Dissertation

Discrete Ordinates Radiation Transport using High Order Finite Element Spatial Discretizations on Meshes with Curved Surfaces

Douglas N. Woods

11 June 2018

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 Introduction

There are several applications for the thermal radiation transport (TRT) equation for nuclear fusion. Two of which include astrophysics such as stars or supernovae [2], and inertial confinment fusion such as the National Ignition Facility[1] (NIF). Nuclear fusion occurs in the high energy density physics (HEDP) regime where mass and energy densities of a material are very high [3, 2]. Since material temperatures are very high, materials emit black body radiation in tremendous quantities. This thermal radiation field deposits energy back to the material influencing the material internal energy, temperature, and density. The radiation field can exchange enough momentum with the material that it directly affects the fluid motion [2]. Radiation transport and material absorption and emission are interdependent mechanisms that must be modeled simultaneously to obtain an energy distribution throughout the material. The material may also be exposed to additional hydrodynamic forces that change pressures, introduce density fluctuations, and cause fluid motion. In turn, the motion and energy density of the fluid affects the location and quantity of radiation emission. This complicated system of thermal radiation transport with hydrodynamics is studied in radiation hydrodynamics. While all of the forces in these HEDP systems are important, we can study some of the effects independently. For instance, we study hydrodynamics separately from TRT.

The staggered grid hydrodynamics (SGH) approach is considered a traditional approach that uses the finite difference or finite volume methods, and has to compensate for calculating the spatial gradients, which are highly dependent on the mesh resolution [4]. The finite element method (FEM) has been employed, approximating the thermodynamic variables as piecewise constant and the kinematic variables as linear continuous[5], and higher order finite elements [6].

[1] `https://lasers.llnl.gov`

Lawrence Livermore National Laboratory (LLNL) is developing a hydrodynamics code called BLAST.[2] [4]. It solves the Euler equations using a general finite element method (FEM) on meshes with curved surfaces for the conservation of mass, energy, and momentum of a fluid [6]. Novel features include: higher order elements to represent the thermodynamic and kinematic variables, and meshes with curved surfaces. Compared to SGH, Dobrev et al. [6] demonstrated their method can more accurately model flow geometry, symmetry of radial flow, and an increased resolution of a shock front saying their method has the ability to model the shock within a single mesh element. These improvements have reduced some of the numerical errors that have been exhibited by previous methods.

BLAST utilizes MFEM [7], a general finite element library also being developed at LLNL, to spatially discretize using the high order FEM on meshes with curved surfaces. This proposed TRT research could be combined with a hydrodynamics code such as BLAST to develop a radiation hydrodynamics package to better model HEDP problems such as those produced during inertial confinement fusion at the NIF. The integrated radiation hydrodynamics code will need each physics package to numerically solve the same physical problem and communicate between packages. Hence, it is advantageous to utilize the same general finite element library when considering the integration of code packages.

This chapter introduces TRT and establishes our motivation for using the proposed methods. In Section 1.1, we give a brief introduction to the TRT equations and nomenclature. In Section 1.2, we describe various discretization methods used to solve the TRT equations. In Section 1.3, we describe the diffusion limit and it's application to optically thick problems found in HEDP regimes. Finally, in Section 1.4, we outline the remained of this research proposal.

[2] https://computation.llnl.gov/project/blast/

1.1 Radiation Transport

The steady-state, mono-energetic radiation transport equation,

$$\boldsymbol{\Omega}_m \cdot \boldsymbol{\nabla}\psi(\boldsymbol{x}, \boldsymbol{\Omega}_m) + \sigma_t(\boldsymbol{x})\psi(\boldsymbol{x}, \boldsymbol{\Omega}_m) = \frac{1}{4\pi}\sigma_s(\boldsymbol{x}) \int_{4\pi} \psi(\boldsymbol{x}, \boldsymbol{\Omega}')d\Omega' + \frac{1}{4\pi}S_0(\boldsymbol{x}), \qquad (1)$$

describes the angular flux $\psi(\boldsymbol{x}, \boldsymbol{\Omega})$, which is a function of position ($\boldsymbol{x}$) and direction of travel ($\boldsymbol{\Omega}$). We also define the scalar flux by

$$\phi(\boldsymbol{x}) = \int_{4\pi} \psi(\boldsymbol{x}, \boldsymbol{\Omega}')d\Omega', \qquad (2)$$

which accounts for the angular fluxes in all directions at spatial position $\boldsymbol{x}$. Both the angular and scalar fluxes describe a particle density in a phase space denoted by their dependent variables and have units $\text{cm}^{-2}\text{s}^{-1}$. In general, the total macroscopic cross section, $\sigma_t(\boldsymbol{x}) = \varsigma_t(\boldsymbol{x})N(\boldsymbol{x})$, is a property of the microscopic total cross section, $\varsigma_t(\boldsymbol{x})$ cm^2, and the number density of the material, $N(\boldsymbol{x})$ cm^{-3}. The absorption and scattering cross sections are related to the total cross section by $\sigma_t(\boldsymbol{x}) = \sigma_a(\boldsymbol{x}) + \sigma_s(\boldsymbol{x})$. Additionally, the isotropic volumetric source, $S_0(\boldsymbol{x})/4\pi$ is some isotropically emitting external source (e.g., fission neutrons or blackbody radiation).

There are only a few specific cases where the radiation transport equation can be solved exactly. Problems of interest typically fall outside of this subset and discretization methods must be employed to obtain approximate solutions. There are two general categories of methods used to solve this equation: stochastic and deterministic methods.

Stochastic methods take a statistical approach. In any particular source region, a single particle is emitted and tracked through its "random walk" through the problem domain until it is either absorbed or escapes the domain. One can simply tally the number of particles that get absorbed into a region on interest. If enough particles are simulated, a statistically significant conclusion can be drawn about the particle density in a region of interest. This is very accurate but also computationally demanding

because of the large number of particles required to be simulated. This method does not solve any equation, but rather is based on the physics of the particle-material interactions.

Alternatively, deterministic methods solve the radiation transport equation for the scalar flux by discretizing the equation in each of the dependent variables (i.e., space and direction of travel). The discretized equation results in a linear system of equations that can be solved by linear algebra techniques. Deterministic methods are generally faster than stochastic methods but involve discretization or truncation errors. However, employing certain discretizations can reduce the impact of these errors.

1.2 Discretization

Physically, 3-dimensional space is continuous: a particle could exist at any one of an infinite number of locations. Numerically, it is impossible to compute a solution at an infinite number of locations so we discretize the spatial domain into a small subset of locations. Similarly, a particle can travel in an infinite number of directions so we approximate it traveling in only a few. Discretizing the direction of travel by using the discrete ordinates (S_N) method [9] is common. Particles with various energies may behave or interact with material differently. It is common to approximate an infinite number of particle energies by grouping them into a finite number of energy groups. If a problem evolves through time, we discretize the time continuum with small discrete steps through time. In particular, this research discretizes the energy domain into one group, thereby making the problem energy independent. We also assume the solution of the transport equation has reached steady-state, making the problem time independent. We discretize the two remaining dependent variables (space and direction of travel) but the spatial discretization is the main focus of this

research.

We approximate the continuum of directions that a particle can travel by assuming they only travel in discrete ordinate directions. Figure **??** shows an example of a set of discrete ordinates.

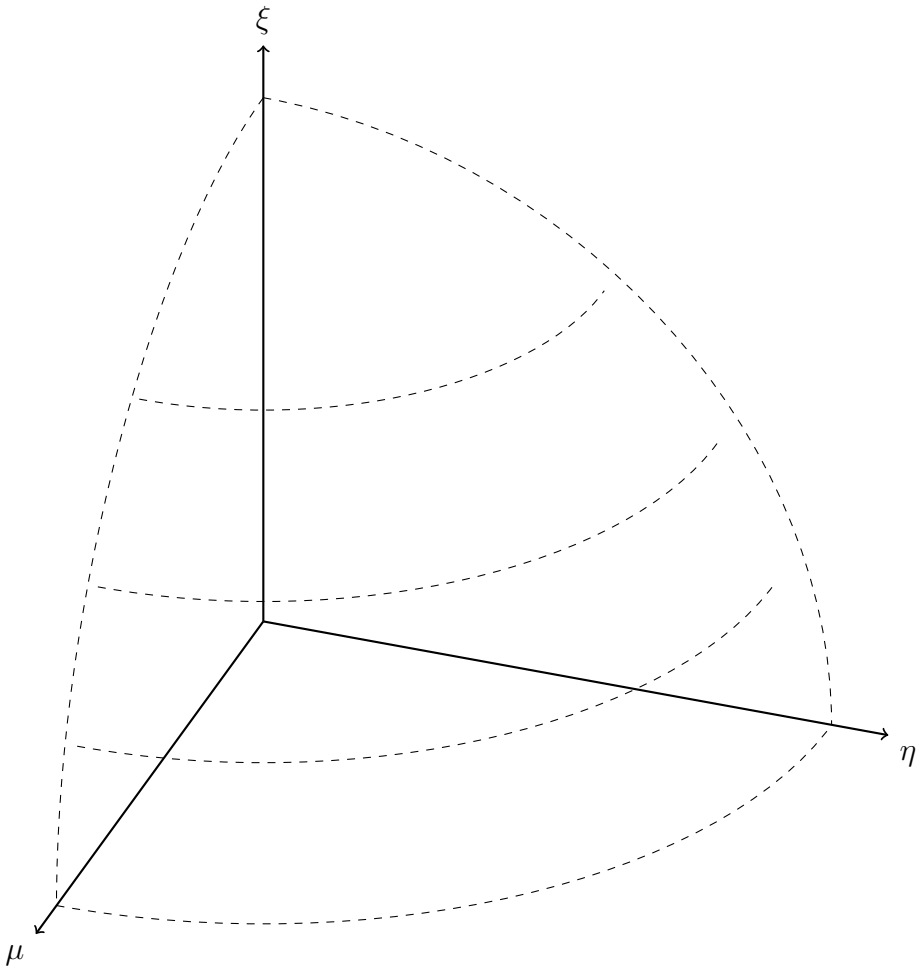


Figure 1: Discrete ordinates.

We utilize the discrete ordinate weights to approximate the integral in Equation 2a by

$$\phi \approx \sum_m w_m \psi_m. \tag{3}$$

There are several methods of spatial discretization that appear in the transport community. Among the most common are characteristic methods [10], finite difference methods [1], finite volume methods, and finite element methods (FEMs) [1]. Typically, the problem domain is divided to a larger number of smaller domains. The equations are then numerically solved on each of these smaller regions where the solution is likely less varying when compared to the entire problem. The FEM was introduced to the transport community in the 1970's. The utility of this method was that it was more accurate than finite differencing methods [**?**]. Since then, radiation transport research using the FEM has proliferated. In particular, the discontinuous FEM (DFEM) has been favored for it's ability to perform in the thick diffusion limit [11] (discussed in Sections 1.3 and **??**). Thus, this research uses the DFEM in which the solution is approximated to have a functional form within each mesh element. More discussion of various discretization methods can be found in Lewis and Miller [1].

Additionally, these radiation transport problems are solved in each of Cartesian, cylindrical (R-Z), and spherical coordinates. The present research is concerned with Cartesian and R-Z geometries. Difficulties arise in R-Z geometry because of the introduction of angular derivatives. While a particle travels in a straight line in direction $\mathbf{\Omega}$, the cosines of the angles relative to the coordinate axes change as the r and z coordinates change.

1.3 Diffusion Limit

High energy density physics (HEDP) problems are examples of scenarios where a particle has a very small mean free path (mfp) compared to the size of a spatial zone. This happens when the total macroscopic cross section σ_t becomes very large, where the mfp is the inverse of the total cross section $\Lambda = \sigma_t^{-1}$. These problems are called

"optically thick". Equation 2a must behave well in the optically thick regime if we are to expect the same from the time and/or energy dependent radiation transport equation. The transport equation behaves well when the spatial mesh is optically thin. But we can assess the behavior of Equation 2a as the problem becomes increasingly optically thick by performing an asymptotic analysis. Specifically, if a small factor ε is used to scale the physical processes of Equation 2a,

$$\mathbf{\Omega}_m \cdot \boldsymbol{\nabla}\psi_m + \frac{\sigma_t}{\varepsilon}\psi_m = \frac{1}{4\pi}\left(\frac{\sigma_t}{\varepsilon} - \varepsilon\sigma_a\right)\int_{4\pi}\psi d\Omega' + \varepsilon S_0 \tag{4}$$

where arguments have been dropped for brevity. Then, as $\varepsilon \to 0$ the mean free path $\Lambda = \varepsilon/\sigma_t \to 0$. In this limit, the problem is said to be optically thick and diffusive. It can be shown [12] that this scaled analytic transport equation limits to the analytic radiation diffusion equation to $O(\varepsilon^2)$. The radiation diffusion equation provides accurate solutions to optically thick and diffusive problems in the problem interior. Hence, concluding that the transport equation limits to the diffusion equation is physically meaningful. Optically thick and diffusive problems that are typically solved with the radiation diffusion equation can also be solved using the transport equation.

The diffusion equation has substantially fewer degrees of freedom making it much quicker to solve numerically. This benefit is accompanied by some drawbacks. The diffusion equation cannot resolve optically thin problems nor problems with strong material discontinuities, including vacuum boundaries. Partly due to these drawbacks, our present work employs the transport equation to solve problems that are optically thick and diffusive.

The source iteration (SI) method [1] is commonly employed to solve the discretized

transport equation. The algorithm

$$\mathbf{\Omega} \cdot \mathbf{\nabla} \psi^{(\ell+1/2)} + \sigma_t \psi^{(\ell+1/2)} = \frac{1}{4\pi} \sigma_s \phi^{(\ell)} + S_0 \tag{5a}$$

$$\phi^{(\ell+1/2)} = \sum_m w_m \psi_m^{(\ell+1/2)} \tag{5b}$$

$$\phi^{(\ell+1)} = \phi^{(\ell+1/2)} \tag{5c}$$

describes the calculation of the angular flux using the lagged scalar flux followed by an update to the scalar flux using Equation **??**.

The transport equation can converge arbitrarily slowly in these optically thick regimes [13]. To speed up the SI, one option is to refine the mesh until the optical thickness of a typical mesh cell is on the order of a mean-free-path to, effectively, solve an optically thin problem in each mesh zone. This option is not very efficient because it can introduce a large number of degrees of freedom to the problem, thereby increasing the solution time. Alternatively, acceleration techniques can be applied to the SI to compensate for slow convergence. Diffusion synthetic acceleration is commonly used to accelerate the source iteration and is discussed in more detail below.

1.4 Outline

The remainder of this thesis is outlined as follows.

2 High Order DFEM

This section describes the methods to be carried out to accomplish the proposed research. We also consider some potential issues that we may encounter with some initial thoughts about mitigation.

2.1 Basis Functions

The basis functions are mesh zone dependent. So, by transforming each physical element into the reference element, the basis functions are uniform for each physical element. The basis functions are allowed to be unique to each element. However, in practice, each mesh element is transformed to the reference element, removing the mesh dependence for the basis functions.

One requirement for the basis functions is that they are unity at the integration point they "live" at and zero at all of the others. In 2-D, the general form of the first-order polynomial basis function is

$$b(x, y) = axy + bx + cy + d, \tag{6}$$

and the second-order basis function,

$$b(x, y) = ax^2y^2 + bx^2y + cx^2 + dxy^2 + exy + fx + gy^2 + hy + j. \tag{7}$$

The location of the integration points is important. We may place these evenly across the reference element or at Gaussian quadrature locations. On the unit square $[0, 1] \times [0, 1]$, the Gaussian integration points are located at the cross product of the nodes on $[0, 1]$. Listed for linear, quadratic, and cubic finite elements in Table 1, the Gaussian integration point locations were transformed from the traditional $[-1, 1]$ to our reference element length $[0, 1]$.

Once the finite element order and the set of integration points is determined,

Table 1: Gaussian quadrature locations.

