \sigma_g \left.\dfrac{\partial B}{\partial T}\right|_{T^{bot*}} \end{bmatrix}_j \begin{bmatrix} \left(T^{top} - T^{top*}\right) \\ \left(T^{bot} - T^{bot*}\right) \end{bmatrix}_j + \\ \begin{bmatrix} \sigma_g B_g\left(T^{top*}\right) \\ \sigma_g B_g\left(T^{bot*}\right) \end{bmatrix}_j + \begin{bmatrix} Q_r^{top} \\ Q_r^{bot} \end{bmatrix}_j \end{array} \right\} \tag{B.65}$$

This expression contains the temperature unknowns that were solved for in Eq. (B.63) and represents the transport equation that is solved in the transport step of the solve_ard( ) method of the RadSolverSTLC method. By substituting the expression for these temperature unknowns into Eq. (B.65), the source needed to solve for the absorption rate density is defined for this PPWL-STLC method.

Next, the implementation of these equations in the RadSolverSTLC.h file in PDT is outlined. Although this file follows the same flow and the same basic routines as the RadSolver, there are many differences that are specific for the PPWL-STLC method.

Beginning with the td_solve( ) routine, the first method that is called is the initialize( ) method. This method initializes all phi and temperature values at n_plus and n_minus for each element. This includes initializing the values of the matrix inverse defined by the C matrix above to zero. These values are stored on the element as four double values called STLC_inv_00, STLC_inv_01, STLC_inv_10, and STLC_inv_11. The angular intensity values incoming on the time boundary are also set to the Planckian evaluated at an initial temperature in the psiLC vector stored on the cell.

Next, the rad_T_solver( ) routine is called from td_solve for the current time step. The first step in this solver is to extrapolate the temperature values on each element for the current time step. Next, inside the point-wise temperature convergence loop, the calculate_q_fixed_rad( ) routine is called. This routine sets the values of the inverse C matrix and puts the terms from Eq. (B.65) using Eq. (B.63) into the variables called qfixed_n_minus and qfixed_n_plus using the current temperature values. The average qfixed value is also found and stored. Next, the steady state solver is called via the ss_solve( ) routine. This routine adds the contribution to the fixed source from the radiation source and then calls the solve_ard( ) method.

In the solve_ard( ) method, first the compute_b( ) routine using the given iterative algorithm is called. After this compute_b is performed, the usual solving routine is followed to find the ARD on each element using the current temperature values. Remember the algorithms called from this solve_ard( ) routine use the ard_stlc_iter_op. After this solve_ard( ) routine has converged, the updated temperatures are calculated using these ARD values in the calculate_Te_n( ) routine.

This routine uses Eq. (B.63) to solve for the updated Te_n_minus and Te_n_plus values for each element. With these new temperature values, the average temperature on each element is checked for point-wise convergence. If convergence hasn't been reached the temperature values are reset and the rad_T_solver( ) continues until the temperature has converged or the iteration limits are reached.

After the rad_T_solver( ) routine is complete, temperature values on each element are updated for the next time step, the angular intensity values entering the next time step are updated for each cell, and the incoming tracks and associated boundary condition are found for the next time step by calling the Set_STLC_new_ts_Boundary( ) routine. This concludes the details of the td_solve( ) routine for the PPWL-STLC method for a radiative transfer problem found in RadSolverSTLC.

Reporting average values in PDT

After solving a problem, for the current time-step or in steady-state, PDT reports values to an output file via reporting functions in PolygonalCell.h and PolyhedralCell.h. For a neutronics problem, the average scalar flux and other moments are reported for the whole cell and on each element in the cell. For a radiative transfer problem, other average values including the emission rate density and absorption rate density are reported as well as the element values including temperatures in the cell. No special output is needed for a LC or STLC problem, but it should be noted that the values reported for the STLC method are at the midpoint of every time step, except for the temperature values reported at the end of the time step (n_plus). The scalar flux could be reported at the beginning and end of each time step for an STLC problem, as an

average in the cell, or for each element of the cell. This would need to be implemented in the reporting function in PolygonalCell.h.