FE order	Points on $[-1,1]$	Points on $[0,1]$
1	$\pm\frac{1}{\sqrt{3}}$	$\frac{1}{3+\sqrt{3}}$ $1-\frac{1}{3+\sqrt{3}}$
2	0 $\pm\sqrt{\frac{3}{5}}$	$\frac{1}{5+\sqrt{15}}$ 0.5 $1-\frac{1}{5+\sqrt{15}}$
3	$\pm\sqrt{\frac{3}{7}-\frac{2}{7}\sqrt{\frac{6}{5}}}$ $\pm\sqrt{\frac{3}{7}+\frac{2}{7}\sqrt{\frac{6}{5}}}$	$\frac{10+\sqrt{30}}{35+\sqrt{35(15-2\sqrt{30})}}$ $\frac{10-\sqrt{30}}{35+\sqrt{35(15-2\sqrt{30})}}$ $1-\frac{10+\sqrt{30}}{35+\sqrt{35(15-2\sqrt{30})}}$ $1-\frac{10-\sqrt{30}}{35+\sqrt{35(15-2\sqrt{30})}}$

the linear system can be arranged to determine the coefficients. For example, to determine the first-order basis function that "lives" at integration point (x_1, y_1) (and hence is equal to unity at that integration point and is zero at the remaining three), we assemble the system of equations

$$\begin{bmatrix} x_1y_1 & x_1 & y_1 & 1 \\ x_2y_2 & x_2 & y_2 & 1 \\ x_3y_3 & x_3 & y_3 & 1 \\ x_4y_4 & x_4 & y_4 & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}. \tag{8}$$

Moving the 1 inside the solution vector on the right-hand-side determines which basis function coefficients will be obtained. The vector $[0,0,1,0]^T$ will return the basis

function for the third integration point at (x_3, y_3).

3 Meshes with Curved Surfaces

The complicated shapes of each mesh zone create a challenge by having to solve the discretized equations for each unique mesh zone. We can avoid having to solve a unique set of equations for each mesh zone by transforming the mesh zone into the reference element. Each mesh zone will have a unique transformation but an identical set of basis functions to obtain the solution on the reference element.

These meshes add complications to the numerical methods used to solve the transport equation. In particular, cycles can be present during the spatial sweep of the mesh. It is common to solve for a single mesh zone using incident angular flux information and propagate that angular flux from mesh zone to mesh zone, sweeping through the grid. However, if any particular mesh zone has both incident and outgoing angular fluxes to another mesh zone, they depend upon each other in a cyclic manner. "Breaking the cycle" is some fashion is necessary to perform the numerical computation. Alternatively, instead of sweeping through the grid, we utilize MFEM to generate the solution matrix for all of the mesh zones for the entire problem. This is computationally intensive, but it bypasses the need to consider any cycles that may occur.

3.1 Transformation

We set up the system of equations (Section **??**) on each individual mesh zone after we transform it to the reference element. After performing the following integrations, we map the solution back to the physical element. The bi-quadratic mapping from the reference element to the physical element, shown in Figure 1, has the following

functional form

$$\begin{bmatrix} x(\rho,\kappa) \\ y(\rho,\kappa) \end{bmatrix} = \sum_{i=1}^{J_k} \sum_{j=1}^{J_k} \begin{bmatrix} x_{ij} \\ y_{ij} \end{bmatrix} N_i(\rho) N_j(\kappa) \tag{9}$$

where

$$N_l(\xi) = \begin{cases} (2\xi - 1)(\xi - 1), & l = 1 \\ 4\xi(1 - \xi), & l = 2 \\ \xi(2\xi - 1), & l = 3 \end{cases} \tag{10}$$

are the quadratic basis functions that have support points at typical locations shown in the left image of Figure 1. The (x_{ij}, y_{ij}) coordinates are the locations of the support points in the physical element and are generally known. For example, the node (x_{12}, y_{12}) is the location on the physical zone that is mapped from $(\rho, \kappa) = (0, 0.5)$ on the reference element.

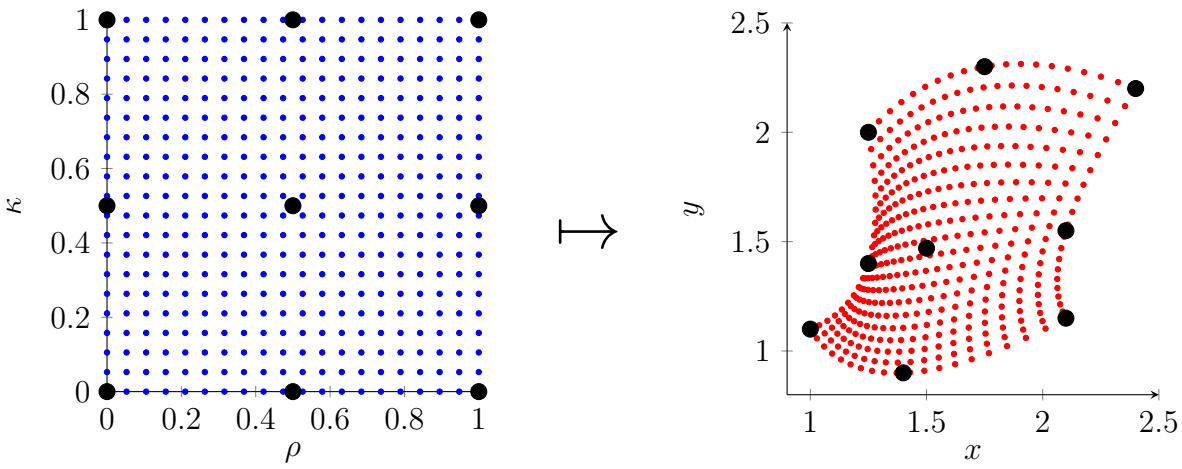


Figure 2: Example of mapping the reference element to a physical element. Switch these around to map the physical element to the reference element.

The determinant of the Jacobian of the transformation,

$$\det(J) = \begin{vmatrix} \frac{\partial x}{\partial \rho} & \frac{\partial y}{\partial \rho} \\ \frac{\partial x}{\partial \kappa} & \frac{\partial y}{\partial \kappa} \end{vmatrix}, \tag{11}$$

is required for the volume integrations in the next section.

4 Diffusion Synthetic Acceleration

This section describes the methods to be carried out to accomplish the proposed research. We also consider some potential issues that we may encounter with some initial thoughts about mitigation.

4.1 Modified Interior Penalty DSA

4.1.1 Methodology

4.1.2 Fourier Analysis

4.1.3 Results

4.1.4 As a Preconditioner

4.2 MIP DSA with Robin Boundary Conditions

The diffusion synthetic acceleration (DSA) algorithm first solves the RTE, called a "transport sweep", followed by a diffusion solve for each source iteration. The RTE solve is performed exactly as before (Eqs. 5) with notation changing to indicate a "half step":

$$\boldsymbol{\Omega} \cdot \boldsymbol{\nabla} I^{(l+1/2)} + \sigma_t I^{(l+1/2)} = \sigma_s E^{(l)} + S_0 \tag{12a}$$

$$E^{(l+1/2)} = \sum_{m=1}^{M} w_m \; I_m^{(l+1/2)} \left(\boldsymbol{r}, \boldsymbol{\Omega}\right) \tag{12b}$$

(Recall the energy density E defined by Equation 6.) Next, we solve the diffusion equation with a modified source term utilizing the half step and previous iteration RTE solutions.

$$-\boldsymbol{\nabla} \cdot D \boldsymbol{\nabla} \phi^{(I+1/2)} + \sigma_a \phi^{(l+1/2)} = \sigma_s \left(E^{(l+1/2)} - E^{(l)}\right) \tag{12c}$$

The diffusion solution, $\phi^{(l+1/2)}$, becomes the correction factor at the half step.

This correction factor is added to the energy density at the half step to complete the full DSA iteration.

$$E^{(l+1)} = E^{(l+1/2)} + \phi^{(l+1/2)} \tag{12d}$$

This iterative process continues until the convergence criteria

$$\left\| E^{(l+1)} - E^{(l)} \right\|_{\infty} < \varepsilon_{\text{conv}} \left(1 - \rho\right) \left\| E^{l+1} \right\|_{\infty}, \tag{13}$$

is met, where $\varepsilon_{\text{conv}}$ is a small number.

Wang and Ragusa [14] proposed the modified interior penalty (MIP) equations for discretizing the DSA equation using the DGFEM:

$$\begin{aligned} b_{MIP}\left(\phi, w\right) = &\left(\sigma_a \phi, w\right)_{\mathbb{V}} + \left(D \boldsymbol{\nabla} \phi, \boldsymbol{\nabla} w\right)_{\mathbb{V}} \\ &+ \left(\kappa_e [\![\phi]\!], [\![w]\!]\right)_{\partial \mathbb{V}^i} + \left([\![\phi]\!], \{\!\{D \partial_n w\}\!\}\right)_{\partial \mathbb{V}^i} + \left(\{\!\{D \partial_n \phi\}\!\}, [\![w]\!]\right)_{\partial \mathbb{V}^i} \\ &+ \left(\kappa_e \phi, w\right)_{\partial \mathbb{V}^d} - \frac{1}{2}\left(\phi, D \partial_n w\right)_{\partial \mathbb{V}^d} - \frac{1}{2}\left(D \partial_n \phi, w\right)_{\partial \mathbb{V}^d} \end{aligned} \tag{14}$$

and

$$l_{MIP}\left(w\right) = \left(Q_0, w\right)_{\mathbb{V}} \tag{15}$$

where b_{MIP} is the bilinear form, l_{MIP} is the linear form, w is the weight/test function, $\mathbb{V}$ denotes the element volume, $\partial \mathbb{V}^i$ is the internal edges, $\partial \mathbb{V}^d$ is on the problem boundary, ∂_n is the partial derivative perpendicular to the edge (i.e. $\boldsymbol{\nabla} \cdot \hat{n}$). The following definitions accompany Equations 15 and 16.

$$[\![\phi]\!] = \phi^+ - \phi^- \tag{16}$$

$$\{\!\{\phi\}\!\} = \frac{(\phi^+ + \phi^-)}{2} \tag{17}$$

$$Q_0 = \sigma_s \left(E^{(l+1/2)} - E^{(l)}\right) \tag{18}$$

$$\kappa_e^{IP} = \begin{cases} \frac{c\left(p^+\right)}{2}\frac{D^+}{h_\perp^+} + \frac{c\left(p^-\right)}{2}\frac{D^-}{h_\perp^-}, & \text{on interior edges (i.e., } \boldsymbol{r} \in E_h^i) \\ c\left(p\right)\frac{D}{h_\perp}, & \text{on boundary edges } \left(\text{i.e., } \boldsymbol{r} \in \partial D^d\right) \end{cases} \tag{19}$$

$$\kappa_e = \max\left(\kappa_e^{IP}, \frac{1}{4}\right) \tag{20}$$

$$c\left(p\right) = Cp\left(p+1\right) \tag{21}$$

where C is a an arbitrary constant, $D^\pm$ is the diffusion coefficient, $\phi^\pm$ is the scalar flux, $p^\pm$ is the order of finite elements, and $h_\perp^\pm$ is the perpendicular length of the cell, where the $\pm$ denotes either side of an element boundary. Equation 21 is a "switch" between two methods (interior penalty and diffusion conforming form), one of which is stable for optically thin regions and the other for optically thick regions. Wang and Ragusa [14] used $C = 2$ and Turcksin and Ragusa [15] used $C = 4$.

4.2.1 Zero Incident Current

Kanschat [16] shows that Equations 15 and 16 employ Nitsche's method for "a fully conforming method of treating Dirichlet boundary values." The boundary terms $\left(\partial\mathbb{V}^d\right)$ in this form are homogeneous Dirichlet boundary conditions. The result is that the DSA correction for the energy density at the problem boundaries is zero, so the energy density is only updated by the RTE solution. That is, the DSA correction only accelerates the interior solution. Consequently, the energy densities on the problem boundary is only subjected to the RTE solution source iterations.

Instead, a DSA update equation should incorporate Robin boundary conditions

(zero incident partial current) on the boundaries,

$$\boldsymbol{J}_{-} = 0 = \frac{1}{4}\phi + \frac{1}{2}D\boldsymbol{\nabla}\phi \cdot \hat{n}, \tag{22}$$

$$-\frac{1}{2}\phi = D\boldsymbol{\nabla}\phi \cdot \hat{n}, \tag{23}$$

thereby allowing a correction of the boundary energy densities. This boundary condition requires modification of Equation 15. Three implementaion methods are proposed here. Method 1 substitutes Equation 24 into Equation 15:

$$\begin{aligned} b_{MIP,1}(\phi, w) = {} & (\sigma_a\phi, w)_{\mathbb{V}} + (D\boldsymbol{\nabla}\phi, \boldsymbol{\nabla}w)_{\mathbb{V}} \\ & + (\kappa_e[\![\phi]\!], [\![w]\!])_{\partial\mathbb{V}^i} + ([\![\phi]\!], \{\!\{D\partial_n w\}\!\})_{\partial\mathbb{V}^i} + (\{\!\{D\partial_n\phi\}\!\}, [\![w]\!])_{\partial\mathbb{V}^i} \\ & + (\kappa_e\phi, w)_{\partial\mathbb{V}^d} - \frac{1}{2}(\phi, D\partial_n w)_{\partial\mathbb{V}^d} + \frac{1}{4}(\phi, w)_{\partial\mathbb{V}^d} \end{aligned} \tag{24}$$

Method 2 is very similar to Method 1 with one term removed:

$$\begin{aligned} b_{MIP,2}(\phi, w) = {} & (\sigma_a\phi, w)_{\mathbb{V}} + (D\boldsymbol{\nabla}\phi, \boldsymbol{\nabla}w)_{\mathbb{V}} \\ & + (\kappa_e[\![\phi]\!], [\![w]\!])_{\partial\mathbb{V}^i} + ([\![\phi]\!], \{\!\{D\partial_n w\}\!\})_{\partial\mathbb{V}^i} + (\{\!\{D\partial_n\phi\}\!\}, [\![w]\!])_{\partial\mathbb{V}^i} \\ & + (\kappa_e\phi, w)_{\partial\mathbb{V}^d} + \frac{1}{4}(\phi, w)_{\partial\mathbb{V}^d} \end{aligned} \tag{25}$$

Method 3 stems from a different approach. After performing the integration by parts on the diffusion term,

$$-(\boldsymbol{\nabla} \cdot D\boldsymbol{\nabla}\phi, w)_{\mathbb{V}} = (D\boldsymbol{\nabla}\phi, \boldsymbol{\nabla}w)_{\mathbb{V}} - (D\boldsymbol{\nabla}\phi \cdot \hat{n}, w)_{\partial\mathbb{V}^i} - (D\boldsymbol{\nabla}\phi \cdot \hat{n}, w)_{\partial\mathbb{V}^d}, \tag{26}$$

we employ Equation 24:

$$-(\boldsymbol{\nabla} \cdot D\boldsymbol{\nabla}\phi, w)_{\mathbb{V}} = (D\boldsymbol{\nabla}\phi, \boldsymbol{\nabla}w)_{\mathbb{V}} - (D\boldsymbol{\nabla}\phi \cdot \hat{n}, w)_{\partial\mathbb{V}^i} + \frac{1}{2}(\phi, w)_{\partial\mathbb{V}^d} \tag{27}$$

Thus, Method 3 is

$$b_{MIP,3}(\phi, w) = (\sigma_a \phi, w)_{\mathbb{V}} + (D\boldsymbol{\nabla}\phi, \boldsymbol{\nabla} w)_{\mathbb{V}} + (\kappa_e [\![\phi]\!], [\![w]\!])_{\partial \mathbb{V}^i} + ([\![\phi]\!], \{\!\{D\partial_n w\}\!\})_{\partial \mathbb{V}^i} + (\{\!\{D\partial_n \phi\}\!\}, [\![w]\!])_{\partial \mathbb{V}^i} + \frac{1}{2}(\phi, w)_{\partial \mathbb{V}^d} \quad (28)$$

We will implement these three methods and investigate their impact on the solution behavior. We will first solve a 1-D analytic diffusion equation problem with zero incident current boundary conditions to evaluate our implementation.

4.2.2 Fourier Analysis

4.2.3 Results

4.2.4 As a Preconditioner

5 R-Z Geometry

Solving the transport equation in different coordinate systems may provide simpler ways of modeling a particular geometry or symmetry. In this section, we derive the R-Z transport equation to be solved. It assumes there is no variation in the azimuthal direction (of a cylinder), hence problems in R-Z geometry look very similar to problems in X-Y geometry. The streaming operator in cylindrical geometry is [1]

$$\mathbf{\Omega} \cdot \mathbf{\nabla}\psi = \frac{\mu}{r}\frac{\partial}{\partial r}(r\psi) + \frac{\eta}{r}\frac{\partial \psi}{\partial \zeta} + \xi\frac{\partial \psi}{\partial z} - \frac{1}{r}\frac{\partial}{\partial \omega}(\eta\psi), \tag{29}$$

where $\mathbf{\Omega}$ is the direction of travel unit vector, ψ is the angular flux, and

$$\mu \equiv \mathbf{\Omega} \cdot \hat{e}_r = \sqrt{1-\xi^2}\cos\omega = \sin(\theta)\cos(\omega), \tag{30}$$

$$\eta \equiv \mathbf{\Omega} \cdot \hat{e}_\theta = \sqrt{1-\xi^2}\sin\omega = \sin(\theta)\sin(\omega), \tag{31}$$

$$\xi \equiv \mathbf{\Omega} \cdot \hat{e}_z = \cos(\theta). \tag{32}$$

The variables μ, η, ξ, ω, and θ are shown in the cylindrical coordinate system in Figure 2. We assume there is no solution variation in the azimuthal direction, i.e.

$$\frac{\partial \psi}{\partial \zeta} \equiv 0, \tag{33}$$

which simplifies the streaming term to

$$\mathbf{\Omega} \cdot \mathbf{\nabla}\psi = \frac{\mu}{r}\frac{\partial}{\partial r}(r\psi) + \xi\frac{\partial \psi}{\partial z} - \frac{1}{r}\frac{\partial}{\partial \omega}(\eta\psi). \tag{34}$$

The transport equation in R-Z geometry is then

$$\frac{\mu}{r}\frac{\partial}{\partial r}r\psi\left(r,z,\mathbf{\Omega}\right) + \xi\frac{\partial}{\partial z}\psi\left(r,z,\mathbf{\Omega}\right) - \frac{1}{r}\frac{\partial}{\partial \omega}\eta\psi\left(r,z,\mathbf{\Omega}\right) + \sigma_t\left(r,z\right)\psi\left(r,z,\mathbf{\Omega}\right) = \frac{1}{4\pi}\int_{4\pi}\sigma_s\left(r,z\right)I\left(r,z,\mathbf{\Omega}'\right)d\Omega' + S_0\left(r,z,\mathbf{\Omega}\right) \tag{35}$$

where σ_t is the total cross section, σ_s is the scattering cross section, and S_0 is an isotropic source as before.

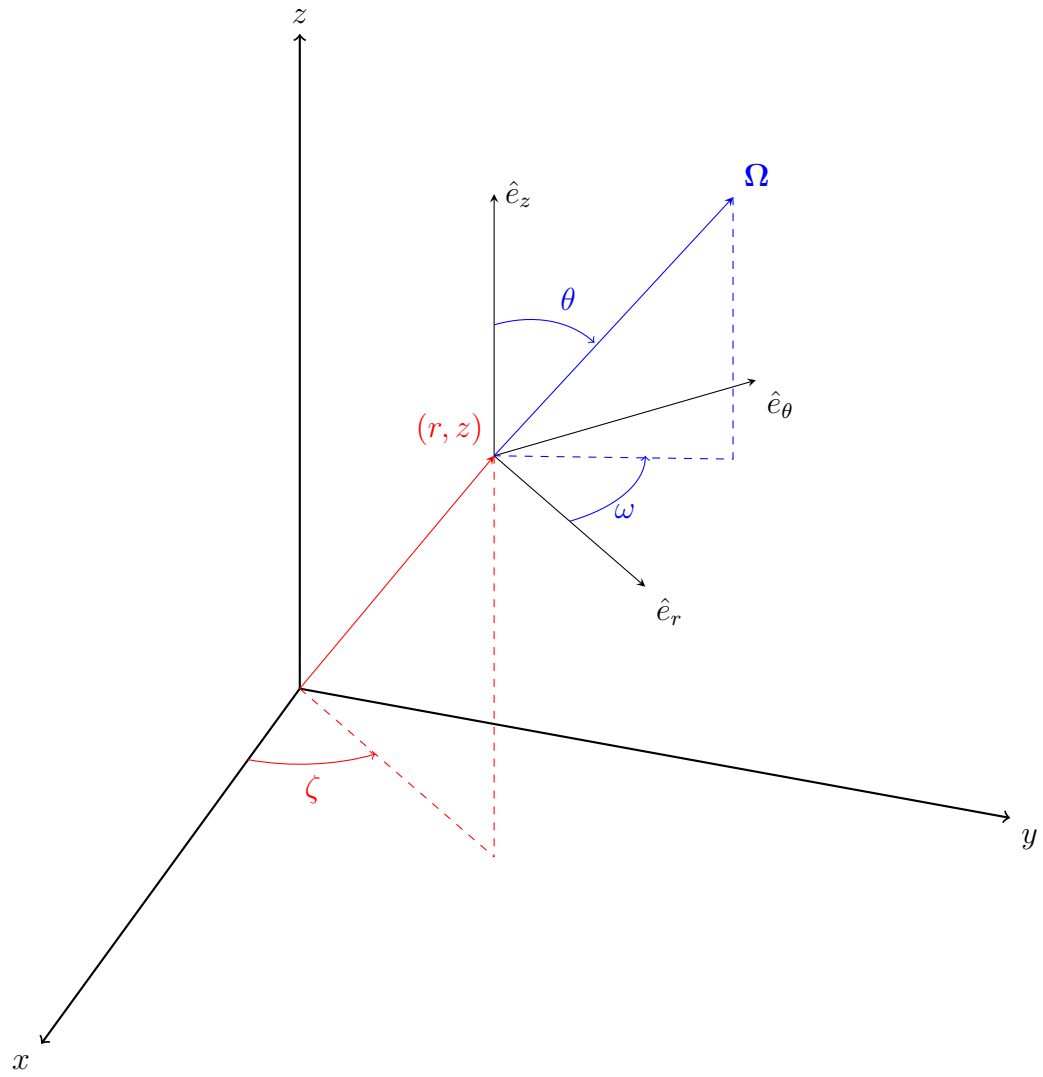


Figure 3: Cylindrical space-angle coordinate system showing the position (r, z) and direction of travel $\boldsymbol{\Omega}$.

5.1 Angular Discretization

Discretizing Equation 36 with a level-symmetric angular quadrature results in

$$\frac{\mu_{n,m}}{r}\frac{\partial}{\partial r}r\psi_{n,m}\left(r,z\right)+\xi_n\frac{\partial}{\partial z}\psi_{n,m}\left(r,z\right)-\frac{1}{r}\frac{\partial}{\partial\omega}\eta_{n,m}\psi_{n,m}\left(r,z\right)+\sigma_t\left(r,z\right)\psi_{n,m}\left(r,z\right)$$
$$=\frac{1}{4\pi}\int_{4\pi}\sigma_s\left(r,z\right)I\left(r,z,\boldsymbol{\Omega}'\right)d\Omega'+S_0\left(r,z,\boldsymbol{\Omega}\right) \quad (36)$$

for direction $\boldsymbol{\Omega}_{n,m}$, where index n describes a level of quadrature with constant ξ and the m index denotes the quadrature point on that level. The (n, m) indexing is shown

in Figure 3.

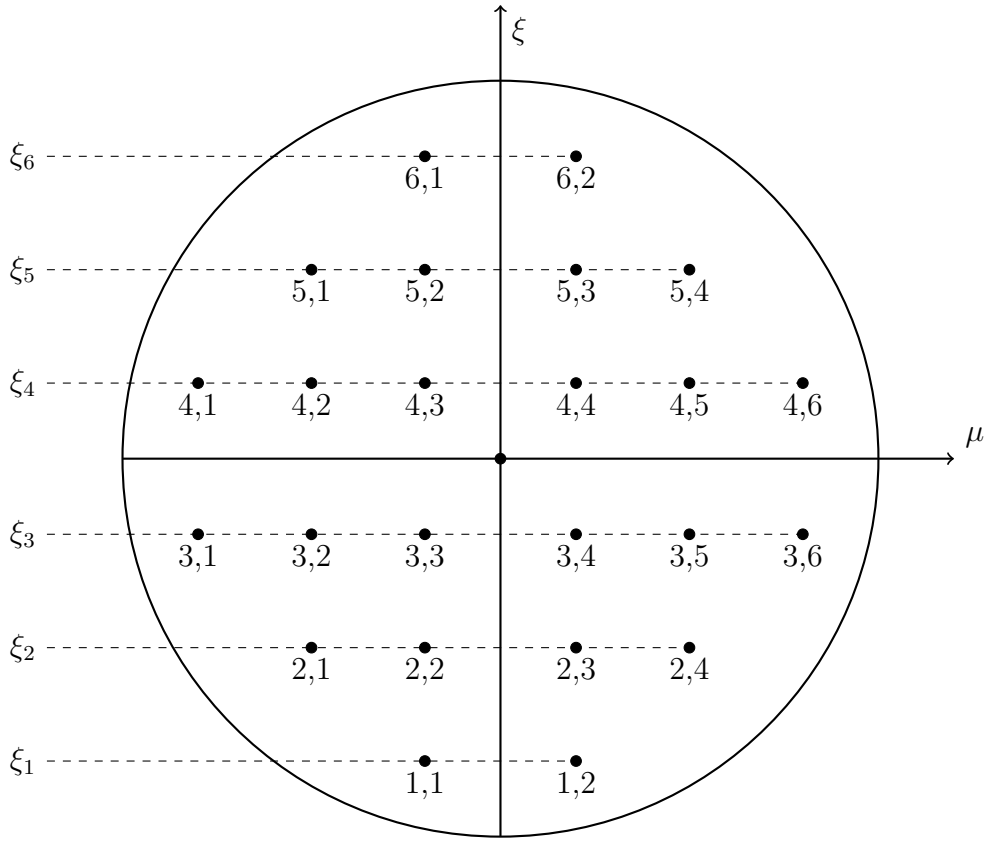


Figure 4: Angular discretization showing (ξ, μ) pairs; adapted from [1]

One of the major challenges is handling the anglar derivative term. Lewis and Miller [1] describes an approximation for the partial derivative of the intensity with respect to ω:

$$-\frac{1}{r}\frac{\partial}{\partial\omega}\eta_{m,n}\psi_{n,m}\left(r,z\right) = \frac{\alpha^n_{m+1/2}\psi_{n,m+1/2}(r,z) - \alpha^n_{m-1/2}\psi_{n,m-1/2}(r,z)}{rw_{n,m}} \tag{37}$$

where $\alpha^n_{m+1/2}$ and $\alpha^n_{m-1/2}$ are angular differencing coefficients, and $w_{n,m}$ is the angular

quadrature weight. We substitute this into Equation 47,

$$\frac{\mu_{n,m}}{r}\frac{\partial}{\partial r}r\psi_{n,m}\left(r,z\right)+\xi_n\frac{\partial}{\partial z}\psi_{n,m}\left(r,z\right) + \frac{\alpha^n_{m+1/2}\psi_{m+1/2,n}(r,z)-\alpha^n_{m-1/2}\psi_{m-1/2,n}(r,z)}{rw_{n,m}}+\sigma_t\left(r,z\right)\psi_{n,m}\left(r,z\right) = \frac{1}{4\pi}\int_{4\pi}\sigma_s\left(r,z\right)\psi\left(r,z,\mathbf{\Omega}'\right)d\Omega'+\frac{1}{4\pi}S_0\left(r,z\right) \quad (38)$$

Here, we pause to notice that there are similarities and differences between our Cartesian discretization. The absorption term, axial derivative term, and right-hand-side are the same in both coordinate systems. The differences arise in the radial and angular derivative terms.

After multiplying through by the radius r, the radial derivative term has a factor of r inside the derivative. The angular derivative term is also new and does not resemble a mass matrix so MFEM will require additional modification.

Requiring Equation 39 to satisfy the uniform infinite medium solution results in the condition,

$$\alpha^n_{m+1/2}=\alpha^n_{m-1/2}-\mu_{n,m}w_{n,m} \quad (39)$$

If $\alpha^n_{1/2}$ is known, then the remaining coefficients are uniquely determined. To find $\alpha^n_{1/2}$, we require that Equation 39 satisfy the conservation equation (Eq. 36). Given a quadrature set with an even number of $\mu_{n,m}$ values, setting $\alpha^n_{1/2}=0$ results in $\alpha^n_{M_n+1/2}=0$ per Equation 40 and the conservation equation is satisfied.

A relationship between $\psi_{n,m}$, $\psi_{n,m+1/2}$, and $\psi_{n,m-1/2}$ must be established. A weighted diamond difference scheme has been established by Morel and Montry [17],

$$\psi_{n,m}(r,z)=\tau_{n,m}\psi_{n,m+1/2}+\left(1-\tau_{n,m}\right)\psi_{n,m-1/2} \quad (40)$$

where $\tau_{n,m}$ linearly interpolates μ:

$$\tau_{n,m}=\frac{\mu_{n,m}-\mu_{n,m-1/2}}{\mu_{n,m+1/2}-\mu_{n,m-1/2}} \quad (41)$$

with

$$\mu_{n,m+1/2} = \sqrt{1-\xi_n^2}\cos\left(\varphi_{n,m+1/2}\right) \tag{42}$$

$$\varphi_{n,m+1/2} = \varphi_{n,m-1/2} + \pi\frac{w_{n,m}}{w_n} \tag{43}$$

$$w_n = \sum_{m=1}^{M_n} w_{n,m} \tag{44}$$

We take Equation 39, multiply through by r and perform a product rule on the radial derivative term,

$$\begin{aligned}\mu_{n,m}\left[\psi_{n,m}(r,z) + r\frac{\partial}{\partial r}\psi_{n,m}(r,z)\right] + r\xi_n\frac{\partial}{\partial z}\psi_{n,m}(r,z) \\ + \frac{\alpha^n_{m+1/2}\psi_{m+1/2,n}(r,z) - \alpha^n_{m-1/2}\psi_{m-1/2,n}(r,z)}{w_{n,m}} + r\sigma_t(r,z)\,\psi_{n,m}(r,z) \\ = \frac{r}{4\pi}\int_{4\pi}\sigma_s(r,z)\,\psi(r,z,\mathbf{\Omega}')\,d\Omega' + \frac{r}{4\pi}S_0(r,z).\end{aligned} \tag{45}$$

We solve Equation 41 for $\psi_{n,m+1/2}$, perform a substitution, and move the known quantities to the right-hand-side,

$$\begin{aligned}\mu_{n,m}r\frac{\partial}{\partial r}\psi_{n,m}(r,z) + r\xi_n\frac{\partial}{\partial z}\psi_{n,m}(r,z) + \mu_{n,m}\psi_{n,m}(r,z) \\ + \frac{\alpha^n_{m+1/2}}{\tau_{n,m}w_{n,m}}\psi_{n,m}(r,z) + r\sigma_t(r,z)\,\psi_{n,m}(r,z) \\ = \frac{r}{4\pi}\int_{4\pi}\sigma_s(r,z)\,\psi(r,z,\mathbf{\Omega}')\,d\Omega' + \frac{r}{4\pi}S_0(r,z) \\ + \left(\frac{1-\tau_{n,m}}{\tau_{n,m}}\frac{\alpha^n_{m+1/2}}{w_{n,m}} + \frac{\alpha^n_{m-1/2}}{w_{n,m}}\right)\psi_{n,m-1/2}(r,z).\end{aligned} \tag{46}$$

Given a level-symmetric quadrature set, all of the $\alpha^n_{n,m\pm1/2}$ and $\tau_{n,m}$ values can be computed. We solve the starting direction equation to obtain $\psi_{n,1/2}$. That is, we solve the X-Y system for directions $\mathbf{\Omega}_{n,1/2}$,

$$\mathbf{\Omega}_{n,1/2}\cdot\boldsymbol{\nabla}\psi_{n,1/2} + \sigma_t\psi_{n,1/2} = \frac{1}{4\pi}\sigma_s\phi + \frac{1}{4\pi}S_0 \tag{47}$$

There is an alternate angular discretization method developed by Warsa and Prinja [18]. Instead of finding an approximation for the angular derivative, they

perform a product rule:

$$\frac{\partial\psi}{\partial\omega} \equiv \frac{\partial\mu}{\partial\omega}\frac{\partial\psi}{\partial\mu} \tag{48}$$

Since,

$$\frac{\partial\mu}{\partial\omega} \equiv -\xi, \tag{49}$$

The angular derivative can be written

$$\frac{\partial\psi}{\partial\omega} \equiv -\xi\frac{\partial\psi}{\partial\mu} \tag{50}$$

Here, an approximation for the μ-derivative must be established.

5.2 Spatial Discretization

The finite element discretization is performed here. The methodology is similar to the Cartesian geometry. First, we subdivide a problem domain using a spatial mesh. Then, we multiply Equation 47 by a test function and integrate over the volume of a single mesh zone,

$$\begin{aligned}(r\boldsymbol{\Omega}_{n,m}\cdot\boldsymbol{\nabla}\psi_{n,m}, v_i)_{\mathbb{D}} + (\mu_{n,m}\psi_{n,m}, v_i)_{\mathbb{D}} \\ + \left(\frac{\alpha^n_{m+1/2}}{\tau_{n,m}w_{n,m}}\psi_{n,m}, v_i\right)_{\mathbb{D}} + (r\sigma_t\psi_{n,m}, v_i)_{\mathbb{D}} \\ = \left(\frac{r}{4\pi}\int_{4\pi}\sigma_s\psi d\Omega', v_i\right)_{\mathbb{D}} + \left(\frac{r}{4\pi}S_0, v_i\right)_{\mathbb{D}} \\ + \left(\left(\frac{1-\tau_{n,m}}{\tau_{n,m}}\frac{\alpha^n_{m+1/2}}{w_{n,m}} + \frac{\alpha^n_{m-1/2}}{w_{n,m}}\right)\psi_{n,m-1/2}, v_i\right)_{\mathbb{D}},\end{aligned} \tag{51}$$

where the Cartesian gradient operator is used and the inner product notation,

$$(a,b)_{\mathbb{D}} \equiv \int_{\mathbb{D}} ab, \tag{52}$$

is used. We perform an integration by parts,

$$\begin{aligned}(r\mathbf{\Omega}_{n,m}\cdot\hat{n}\psi_{n,m},v_i)_{\partial\mathbb{D}}-(r\psi_{n,m},\mathbf{\Omega}_{n,m}\cdot\boldsymbol{\nabla}v_i)_{\mathbb{D}}+(\mu_{n,m}\psi_{n,m},v_i)_{\mathbb{D}}\\ +\left(\frac{\alpha^n_{m+1/2}}{\tau_{n,m}w_{n,m}}\psi_{n,m},v_i\right)_{\mathbb{D}}+(r\sigma_t\psi_{n,m},v_i)_{\mathbb{D}}\\ =\left(\frac{r}{4\pi}\int_{4\pi}\sigma_s\psi d\Omega',v_i\right)_{\mathbb{D}}+\left(\frac{r}{4\pi}S_0,v_i\right)_{\mathbb{D}}\\ +\left(\left(\frac{1-\tau_{n,m}}{\tau_{n,m}}\frac{\alpha^n_{m+1/2}}{w_{n,m}}+\frac{\alpha^n_{m-1/2}}{w_{n,m}}\right)\psi_{n,m-1/2},v_i\right)_{\mathbb{D}},\end{aligned}\tag{53}$$

to obtain our angular and spatially discretized R-Z transport equation.

WE PERFORMED SOME STUDIES TO MAKE SURE IT'S RIGHT...

We first solved a uniform infinite medium problem with $\sigma_t = 1.0$, $\sigma_s = 0.3$, and $S_0 = 0.7$ for 1^{st}-order FEM on a 2^{nd}-order mesh using S_4 level-symmetric angular quadrature. The solution, shown in Figure 4, demonstrates we get the exact flat solution of $\phi = 1.0$.

We tested several MMS problems as well. First, we defined the manufactured solution

$$\psi = (1-\mu^2)(1-\xi^2)\sin\left(\frac{\pi}{2}r\right)\cos(\pi z)\tag{54}$$

with 2^{nd}-order FEM, Orthogonal quadrilateral mesh, $\sigma_t = 1.0$, $\sigma_s = 0.3$, $S_0 = 0.7$, S_4 level-symmetric angular quadrature. The solution is shown in Figure 5 and the L^2-error was 0.132.

Removing the angular dependence in the manufactured solution,

$$\psi = \sin\left(\frac{\pi}{2}r\right)\cos(\pi z),\tag{55}$$

increased the accuracy of our DGFEM approximation. Shown in Figure 6, the L^2-error was reduced to 4.59×10^{-5}.

Bailey et al. [**?**] showed 2^{nd}-order convergence using PWLD and BLD using the

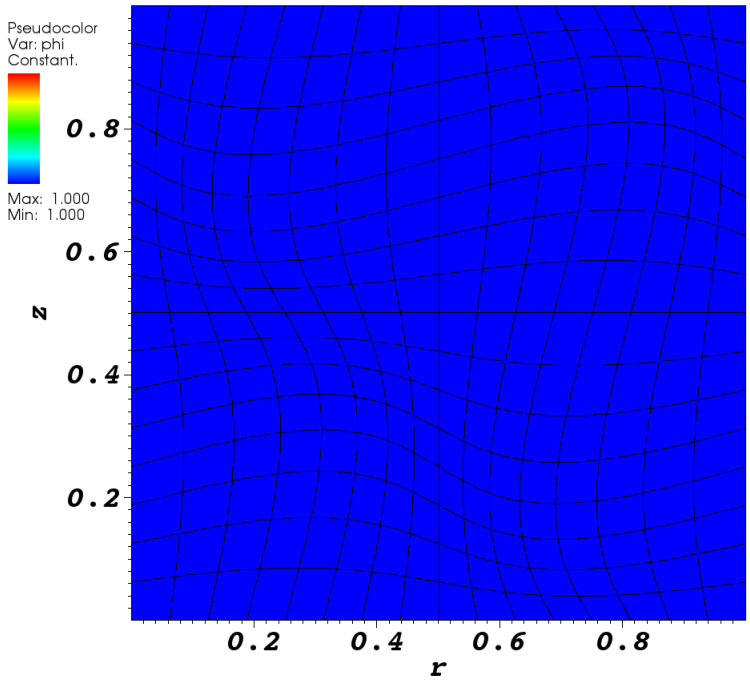


Figure 5: Uniform infinite medium solution.

manufactured solution

$$\psi_{\text{MMS}}(r,z) = (\sin(\pi r) + 1 - r)\sin(\pi z), \tag{56}$$

for $\sigma_t = 3\ \text{cm}^{-1}$ and $\sigma_s = 0.9999\sigma_t$. We solved this same problem using $p = \{1,2,4,6,8\}$ on an orthogonal and 2^{nd}-order curved mesh using S_8 level-symmetric angular quadrature. The incident angular flux is equal to Equation **??**. Figure **??** shows the $p = 2$ solution on a 2^{nd}-order mesh.

The spatial convergence study performed by Bailey et al. demonstrated 2^{nd}-order converge for their 1^{st}-order methods. Figures **??** and **??** demonstrate $O(p+1)$ convergence on an orthogonal mesh and 2^{nd}-order mesh, respectively. Reference lines are also provided for comparison.

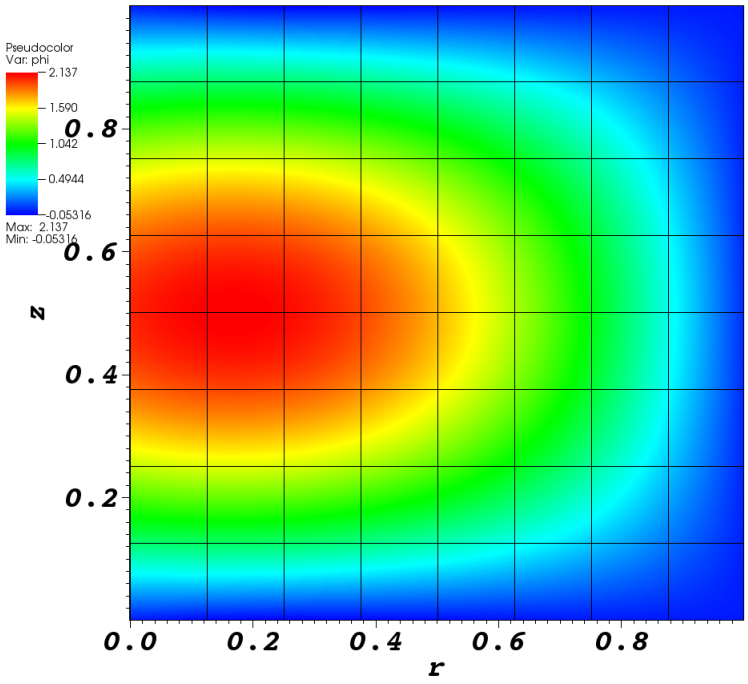


Figure 6: MMS solution for Equation 55.

5.3 Lumping

5.4 Diffusion Synthetic Acceleration

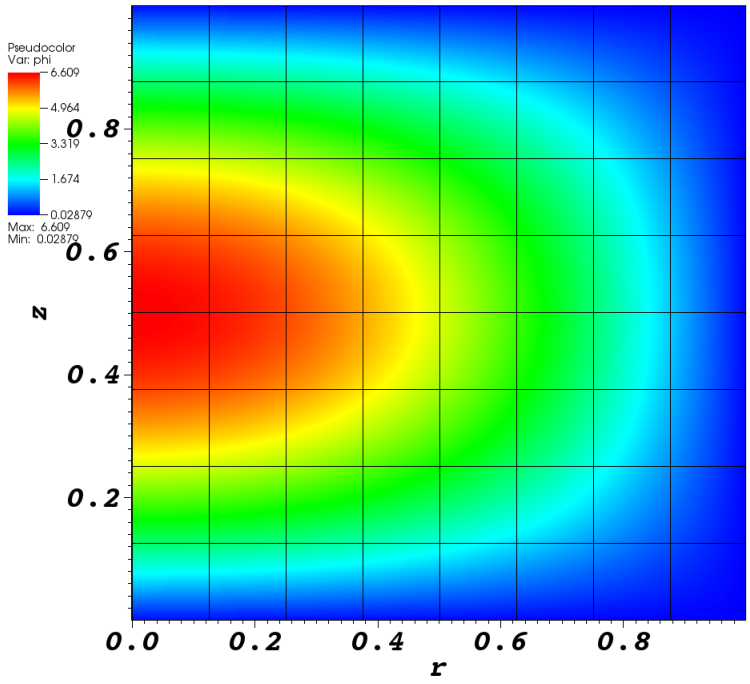


Figure 7: MMS solution for Equation 56.

5.5 Symmetry Preservation

We want R-Z geometry to solve and preserve 1-dimensional spherical solutions. The transport equation in spherical geometry is

$$\mathbf{\Omega}\cdot\mathbf{\nabla}\psi = \frac{\partial\psi}{\partial r}\mathbf{\Omega}\cdot\mathbf{\nabla}r + \frac{\partial\psi}{\partial\mu}\mathbf{\Omega}\cdot\mathbf{\nabla}\mu \tag{57}$$

$$= \mu\frac{\partial\psi}{\partial r} + \frac{(1-\mu^2)}{r}\frac{\partial\psi}{\partial\mu} \tag{58}$$

$$= \frac{\mu}{r^2}\frac{\partial}{\partial r}(r^2\psi) + \frac{\partial}{\partial\mu}\frac{[(1-\mu^2)\psi]}{r} \tag{59}$$

$$\mu\frac{\partial}{\partial r}(r^2\psi) + r\frac{\partial}{\partial\mu}((1-\mu^2)\psi) + r^2\sigma_t\psi = \frac{1}{4\pi}\sigma_s\phi + \frac{1}{4\pi}S_0. \tag{60}$$

In 1-dimensional geometry, this simplifies to

$$\mu\frac{\partial\psi}{\partial r} + \sigma_t\psi = \frac{1}{4\pi}\sigma_s\phi + \frac{1}{4\pi}S_0. \tag{61}$$

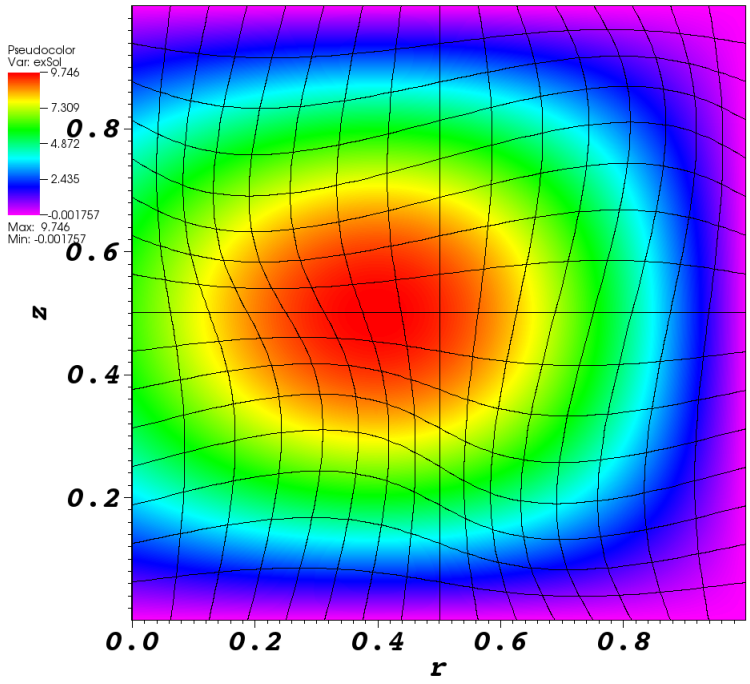


Figure 8: MMS solution to Equation ??.

We can solve this for ψ.

$$\frac{\partial \psi(r)}{\partial r} e^{\sigma_t r/\mu} + \frac{\sigma_t}{\mu} \psi(r) e^{\sigma_t r/\mu} = \frac{1}{4\pi\mu} \left(\sigma_s \phi + S_0\right) e^{\sigma_t r/\mu} \tag{62}$$

$$\frac{\partial}{\partial r} \left(\psi(r) e^{\sigma_t r/\mu}\right) = \frac{1}{4\pi\mu} \left(\sigma_s \phi + S_0\right) e^{\sigma_t r/\mu} \tag{63}$$

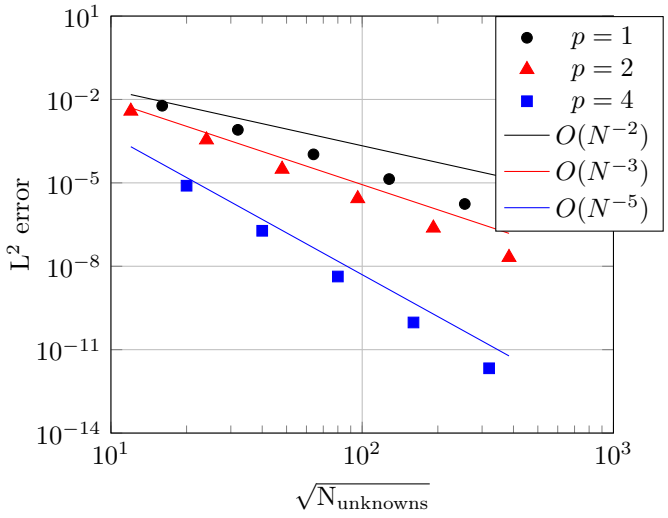
$$\int \frac{\partial}{\partial r'} \left(\psi(r') e^{\sigma_t r'/\mu}\right) dr' = \frac{1}{4\pi\mu} \int \left(\sigma_s \phi(r') + S_0\right) e^{\sigma_t r'/\mu} dr' \tag{64}$$

$$\psi(r) e^{\sigma_t r/\mu} - c = \frac{1}{4\pi\mu} \int \left(\sigma_s \phi(r') + S_0\right) e^{\sigma_t r'/\mu} dr' \tag{65}$$

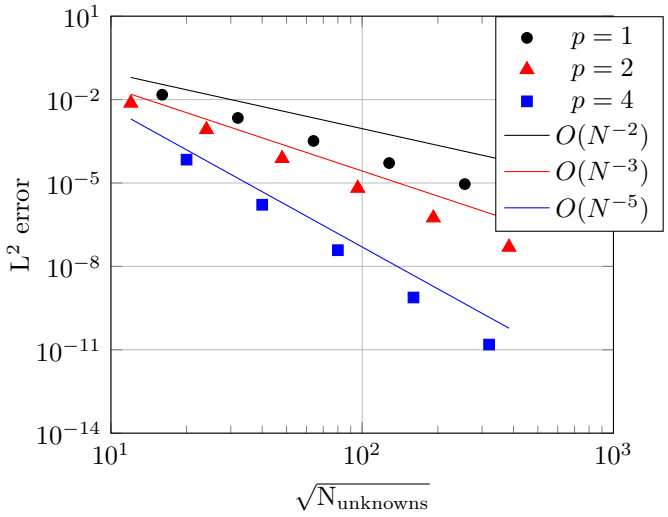
$$\psi(r) = e^{-\sigma_t r/\mu} \frac{1}{4\pi\mu} \int \left(\sigma_s \phi(r') + S_0\right) e^{\sigma_t r'/\mu} dr' + c e^{-\sigma_t r/\mu} \tag{66}$$

$$\psi(1) = 1 = e^{-\sigma_t/\mu} \frac{1}{4\pi\mu} \int \left(\sigma_s \phi(r') + S_0\right) e^{\sigma_t r'/\mu} dr' + c e^{-\sigma_t/\mu} \tag{67}$$

$$c = e^{\sigma_t/\mu} - \frac{1}{4\pi\mu} \int \left(\sigma_s \phi(r') + S_0\right) e^{\sigma_t r'/\mu} dr' \tag{68}$$



(a) Orthogonal quadrilateral mesh.



(b) 2nd-order curved mesh.

Figure 9: L^2-norm of the errors from the manufactured solution and reference lines, where $N_{\text{unknowns}} = N_{\text{cells}}(p+1)^2$.

$$\psi(r) = e^{-\sigma_t r/\mu}\frac{1}{4\pi\mu}\int (\sigma_s\phi(r') + S_0)\, e^{\sigma_t r'/\mu} dr' + \left[e^{\sigma_t/\mu} - \frac{1}{4\pi\mu}\int (\sigma_s\phi(r') + S_0)\, e^{\sigma_t r'/\mu} dr'\right] e^{-\sigma_t r/\mu} \quad (69)$$

$$\psi(r) = e^{-\sigma_t r/\mu}\frac{1}{4\pi\mu}\int (\sigma_s\phi(r') + S_0)\, e^{\sigma_t r'/\mu} dr' + e^{(1-r)\sigma_t/\mu} - e^{-\sigma_t r/\mu}\frac{1}{4\pi\mu}\int (\sigma_s\phi(r') + S_0)\, e^{\sigma_t r'/\mu} dr' \quad (70)$$

$$\psi(r) = e^{(1-r)\sigma_t/\mu} \tag{71}$$

We use MMS to solve a 1-D spherical problem using the R-Z geometry spatial discretization. The manufactured solution is

$$\psi_{\text{MMS}}(\rho) = \sin(\pi\rho) + 2 - \rho, \tag{72}$$

where $\rho = \sqrt{r^2 + z^2}$ is the distance from the origin (i.e. the spherical radius). Plugging this into the R-Z transport equation,

$$\begin{aligned}\frac{\mu}{r}\frac{\partial}{\partial r}\left[r\left(\sin(\pi\rho) + 2 - \rho\right)\right] + \frac{\eta}{r}\frac{\partial}{\partial r}\left(\sin(\pi\rho) + 2 - \rho\right) + \xi\frac{\partial}{\partial z}\left(\sin(\pi\rho) + 2 - \rho\right) \\ - \frac{1}{r}\frac{\partial}{\partial \omega}\left[\eta\left(\sin(\pi\rho) + 2 - \rho\right)\right] + \sigma_t\left(\sin(\pi\rho) + 2 - \rho\right) \\ = \frac{1}{2\pi}\sigma_s\phi_{\text{MMS}} + \frac{1}{2\pi}S_0. \end{aligned} \tag{73}$$

Integrating ψ_{MMS} over all directions reveals ϕ_{MMS},

$$\begin{aligned}\frac{\mu}{r}\frac{\partial}{\partial r}\left[r\left(\sin(\pi\rho) + 2 - \rho\right)\right] + \frac{\eta}{r}\frac{\partial}{\partial r}\left(\sin(\pi\rho) + 2 - \rho\right) + \xi\frac{\partial}{\partial z}\left(\sin(\pi\rho) + 2 - \rho\right) \\ - \frac{1}{r}\frac{\partial}{\partial \omega}\left[\eta\left(\sin(\pi\rho) + 2 - \rho\right)\right] + \sigma_t\left(\sin(\pi\rho) + 2 - \rho\right) \\ = \sigma_s\left(\sin(\pi\rho) + 2 - \rho\right) + \frac{1}{2\pi}S_0. \end{aligned} \tag{74}$$

We perform some simplifications,

$$\begin{aligned}\frac{\mu}{r}\frac{\partial}{\partial r}\left[r\left(\sin(\pi\rho) + 2 - \rho\right)\right] + \frac{\eta}{r}\frac{\partial}{\partial r}\left(\sin(\pi\rho) + 2 - \rho\right) + \xi\frac{\partial}{\partial z}\left(\sin(\pi\rho) + 2 - \rho\right) \\ - \frac{1}{r}\frac{\partial}{\partial \omega}\left[\eta\left(\sin(\pi\rho) + 2 - \rho\right)\right] + \sigma_t\left(\sin(\pi\rho) + 2 - \rho\right) \\ = \sigma_s\left(\sin(\pi\rho) + 2 - \rho\right) + \frac{1}{2\pi}S_0. \end{aligned} \tag{75}$$

Because of the angular derivative in the streaming term, we reduce the influence of the direction dependence as much as possible by using higher order level-symmetric angular quadrature.

We evaluate the relative asymmetry by calculating the averages of all nodes at each ρ value and

$$\phi_{\text{sym}}(\rho,\theta) = \frac{\phi_{\text{code}}(\rho,\theta) - \phi_{\text{avg}}(\rho)}{\phi_{\text{avg}}(\rho)}, \tag{76}$$

where

$$\phi_{\text{avg}}(\rho) = \frac{1}{N_{\text{nodes}}(\rho)} \sum_{i=1}^{N_{\text{nodes}}(\rho)} \phi(\rho,\theta_i) \tag{77}$$

is the average scalar flux at all nodes at the same spherical radius ρ.

5.6 Other

5.7 Material Discontinuity Stress Test

We adapted this problem from Palmer [**?**] and solved it without DSA in Woods et al. [**?**]. There are five different material regions described in Table 2 and Figure 7.

Table 2: Material discontinuity stress test with MIP DSA material properties.

Material Region	σ_t cm^{-1}	σ_s cm^{-1}	S_0 cm^{-2} s^{-1}
Source	1.0	1.0	1.0
Very thin absorber	0.0001	0.0	0.0
Thick absorber	10.0	0.0	0.0
Very thick absorber	100.0	0.0	0.0
Very thick scatterer	1000.0	1000.0	0.0

This problem has opacities that range several orders of magnitude, resulting in strong material discontinuities. We also introduce anisotropic incident intensities into the scattering region by preferentially attenuating intensities that are not perpendicular to the thick absorber. We expect some degradation in the DSA in problems with

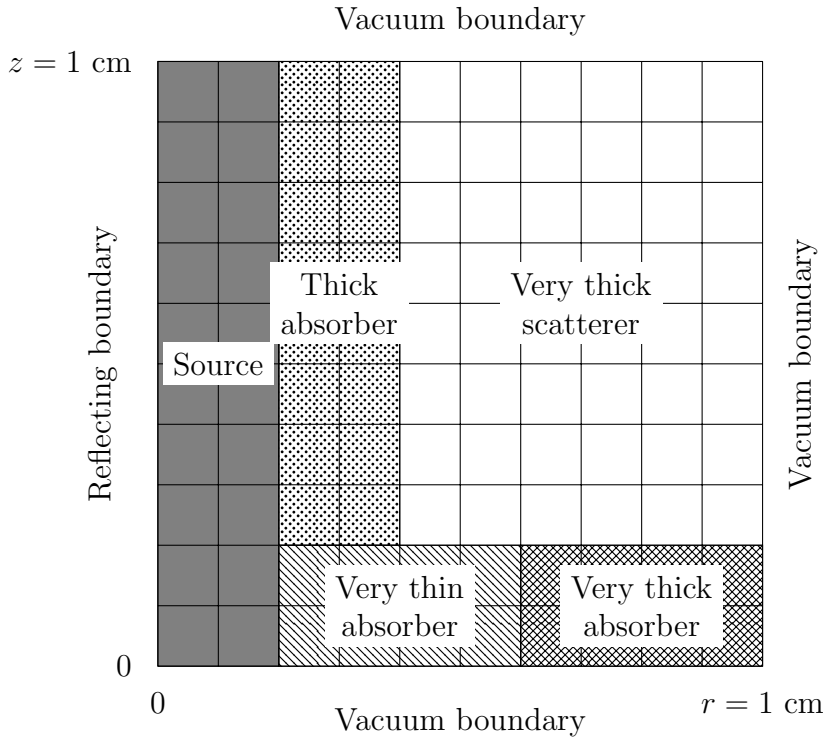
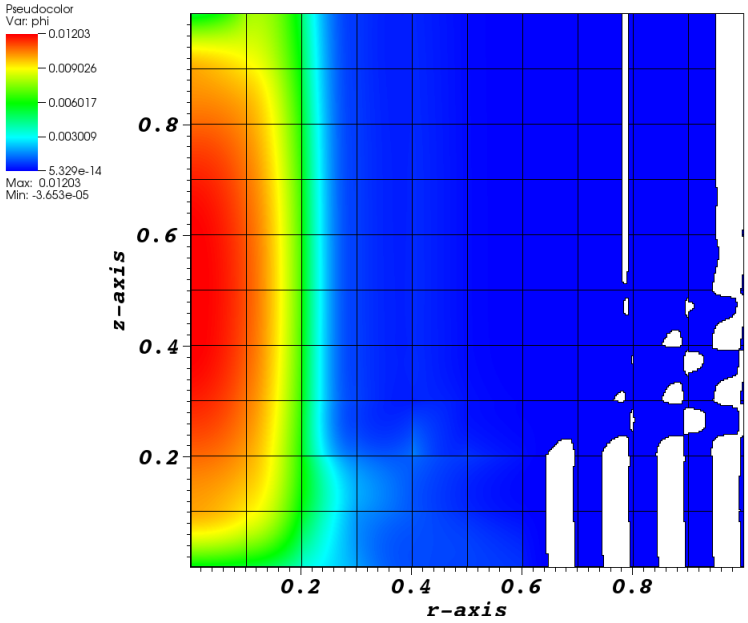
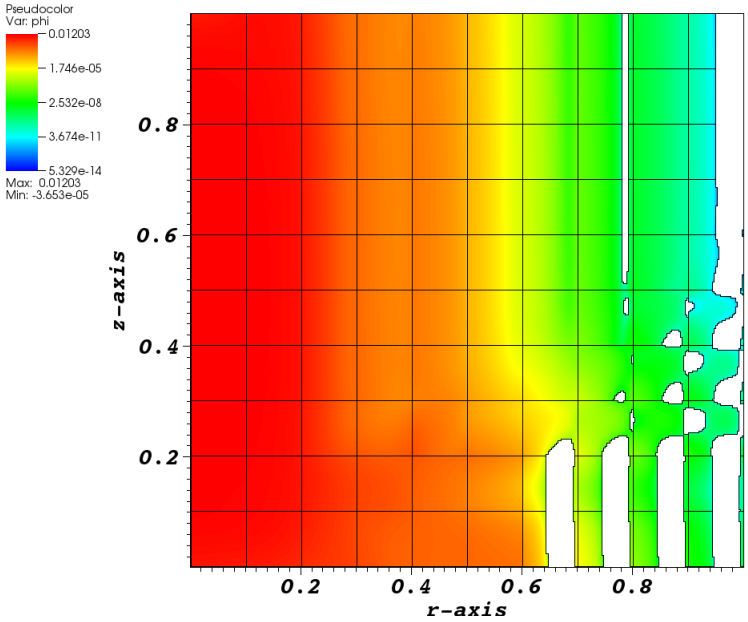


Figure 10: Material discontinuity stress test with MIP DSA problem geometry; materials defined in Table 2.

strong material discontinuities [?]. We also expect boundary layers to form from the anisotropic incident intensities [?]. The solution is shown in Figure 8.



(a) Scalar flux.



(b) Log of scalar flux.

Figure 11: Solution to multi-material stress test. White regions indicate negative scalar fluxes. This was solved without DSA and was only allowed 10,000 source iterations to complete.

5.8 Reflecting Boundary Conditions

To incorporate reflecting boundary conditions, we will "guess" the incident angular fluxes, update them with outgoing angular fluxes from the previous iteration, and adapt a convergence criterion for those fluxes. Along the z-axis, the reflection for direction $\mathbf{\Omega} = (\mu, \eta, \xi)$ is $\mathbf{\Omega}_R = (-\mu, \eta, \xi)$.

5.9 Reflecting Boundary Conditions

This may not warrant an entire subsection.

Reflecting boundaries are dependent upon the direction of the outgoing angular flux, $\mathbf{\Omega}_m$. The reflected incident direction is

$$\mathbf{\Omega}^R = \mathbf{\Omega}_m - 2\left(\mathbf{\Omega}_m \cdot \hat{n}\right)\hat{n} \tag{78}$$

where $\mathbf{\Omega}_m$ is the outgoing direction and $\hat{n}$ is the unit normal vector on the boundary (pointing outward). We apply a Dirichlet boundary condition for the angular flux in the reflected direction,

$$\psi^b(\boldsymbol{r}, \mathbf{\Omega}^R) = \psi_m \tag{79}$$

6 Conclusions

6.1 FutureWork

Future work beyond the scope of these research objectives could include methods for solving the system of equations. Currently we simultaneously solve for every degree of freedom in the problem. This limits the overall number of unknowns that we can accommodate because they all get stored in memory. However, if we solved individual mesh cells and systematically "swept through the mesh" (solve the sparse system of equations for each quadrature direction, in parallel or sequentially), we eliminate this limitation. This allows for cycles (discussed in Section **??**) in the mesh, which require careful handling.

Negative energy densities, observed in some of the test problems above, may contribute to negative mass densities in the equations of state of multiphysics problems. This must be addressed. Mentioned previously are lumping the matrices that constitute the bilinear form or perform a negative energy density fix up. The latter results in a nonlinear system of equations, which may be satisfactory given the TRT equations are already nonlinear. Oscillations could be reduced by using lower order elements in susceptible regions. Investigating an adaptive element could prove beneficial.

Coupling the TRT equations with the hydrodynamics equations is a logical progression toward modeling a realistic problem. Understanding the coupling between equations may be challenging given the variety of physical phenomena in these problems. The increase in the number of degrees of freedom upon this coupling begs for an increase in efficiency. An investigation into increased parallelization is warranted along with solution methods such as sweeping through the mesh.

Mesh sweeping, breaking cycles.

References

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

[2] John Castor. *Radiation Hydrodynamics*. Cambridge, 2007.

[3] R. Paul Drake. High-energy-density physics. *Physics Today*, 63(3):8–9, June 2010.

[4] Veselin A. Dobrev, Tzanio V. Kolev, and Robert N. Rieben. High-order curvilinear finite element methods for Lagrangian hydrodynamics. *SIAM Journal of Scientific Computing*, 34(5):B606 – B641, 2012.

[5] G. Scovazzi, E. Love, and M. J. Shashkov. Multi-scale Lagrangian shock hydrodynamics on Q1/P0 finite elements: Theoretical framework and two-dimensional computations. *Computer Methods in Applied Mechanics and Engineering*, 197:1056–1079, 2008.

[6] V. A. Dobrev, T. E. Ellis, Tz. V. Kolev, and R. N. Rieben. Curvilinear finite elements for lagrangian hydrodynamics. *International Journal for Numerical Methods in Fluids*, 2010.

[7] MFEM: Modular finite element methods. `mfem.org`, 2015.

[8] Thomas A. Brunner. Forms of approximate radiation transport. Technical report, Sandia National Laboratories, 2002.

[9] K. D. Lathrop and B. G. Carlson. Discrete ordinates angular quadrature of the neutron transport equation. Technical report, Los Alamos Scientific Laboratory of the University of California, 1965.

[10] Marvin L. Adams, Todd A. Wareing, and Wallace F. Walters. Characteristic methods in thick diffusive problems. *Nuclear Science and Engineering*, 130:18–46, 1998.

[11] Edward W. Larsen and J. E. Morel. Asymptotic solutions of numerical transport problems in optically thick, diffusive regimes II. *Journal of Computational Physics*, 83:212–236, 1989.

[12] F. Malvagi and G. C. Pomraning. Initial and boundary conditions for diffusive linear transport problems. *Journal of Mathematical Physics*, 32(3):805–820, 1991.

[13] Edward W. Larsen. Unconditionally stable diffusion-synthetic acceleration methods for the slab geometry discrete ordinates equations. part I: Theory. *Nuclear Science and Engineering*, 82(1):47–63, 1982.

[14] Yaqi Wang and Jean C. Ragusa. Diffusion synthetic acceleration for high-order discontinuous finite element S_N transport schemes and application to locally refined unstructured meshes. *Nuclear Science and Engineering*, 166:145–166, 2010.

[15] Bruno Turcksin and Jean C. Ragusa. Discontinuous diffusion synthetic acceleration for S_N transport on 2D arbitrary polygonal meshes. *Journal of Computational Physics*, 274:356–369, 2014.

[16] Guido Kanschat. *Discontinuous Galerkin Methods for Viscous Incompressible Flow*. Deutscher Universit'ats-Verlag, 2007.

[17] J. E. Morel and G. R. Montry. Analysis and elimination of the discrete-ordinates flux dip. *Transport Theory and Statistical Physics*, 13(5):615–633, 1984.

[18] J. S. Warsa and A. K. Prinja. Differential quadrature approximation of the angular derivative terms in the curvilinear-coordinates S_N equations. *Transactions of the American Nuclear Society*, 111:689–692, 2014.

[19] Timothy A. Davis. *Direct Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics, 2006.

[20] Timothy A. Davis. Algorithm 832: Umfpack v4.3 — an unsymmetric-pattern multifrontal method. *ACM Transactions on Mathematical Software*, 30(2):196–199, June 2004.

[21] Hank Childs, Eric Brugger, Brad Whitlock, Jeremy Meredith, Sean Ahern, David Pugmire, Kathleen Biagas, Mark Miller, Cyrus Harrison, Gunther H. Weber, Hari Krishnan, Thomas Fogal, Allen Sanderson, Christoph Garth, E. Wes Bethel, David Camp, Oliver Rübel, Marc Durant, Jean M. Favre, and Paulr Navrátil. Visit: An end-user tool for visualizing and analyzing very large data. In *High Performance Visualization–Enabling Extreme-Scale Scientific Insight*, pages 357–372. Oct 2012.

A Implementation in MFEM

The open source finite element library Modular Finite Elements Method (MFEM[3]) [7] was used to create the system of linear equations to be solved by a linear algebra solver. The user chooses various parameters to create the system of equations and passes them into MFEM as arguments (i.e. the order of finite elements, the mesh, the number of times to refine the mesh, the order of the mesh, any mesh transformations, the linear algebra solver method, the source iteration convergence criteria, the maximum number of source iterations to perform, the initial guess for the scalar flux, and, in diffusion limit problems, the scaling factor to be applied).

MFEM creates the matrices and a linear solver computes the angular flux. This research utilizes the serial version of MFEM (as opposed to the parallel version where the spatial domain is solved in parallel) and the direct solver UMFPack[4] [19, 20] to solve the equations using a LU decomposition. It is common for transport solvers to solve the local system of equations for an individual spatial cell and sweep through the problem domain, propagating information from one cell to the next. Instead, we use MFEM to create the system of equations for the entire problem domain and solve for all of the unknowns in all cells simultaneously. This is more computationally intensive because all of the degrees of freedom in the entire problem domain must be solved for simultaneously. However, mesh zone complications, such as cycles, may be present in the mesh. We avoid having to "break the cycles" by solving for the entire problem simultaneously.

[3] `mfem.org`

[4] `http://faculty.cse.tamu.edu/davis/suitesparse.html`

A.1 Transport Operators

Shown in Table 4 are the functions within MFEM that integrate and assemble the various components of the transport equation (Equation **??**) to the linear algebraic system. The functions are displayed along with the general form of their equation and their translation to the applicable component of the transport equation. The last two entries of Table 4 are the interior boundaries using the upstream values (no. 5) and the problem boundary (no. 6). Several of the MFEM equations have coefficients, α and β, that are required input. For item number 1, using $\alpha = 1$ sets the MFEM equation equal to the discretized equation. Similarly, for item number 5 using $\alpha = -1$ and $\beta = 1/2$,

$$\alpha \int_{\partial\mathbb{V}} \mathbf{\Omega} \cdot (-\hat{n}) \{\psi\} w + \beta \int_{\partial\mathbb{V}} |\mathbf{\Omega} \cdot (-\hat{n})| [\![\psi]\!] w = \int_{\partial\mathbb{V}_k} (\mathbf{\Omega} \cdot \hat{n}) \psi_k \; w_{k,i} \tag{80}$$

where $\{\psi\} = 1/2 (\psi_u + \psi_k)$ and $[\![\psi]\!] = \psi_u - \psi_k$, where ψ_u is the upwind angular flux and ψ_k is the angular flux in cell k.

$$-1 \int_{\partial\mathbb{V}} [\mathbf{\Omega} \cdot (-\hat{n})] \left[\frac{1}{2} (\psi_u + \psi_k)\right] w + \frac{1}{2} \int_{\partial\mathbb{V}} |\mathbf{\Omega} \cdot (-\hat{n})| (\psi_u - \psi_k) w = \int_{\partial\mathbb{V}_k} (\mathbf{\Omega} \cdot \hat{n}) \psi_k \; w_{k,i} \tag{81}$$

$$\frac{1}{2} \int_{\partial\mathbb{V}} (\mathbf{\Omega} \cdot \hat{n}) (\psi_u + \psi_k) w + \frac{1}{2} \int_{\partial\mathbb{V}} |\mathbf{\Omega} \cdot (-\hat{n})| (\psi_u - \psi_k) w = \int_{\partial\mathbb{V}_k} (\mathbf{\Omega} \cdot \hat{n}) \psi_k \; w_{k,i} \tag{82}$$

For $\mathbf{\Omega} \cdot \hat{n} < 0$ (incident to cell k),

$$\frac{1}{2} \int_{\partial\mathbb{V}} (\mathbf{\Omega} \cdot \hat{n}) (\psi_u + \psi_k) w - \frac{1}{2} \int_{\partial\mathbb{V}} (\mathbf{\Omega} \cdot \hat{n}) (\psi_u - \psi_k) w = \int_{\partial\mathbb{V}_k} (\mathbf{\Omega} \cdot \hat{n}) \psi_k \; w_{k,i} \tag{83}$$

$$\frac{1}{2} \int_{\partial\mathbb{V}} (\mathbf{\Omega} \cdot \hat{n}) \psi_k \; w + \frac{1}{2} \int_{\partial\mathbb{V}} (\mathbf{\Omega} \cdot \hat{n}) \psi_k \; w = \int_{\partial\mathbb{V}_k} (\mathbf{\Omega} \cdot \hat{n}) \psi_k \; w_{k,i} \tag{84}$$

The normal vector $\hat{n}$ in MFEM is outward of the upwind mesh surface so a negative was applied to the normal vector to convert it to be the outward normal of the surface of cell k like it has been previously defined in this thesis. Similarly, for item number 6, $\alpha = -1$ and $\beta = -1/2$,

$$\frac{\alpha}{2}\int_{\partial\mathbb{V}} \psi_{\text{inc}}\ \mathbf{\Omega}\cdot(-\hat{n})\ \ w - \beta\int_{\partial\mathbb{V}} \psi_{\text{inc}}\,|\mathbf{\Omega}\cdot(-\hat{n})|\,w = \int_{\partial\mathbb{V}_k} (\mathbf{\Omega}\cdot\hat{n})\,\psi_{\text{inc},k}\ w_{k,i} \tag{85}$$

$$-\frac{1}{2}\int_{\partial\mathbb{V}} \psi_{\text{inc}}\ \mathbf{\Omega}\cdot(-\hat{n})\ \ w + \frac{1}{2}\int_{\partial\mathbb{V}} \psi_{\text{inc}}\,|\mathbf{\Omega}\cdot(-\hat{n})|\,w = \int_{\partial\mathbb{V}_k} (\mathbf{\Omega}\cdot\hat{n})\,\psi_{\text{inc},k}\ w_{k,i} \tag{86}$$

$$\frac{1}{2}\int_{\partial\mathbb{V}} \psi_{\text{inc}}\,(\mathbf{\Omega}\cdot\hat{n})\,w + \frac{1}{2}\int_{\partial\mathbb{V}} \psi_{\text{inc}}\,[\mathbf{\Omega}\cdot(-\hat{n})]\,w = \int_{\partial\mathbb{V}_k} (\mathbf{\Omega}\cdot\hat{n})\,\psi_{\text{inc},k}\ w_{k,i} \tag{87}$$

$$\frac{1}{2}\int_{\partial\mathbb{V}} \psi_{\text{inc}}\,(\mathbf{\Omega}\cdot\hat{n})\,w + \frac{1}{2}\int_{\partial\mathbb{V}} \psi_{\text{inc}}\,(\mathbf{\Omega}\cdot\hat{n})\,w = \int_{\partial\mathbb{V}_k} (\mathbf{\Omega}\cdot\hat{n})\,\psi_{\text{inc},k}\ w_{k,i} \tag{88}$$

MFEM automatically determines the degree of numerical integration to integrate each of the integrals of Table 4. These default integration orders are shown in Table 5. It was discovered that integrating all of the terms consistently was important for numeric conservation. For simplicity, each of the integration orders were set to the largest of the default integration orders. Table 5 shows that the integration order is the same for all of the integrators except DomainLFIntegrator, which is the largest integration order only if $p = 0$ (piecewise constant). While the results presented in this thesis do not consider the circumstance of piecewise constant finite elements, this integration order was included in the code for future use.

MFEM is equipped to visualize data using various tools requiring additional user input. The images presented in this thesis were produced with VisIt, an open source visualization analysis tool [21].

no.	Discretized Equation	MFEM Equation	MFEM Integrator Function
1	$(\mathbf{\Omega} \cdot \boldsymbol{\nabla} \psi_j, v_i)_{\mathbb{D}_k}$	$(\alpha\ \mathbf{\Omega} \cdot \boldsymbol{\nabla} \psi, v)_{\mathbb{D}_k}$	ConvectionIntegrator($\mathbf{\Omega}, \alpha$)
2	$(\sigma_t\ \psi_j, v_i)_{\mathbb{D}_k}$	$(\sigma_t\ \psi, v)_{\mathbb{D}_k}$	MassIntegrator(σ_t)
3	$(\sigma_s\ \phi, v_i)_{\mathbb{D}_k}$	$(\varphi, v)_{\mathbb{D}_k}$	DomainLFIntegrator(φ)
4	$(S_0, v_i)_{\mathbb{D}_k}$	$(S_0, v)_{\mathbb{D}_k}$	DomainLFIntegrator(S_0)
5	$(\mathbf{\Omega} \cdot \hat{n}\ \psi_j, v_i)_{\partial\mathbb{D}_k}$	$\alpha\,(\mathbf{\Omega} \cdot \hat{n}\ \psi, v)_{\partial\mathbb{D}_k}$ $+\beta\,(\lvert\mathbf{\Omega} \cdot \hat{n}\rvert\ \psi, v)_{\partial\mathbb{D}_k}$	DGTraceIntegrator($\mathbf{\Omega}, \alpha, \beta$)
6	$(\mathbf{\Omega} \cdot \hat{n}\ \psi_{\text{inc}}, v_i)_{\partial\mathbb{D}_k}$	$\frac{\alpha}{2}\,(\psi_{inc}\ \mathbf{\Omega} \cdot \hat{n}, v)_{\partial\mathbb{D}_k}$ $-\beta\,(\psi_{inc}\,\lvert\mathbf{\Omega} \cdot \hat{n}\rvert\,, v)_{\partial\mathbb{D}_k}$	BoundaryFlowIntegrator($\psi_{inc}, \mathbf{\Omega}, \alpha, \beta$)

Table 3: MFEM PDE function calls where the arguments have been dropped (see Equations **??** and **??** for these details).

no.	Discretized Equation	MFEM Equation	MFEM Integrator Function
	$(r\ \mathbf{\Omega}\cdot\boldsymbol{\nabla}\psi_j, v_i)_{\mathbb{D}_k}$	$(r\ \alpha\ \mathbf{\Omega}\cdot\boldsymbol{\nabla}\psi, v)_{\mathbb{D}_k}$	RZConvectionIntegrator$(\mathbf{\Omega}, \alpha)$ †
	$(r\ \mathbf{\Omega}\cdot\hat{n}\ \psi_j, v_i)_{\partial\mathbb{D}_k}$	$\alpha\,(r\ \mathbf{\Omega}\cdot\hat{n}\ \psi, v)_{\partial\mathbb{D}_k}$ $+\beta\,(r\ \lvert\mathbf{\Omega}\cdot\hat{n}\rvert\,\psi, v)_{\partial\mathbb{D}_k}$	RZDGTraceIntegrator$(\mathbf{\Omega}, \alpha, \beta)$ †
	$(\mu_{n,m}\psi_j, v_i)_{\mathbb{D}_k}$		MassIntegrator$(\mu_{n,m})$
	$\left(\frac{\alpha_{m+1/2}n}{\tau_{n,m}w_{n,m}}\psi_j, v_i\right)_{\mathbb{D}_k}$		MassIntegrator$(\frac{\alpha_{m+1/2}n}{\tau_{n,m}w_{n,m}})$
	$(r\ \sigma_t\ \psi_j, v_i)_{\mathbb{D}_k}$	$(r\ \sigma_t\ \psi, v)_{\mathbb{D}_k}$	RZMassIntegrator(σ_t) †
	$\frac{1}{4\pi}\,(r\ \sigma_s\ \phi, v_i)_{\mathbb{D}_k}$	$(r\ \varphi, v)_{\mathbb{D}_k}$	RZDomainLFIntegrator$\left(\frac{1}{4\pi}\sigma_s\ \phi\right)$ †
	$\frac{1}{4\pi}\,(r\ S_0, v_i)_{\mathbb{D}_k}$	$(r\ S_0, v)_{\mathbb{D}_k}$	RZDomainLFIntegrator$\left(\frac{S_0}{4\pi}\right)$ †
	$(r\ \mathbf{\Omega}\cdot\hat{n}\ \psi_{\text{inc}}, v_i)_{\partial\mathbb{D}_k}$	$\frac{\alpha}{2}\,(r\ \psi_{inc}\ \mathbf{\Omega}\cdot\hat{n}, v)_{\partial\mathbb{D}_k}$ $-\beta\,(r\ \psi_{inc}\,\lvert\mathbf{\Omega}\cdot\hat{n}\rvert\,, v)_{\partial\mathbb{D}_k}$	RZBoundaryFlowIntegrator$(\psi_{inc}, \mathbf{\Omega}, \alpha, \beta)$ †

† Modified MFEM operator

Table 4: MFEM PDE function calls where the arguments have been dropped (see Equations **??** and **??** for these details).

Table 5: MFEM default integration orders for transport operators. The notation for the finite element order is p, mesh order is m, and problem dimension is d.

MFEM Integrator	**Default Integration Order**
DGTraceIntegrator	$m \cdot d + 2 \cdot p - 1$
ConvectionIntegrator	$m \cdot d + 2 \cdot p - 1$
MassIntegrator	$m \cdot d + 2 \cdot p - 1$
DomainLFIntegrator	$2 \cdot m$
BoundaryFlowIntegrator	$m \cdot d + 2 \cdot p - 1$

A.1.1 MIP DSA Operators

There are some specific function calls to MFEM for the diffusion equation that are listed in Table 6. Items 3 and 4 have a σ_D value that controls the DG method to be used, where $\sigma_D = -1$ is for the symmetric interior penalty method. Item 6 is for the Robin boundary condition described by Equation 24. Item 4 is the function that will need to be modified to adapt Methods 1 and 2 (this function is not used for Method 3).

no.	FEM Equations 15 & 16	MFEM Equation	User Input
1	$(\sigma_a\phi, w)_{\mathbb{V}}$	$(\sigma_a\phi, w)_{\mathbb{V}}$	MassIntegrator(σ_a)
2	$(D\boldsymbol{\nabla}\phi, \boldsymbol{\nabla}w)_{\mathbb{V}}$	$(D\boldsymbol{\nabla}\phi, \boldsymbol{\nabla}w)_{\mathbb{V}}$	DiffusionIntegrator(D)
3	$(\{\!\{D\partial_n\phi\}\!\}, [\![w]\!])_{\partial\mathbb{V}^i}$ $+([\![\phi]\!], \{\!\{D\partial_n w\}\!\})_{\partial\mathbb{V}^i}$ $+(\kappa_e[\![\phi]\!], [\![w]\!])_{\partial\mathbb{V}^i}$	$(\{\!\{D\boldsymbol{\nabla}\phi\cdot\hat{n}\}\!\}, [\![w]\!])_{\partial\mathbb{V}^i}$ $+\sigma_D([\![\phi]\!], \{\!\{D\boldsymbol{\nabla}w\cdot\hat{n}\}\!\})_{\partial\mathbb{V}^i}$ $+\kappa\left(\left\{\!\!\left\{\frac{D}{h_\perp}\right\}\!\!\right\}[\![\phi]\!], [\![w]\!]\right)_{\partial\mathbb{V}^i}$	DGDiffusionIntegrator(D, σ_D, κ)
4	$(\{\!\{D\partial_n\phi\}\!\}, [\![w]\!])_{\partial\mathbb{V}^d}$ $-\frac{1}{2}([\![\phi]\!], \{\!\{D\partial_n w\}\!\})_{\partial\mathbb{V}^d}$ $-\frac{1}{2}(\kappa_e[\![\phi]\!], [\![w]\!])_{\partial\mathbb{V}^d}$	$(\{\!\{D\boldsymbol{\nabla}\phi\cdot\hat{n}\}\!\}, [\![w]\!])_{\partial\mathbb{V}^d}$ $+\sigma_D([\![\phi]\!], \{\!\{D\boldsymbol{\nabla}w\cdot\hat{n}\}\!\})_{\partial\mathbb{V}^d}$ $+\kappa\left(\left\{\!\!\left\{\frac{D}{h_\perp}\right\}\!\!\right\}[\![\phi]\!], [\![w]\!]\right)_{\partial\mathbb{V}^d}$	DGDiffusionIntegrator(D, σ_D, κ)
5	$(Q_0, w)_{\mathbb{V}}$	$(Q_0, w)_{\mathbb{V}}$	DomainLFIntegrator(Q_0)
6	$\left(\frac{1}{2}\phi, w\right)_{\partial\mathbb{V}^d}$	$\left(\frac{1}{2}\phi, w\right)_{\partial\mathbb{V}^d}$	BoundaryMassIntegrator$\left(\frac{1}{2}\right)$

† $\partial\mathbb{V}^i$ denotes an internal edge
‡ $\partial\mathbb{V}^d$ denotes a boundary edge

Table 6: MFEM diffusion equation function calls.

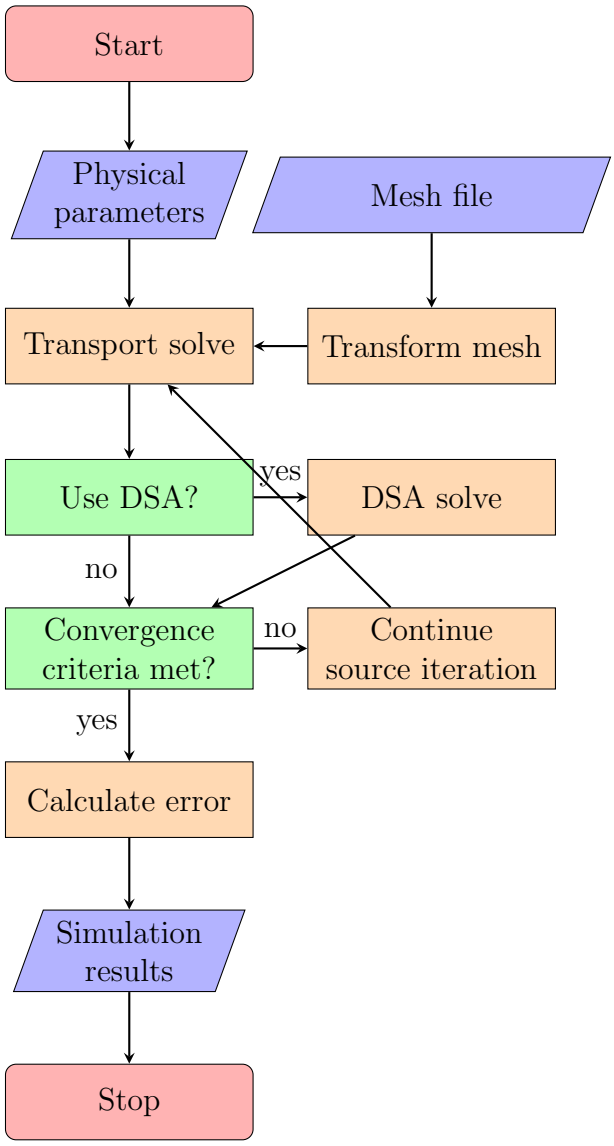


Figure 12: Flow diagram for solution process.

B Mesh Examples

C Mathemtica Scipts

C.1 Mass Matrix Entries for RZ BLD Gauss-Legendre on Quadrilaterals

```
(*Mass matrix entry for RZ BLD Gauss-Legendre on quadrilaterals*)
\
Clear[r0, r1, r2, r3, z0, z1, z2, z3, mr0, mr1, mr2, mr3, mz0, mz1, \
mz2, mz3]
(*Define physical integration point coordinates to compare to MFEM \
output:r0=(1+(-Sqrt[1/3]))/(1-(-1))*(0.25-0);
r1=(1+(Sqrt[1/3]))/(1-(-1))*(0.25-0);
z0=(1+(-Sqrt[1/3]))/(1-(-1))*(0.25-0);
z3=(1+(Sqrt[1/3]))/(1-(-1))*(0.25-0);*)
(*These make the mesh \
rectangular:*)
z2 := z3
z1 := z0
r2 := r1
r3 := r0
(*Quadrilateral transformation back to reference square-more general \
than rectangle transofrmation-but NOT curved mesh*)

r[rho_, kappa_] := (r1 - r0) rho + (r3 - r0) kappa + (r2 + r0 - r1 -
     r3) rho kappa + r0
z[rho_, kappa_] := (z1 - z0) rho + (z3 - z0) kappa + (z2 + z0 - z1 -
     z3) rho kappa + z0
```

```
(*Mesh nodes are needed to conserve cell volume.Cannot define mr0=0 \
nor mz0=0 here.Perform integration and then set the values.*)
\
(*Comment out to get the general mesh element mr1=0.25;
mz3=0.25;*)
mr2 := mr1
mr3 := mr0
mz1 := mz0
mz2 := mz3
mr[rho_, kappa_] := (mr1 - mr0) rho + (mr3 - mr0) kappa + (mr2 + mr0 -
      mr1 - mr3) rho kappa + mr0
mz[rho_, kappa_] := (mz1 - mz0) rho + (mz3 - mz0) kappa + (mz2 + mz0 -
      mz1 - mz3) rho kappa + mz0
a = {mr[rho, kappa], mz[rho, kappa]};
b = {rho, kappa};
(*Determinant of the Jacobian-for determining area of the physical \
element.*)
j = Det[D[a, {b}]];
(*Basis functions.The coefficients are from MATLAB script \
"GaussLegendreQuadrilateral.m".*)

BLD1[rho_, kappa_] :=
 3*rho*kappa - (3 + Sqrt[3])/2*rho - (3 + Sqrt[3])/2*
   kappa + (1 + Sqrt[3]/2)
BLD2[rho_, kappa_] := -3*rho*kappa + (3 + Sqrt[3])/2*rho +
  3/(3 + Sqrt[3])*kappa - 0.5
```

```
BLD3[rho_, kappa_] :=
 3*rho*kappa - 3/(3 + Sqrt[3])*rho -
  3/(3 + Sqrt[3])*kappa + (1 - Sqrt[3]/2)
BLD4[rho_, kappa_] := -3*rho*kappa +
  3/(3 + Sqrt[3])*rho + (3 + Sqrt[3])/2*kappa - 0.5
(*Check for orthogonality.*)
BLD1[r0, z0]*BLD2[r0, z0];
(*The r variable gets transformed back to the reference element*)
M =
 Integrate[
  mr[rho, kappa]*BLD1[rho, kappa]*BLD1[rho, kappa]*j, {rho, 0,
   1}, {kappa, 0, 1}]

mr0 = 0.25;
mz0 = 0.25;
mr1 = 0.5;
mz3 = 0.5;

M
```

C.2 Angular Redistribution Matrix Entries for RZ BLD Gauss-Legendre on Quadrilaterals

```
(*Angular Redistribution matrix for RZ BLD Gauss-Legendre on \
quadrilaterals*)Clear[r0, r1, r2, r3, z0, z1, z2, z3, mr0, mr1, mr2, \
mr3, mz0, mz1, mz2, mz3]
(*Define physical integration point coordinates to compare to MFEM \
```

```
output:r0=(1+(-Sqrt[1/3]))/(1-(-1))*(0.25-0);
r1=(1+(Sqrt[1/3]))/(1-(-1))*(0.25-0);
z0=(1+(-Sqrt[1/3]))/(1-(-1))*(0.25-0);
z3=(1+(Sqrt[1/3]))/(1-(-1))*(0.25-0);*)
(*These make the mesh \
rectangular:*)
z2 := z3
z1 := z0
r2 := r1
r3 := r0
(*Quadrilateral transformation back to reference square-more general \
than rectangle transofrmation-but NOT curved mesh*)

r[rho_, kappa_] := (r1 - r0) rho + (r3 - r0) kappa + (r2 + r0 - r1 -
     r3) rho kappa + r0
z[rho_, kappa_] := (z1 - z0) rho + (z3 - z0) kappa + (z2 + z0 - z1 -
     z3) rho kappa + z0

(*Mesh nodes are needed to conserve cell volume.Cannot define mr0=0 \
nor mz0=0 here.Perform integration and then set the values.*)
\
(*Comment out to get the general mesh element mr1=0.25;
mz3=0.25;*)
mr2 := mr1
mr3 := mr0
mz1 := mz0
mz2 := mz3
```

```
mr[rho_, kappa_] := (mr1 - mr0) rho + (mr3 - mr0) kappa + (mr2 + mr0 -
      mr1 - mr3) rho kappa + mr0
mz[rho_, kappa_] := (mz1 - mz0) rho + (mz3 - mz0) kappa + (mz2 + mz0 -
      mz1 - mz3) rho kappa + mz0
a = {mr[rho, kappa], mz[rho, kappa]};
b = {rho, kappa};
(*Determinant of the Jacobian-for determining area of the physical \
element.*)
j = Det[D[a, {b}]];
(*Basis functions.The coefficients are from MATLAB script \
"GaussLegendreQuadrilateral.m".*)

BLD1[rho_, kappa_] :=
 3*rho*kappa - (3 + Sqrt[3])/2*rho - (3 + Sqrt[3])/2*
   kappa + (1 + Sqrt[3]/2)
BLD2[rho_, kappa_] := -3*rho*kappa + (3 + Sqrt[3])/2*rho +
  3/(3 + Sqrt[3])*kappa - 1/2
BLD3[rho_, kappa_] :=
 3*rho*kappa - 3/(3 + Sqrt[3])*rho -
  3/(3 + Sqrt[3])*kappa + (1 - Sqrt[3]/2)
BLD4[rho_, kappa_] := -3*rho*kappa +
  3/(3 + Sqrt[3])*rho + (3 + Sqrt[3])/2*kappa - 1/2
M = Integrate[
  BLD1[rho, kappa]*BLD1[rho, kappa]*j, {rho, 0, 1}, {kappa, 0, 1}]

mr0 = 0;
mz0 = 0;
```

```
mr1 = 0.25;
mz3 = 0.25;


thisalpha = 0.350021175;
w = 0.523598775598298;
M*thisalpha
```

C.3 r-Leakage Matrix Entries for RZ BLD Gauss-Legendre on Quadrilaterals

```
(*r-Leakage matrix for RZ BLD Gauss-Legendre on \
quadrilaterals*)Clear[r0, r1, r2, r3, z0, z1, z2, z3, mr0, mr1, mr2, \
mr3, mz0, mz1, mz2, mz3, mu, rho, kappa]
(*Define physical integration point coordinates to compare to MFEM \
output: *)
r0 = (1 + (-Sqrt[1/3]))/(1 - (-1))*(0.25 - 0);
r1 = (1 + (Sqrt[1/3]))/(1 - (-1))*(0.25 - 0);
z0 = (1 + (-Sqrt[1/3]))/(1 - (-1))*(0.25 - 0);
z3 = (1 + (Sqrt[1/3]))/(1 - (-1))*(0.25 - 0);
(*Use to define rectangular mesh:*)
r2 = r1;
r3 = r0;
z1 = z0;
z2 = z3;
(*Transform arbitrary quadrilateral to reference square*)


r[rho_, kappa_] := (r1 - r0) rho + (r3 - r0) kappa + (r2 + r0 - r1 -
```

```
     r3) rho kappa + r0
z[rho_, kappa_] := (z1 - z0) rho + (z3 - z0) kappa + (z2 + z0 - z1 -
     z3) rho kappa + z0
(*Mesh nodes are needed to conserve cell volume.Cannot define mr0=0 \
nor mz0=0 here.Perform integration and then set the values.*)
\
(*Comment out to get the general mesh element mr1=0.25;
mz3=0.25;*)
mr2 := mr1
mr3 := mr0
mz1 := mz0
mz2 := mz3
mr[rho_, kappa_] := (mr1 - mr0) rho + (mr3 - mr0) kappa + (mr2 + mr0 -
      mr1 - mr3) rho kappa + mr0
mz[rho_, kappa_] := (mz1 - mz0) rho + (mz3 - mz0) kappa + (mz2 + mz0 -
      mz1 - mz3) rho kappa + mz0
a = {mr[rho, kappa], mz[rho, kappa]};
b = {rho, kappa};
(*Determinant of the Jacobian-for determining area of the physical \
element.*)
j = Det[D[a, {b}]];
(*Basis functions.The coefficients are from MATLAB script \
"GaussLegendreQuadrilateral.m".*)

BLD1[rho_, kappa_] :=
 3*rho*kappa - (3 + Sqrt[3])/2*rho - (3 + Sqrt[3])/2*
   kappa + (1 + Sqrt[3]/2)
```

```
BLD2[rho_, kappa_] := -3*rho*kappa + (3 + Sqrt[3])/2*rho +
  3/(3 + Sqrt[3])*kappa - 1/2
BLD3[rho_, kappa_] :=
 3*rho*kappa - 3/(3 + Sqrt[3])*rho -
  3/(3 + Sqrt[3])*kappa + (1 - Sqrt[3]/2)
BLD4[rho_, kappa_] := -3*rho*kappa +
  3/(3 + Sqrt[3])*rho + (3 + Sqrt[3])/2*kappa - 1/2
(*The r value gets transformed back to the reference element*)
Kr = \
-mu*Integrate[
   BLD1[rho, kappa]*D[BLD1[rho, kappa], rho]*j, {rho, 0, 1}, {kappa,
    0, 1}]
mr0 = 0.0;
mr1 = 0.25;
mz0 = 0.0;
mz3 = 0.25;
mu = -0.495004692;
BLD1[r0, z0]
rho = r1;
D[BLD1[rho, kappa], kappa]
Kr
```

C.4 z-Leakage Matrix Entries for RZ BLD Gauss-Legendre on Quadrilaterals

```
(*z-Leakage matrix for RZ BLD Gauss-Legendre on \
quadrilaterals*)Clear[r0, r1, r2, r3, z0, z1, z2, z3, mr0, mr1, mr2, \
```

```
mr3, mz0, mz1, mz2, mz3, xi, rho, kappa]
(*Define physical integration point coordinates to compare to MFEM \
output:*)
r0 = (1 + (-Sqrt[1/3]))/(1 - (-1))*(0.25 - 0);
r1 = (1 + (Sqrt[1/3]))/(1 - (-1))*(0.25 - 0);
z0 = (1 + (-Sqrt[1/3]))/(1 - (-1))*(0.25 - 0);
z3 = (1 + (Sqrt[1/3]))/(1 - (-1))*(0.25 - 0);

(*Use to define rectangular mesh:*)
r2 = r1;
r3 = r0;
z1 = z0;
z2 = z3;
(*Transform arbitrary quadrilateral to reference square*)

r[rho_, kappa_] := (r1 - r0) rho + (r3 - r0) kappa + (r2 + r0 - r1 -
     r3) rho kappa + r0
z[rho_, kappa_] := (z1 - z0) rho + (z3 - z0) kappa + (z2 + z0 - z1 -
     z3) rho kappa + z0
(*Mesh nodes are needed to conserve cell volume.Cannot define mr0=0 \
nor mz0=0 here.Perform integration and then set the values.*)
\
(*Comment out to get the general mesh element mr1=0.25;
mz3=0.25;*)
mr2 := mr1
mr3 := mr0
mz1 := mz0
```

```
mz2 := mz3
mr[rho_, kappa_] := (mr1 - mr0) rho + (mr3 - mr0) kappa + (mr2 + mr0 -
      mr1 - mr3) rho kappa + mr0
mz[rho_, kappa_] := (mz1 - mz0) rho + (mz3 - mz0) kappa + (mz2 + mz0 -
      mz1 - mz3) rho kappa + mz0
a = {mr[rho, kappa], mz[rho, kappa]};
b = {rho, kappa};
(*Determinant of the Jacobian-for determining area of the physical \
element.*)
j = Det[D[a, {b}]];
(*Basis functions.The coefficients are from MATLAB script \
"GaussLegendreQuadrilateral.m".*)

BLD1[rho_, kappa_] :=
 3*rho*kappa - (3 + Sqrt[3])/2*rho - (3 + Sqrt[3])/2*
   kappa + (1 + Sqrt[3]/2)
BLD2[rho_, kappa_] := -3*rho*kappa + (3 + Sqrt[3])/2*rho +
  3/(3 + Sqrt[3])*kappa - 1/2
BLD3[rho_, kappa_] :=
 3*rho*kappa - 3/(3 + Sqrt[3])*rho -
  3/(3 + Sqrt[3])*kappa + (1 - Sqrt[3]/2)
BLD4[rho_, kappa_] := -3*rho*kappa +
  3/(3 + Sqrt[3])*rho + (3 + Sqrt[3])/2*kappa - 1/2
(*The r value gets transformed back to the reference element*)
Kz = \
-xi*Integrate[
   BLD1[rho, kappa]*D[BLD1[rho, kappa], kappa]*j, {rho, 0, 1}, {kappa,
```

```
     0, 1}]
mr0 = 0.0;
mr1 = 0.25;
mz0 = 0.0;
mz3 = 0.25;
xi = -0.868890301;
D[BLD1[rho, kappa], kappa]
xi*j
Kz
Kr + Kz
```

C.5 z-Leakage Matrix Entries for RZ BLD Gauss-Legendre on Quadrilaterals

```
(*z-Leakage matrix for RZ BLD Gauss-Legendre on \
quadrilaterals*)Clear[r0, r1, r2, r3, z0, z1, z2, z3, mr0, mr1, mr2, \
mr3, mz0, mz1, mz2, mz3, xi, rho, kappa]
(*Define physical integration point coordinates to compare to MFEM \
output:*)
r0 = (1 + (-Sqrt[1/3]))/(1 - (-1))*(0.25 - 0);
r1 = (1 + (Sqrt[1/3]))/(1 - (-1))*(0.25 - 0);
z0 = (1 + (-Sqrt[1/3]))/(1 - (-1))*(0.25 - 0);
z3 = (1 + (Sqrt[1/3]))/(1 - (-1))*(0.25 - 0);

(*Use to define rectangular mesh:*)
r2 = r1;
r3 = r0;
```

```
z1 = z0;
z2 = z3;
(*Transform arbitrary quadrilateral to reference square*)

r[rho_, kappa_] := (r1 - r0) rho + (r3 - r0) kappa + (r2 + r0 - r1 -
     r3) rho kappa + r0
z[rho_, kappa_] := (z1 - z0) rho + (z3 - z0) kappa + (z2 + z0 - z1 -
     z3) rho kappa + z0
(*Mesh nodes are needed to conserve cell volume.Cannot define mr0=0 \
nor mz0=0 here.Perform integration and then set the values.*)
\
(*Comment out to get the general mesh element mr1=0.25;
mz3=0.25;*)
mr2 := mr1
mr3 := mr0
mz1 := mz0
mz2 := mz3
mr[rho_, kappa_] := (mr1 - mr0) rho + (mr3 - mr0) kappa + (mr2 + mr0 -
      mr1 - mr3) rho kappa + mr0
mz[rho_, kappa_] := (mz1 - mz0) rho + (mz3 - mz0) kappa + (mz2 + mz0 -
      mz1 - mz3) rho kappa + mz0
a = {mr[rho, kappa], mz[rho, kappa]};
b = {rho, kappa};
(*Determinant of the Jacobian-for determining area of the physical \
element.*)
j = Det[D[a, {b}]];
(*Basis functions.The coefficients are from MATLAB script \
```

```
"GaussLegendreQuadrilateral.m".*)


BLD1[rho_, kappa_] :=
 3*rho*kappa - (3 + Sqrt[3])/2*rho - (3 + Sqrt[3])/2*
   kappa + (1 + Sqrt[3]/2)
BLD2[rho_, kappa_] := -3*rho*kappa + (3 + Sqrt[3])/2*rho +
  3/(3 + Sqrt[3])*kappa - 1/2
BLD3[rho_, kappa_] :=
 3*rho*kappa - 3/(3 + Sqrt[3])*rho -
  3/(3 + Sqrt[3])*kappa + (1 - Sqrt[3]/2)
BLD4[rho_, kappa_] := -3*rho*kappa +
  3/(3 + Sqrt[3])*rho + (3 + Sqrt[3])/2*kappa - 1/2
(*The r value gets transformed back to the reference element*)
Kz = \
-xi*Integrate[
   BLD1[rho, kappa]*D[BLD1[rho, kappa], kappa]*j, {rho, 0, 1}, {kappa,
     0, 1}]
mr0 = 0.0;
mr1 = 0.25;
mz0 = 0.0;
mz3 = 0.25;
xi = -0.868890301;
D[BLD1[rho, kappa], kappa]
xi*j
Kz
Kr + Kz
```

C.6 r Surface-Integral Matrix Entries for RZ BLD Gauss-Legendre on Quadrilaterals

```
(*r surface-integral matrix for RZ BLD Gauss-Legendre on \
quadrilaterals*)
Clear[r0, r1, r2, r3, z0, z1, z2, z3]
(*Define physical integration point coordinates to compare to MFEM \
output:
r0=(1+(-Sqrt[1/3]))/(1-(-1))*(0.25-0);
r1=(1+(Sqrt[1/3]))/(1-(-1))*(0.25-0);
z0=(1+(-Sqrt[1/3]))/(1-(-1))*(0.25-0);
z3=(1+(Sqrt[1/3]))/(1-(-1))*(0.25-0);*)
(*Use to define rectangular \
mesh:*)
r2 = r1;
r3 = r0;
z1 = z0;
z2 = z3;
(*Transform arbitrary quadrilateral to reference square*)

r[rho_, kappa_] := (r1 - r0) rho + (r3 - r0) kappa + (r2 + r0 - r1 -
     r3) rho kappa + r0
z[rho_, kappa_] := (z1 - z0) rho + (z3 - z0) kappa + (z2 + z0 - z1 -
     z3) rho kappa + z0
a = {r[rho, kappa], z[rho, kappa]};
b = {rho, kappa};
(*Determinant of the Jacobian-for determining area of the physical \
```

```
element.*)
j = Det[D[a, {b}]];
(*Basis functions.The coefficients are from MATLAB script \
"GaussLegendreQuadrilateral.m".*)

BLD1[rho_, kappa_] :=
 3*rho*kappa - (3 + Sqrt[3])/2*rho - (3 + Sqrt[3])/2*
   kappa + (1 + Sqrt[3]/2)
BLD2[rho_, kappa_] := -3*rho*kappa + (3 + Sqrt[3])/2*rho +
  3/(3 + Sqrt[3])*kappa - 1/2
BLD3[rho_, kappa_] :=
 3*rho*kappa - 3/(3 + Sqrt[3])*rho -
  3/(3 + Sqrt[3])*kappa + (1 - Sqrt[3]/2)
BLD4[rho_, kappa_] := -3*rho*kappa +
  3/(3 + Sqrt[3])*rho + (3 + Sqrt[3])/2*kappa - 1/2
(*The r value gets transformed back to the reference element*)
Lr =
 Integrate[r1*BLD1[1, kappa]*BLD1[1, kappa]*j, {kappa, 0, 1}] -
  Integrate[r0*BLD1[0, kappa]*BLD1[0, kappa]*j, {kappa, 0, 1}]
r0 = 0.0;
r1 = 0.25;
z0 = 0.0;
z3 = 0.25;
```

C.7 z Surface-Integral Matrix Entries for RZ BLD Gauss-Legendre on Quadrilaterals

```
(*z surface-integral matrix for RZ BLD Gauss-Legendre on \
quadrilaterals*)
Clear[r0, r1, r2, r3, z0, z1, z2, z3]
(*Define physical integration point coordinates to compare to MFEM \
output:
r0=(1+(-Sqrt[1/3]))/(1-(-1))*(0.25-0);
r1=(1+(Sqrt[1/3]))/(1-(-1))*(0.25-0);
z0=(1+(-Sqrt[1/3]))/(1-(-1))*(0.25-0);
z3=(1+(Sqrt[1/3]))/(1-(-1))*(0.25-0);*)
(*Use to define rectangular \
mesh:*)
r2 = r1;
r3 = r0;
z1 = z0;
z2 = z3;
(*Transform arbitrary quadrilateral to reference square*)

r[rho_, kappa_] := (r1 - r0) rho + (r3 - r0) kappa + (r2 + r0 - r1 -
     r3) rho kappa + r0
z[rho_, kappa_] := (z1 - z0) rho + (z3 - z0) kappa + (z2 + z0 - z1 -
     z3) rho kappa + z0
a = {r[rho, kappa], z[rho, kappa]};
b = {rho, kappa};
(*Determinant of the Jacobian-for determining area of the physical \
```

```
element.*)
j = Det[D[a, {b}]];
(*Basis functions.The coefficients are from MATLAB script \
"GaussLegendreQuadrilateral.m".*)

BLD1[rho_, kappa_] :=
 3*rho*kappa - (3 + Sqrt[3])/2*rho - (3 + Sqrt[3])/2*
   kappa + (1 + Sqrt[3]/2)
BLD2[rho_, kappa_] := -3*rho*kappa + (3 + Sqrt[3])/2*rho +
  3/(3 + Sqrt[3])*kappa - 1/2
BLD3[rho_, kappa_] :=
 3*rho*kappa - 3/(3 + Sqrt[3])*rho -
  3/(3 + Sqrt[3])*kappa + (1 - Sqrt[3]/2)
BLD4[rho_, kappa_] := -3*rho*kappa +
  3/(3 + Sqrt[3])*rho + (3 + Sqrt[3])/2*kappa - 1/2
(*The r value gets transformed back to the reference element*)
Lz =
 Integrate[r[rho, 1]*BLD1[rho, 1]*BLD1[rho, 1]*j, {rho, 0, 1}] -
  Integrate[r[rho, 0]*BLD1[rho, 0]*BLD1[rho, 0]*j, {rho, 0, 1}]
r0 = 0.0;
r1 = 0.25;
z0 = 0.0;
z3 = 0.25;
Lz
Lr + Lz
```

C.8 LinearForm Angular Redistribution Matrix Entries for RZ BLD Gauss-Legendre on Quadrilaterals

```
(*LinearForm Angular Redistribution matrix for RZ BLD Gauss-Legendre \
on quadrilaterals*)Clear[r0, r1, r2, r3, z0, z1, z2, z3, mr0, mr1, \
mr2, mr3, mz0, mz1, mz2, mz3]
(*Define physical integration point coordinates to compare to MFEM \
output:r0=(1+(-Sqrt[1/3]))/(1-(-1))*(0.25-0);
r1=(1+(Sqrt[1/3]))/(1-(-1))*(0.25-0);
z0=(1+(-Sqrt[1/3]))/(1-(-1))*(0.25-0);
z3=(1+(Sqrt[1/3]))/(1-(-1))*(0.25-0);*)
(*These make the mesh \
rectangular:*)
z2 := z3
z1 := z0
r2 := r1
r3 := r0
(*Quadrilateral transformation back to reference square-more general \
than rectangle transofrmation-but NOT curved mesh*)

r[rho_, kappa_] := (r1 - r0) rho + (r3 - r0) kappa + (r2 + r0 - r1 -
     r3) rho kappa + r0
z[rho_, kappa_] := (z1 - z0) rho + (z3 - z0) kappa + (z2 + z0 - z1 -
     z3) rho kappa + z0

(*Mesh nodes are needed to conserve cell volume.Cannot define mr0=0 \
nor mz0=0 here.Perform integration and then set the values.*)
```

```
\
(*Comment out to get the general mesh element mr1=0.25;
mz3=0.25;*)
mr2 := mr1
mr3 := mr0
mz1 := mz0
mz2 := mz3
mr[rho_, kappa_] := (mr1 - mr0) rho + (mr3 - mr0) kappa + (mr2 + mr0 -
      mr1 - mr3) rho kappa + mr0
mz[rho_, kappa_] := (mz1 - mz0) rho + (mz3 - mz0) kappa + (mz2 + mz0 -
      mz1 - mz3) rho kappa + mz0
a = {mr[rho, kappa], mz[rho, kappa]};
b = {rho, kappa};
(*Determinant of the Jacobian-for determining area of the physical \
element.*)
j = Det[D[a, {b}]];
(*Basis functions.The coefficients are from MATLAB script \
"GaussLegendreQuadrilateral.m".*)

BLD1[rho_, kappa_] :=
 3*rho*kappa - (3 + Sqrt[3])/2*rho - (3 + Sqrt[3])/2*
   kappa + (1 + Sqrt[3]/2)
BLD2[rho_, kappa_] := -3*rho*kappa + (3 + Sqrt[3])/2*rho +
  3/(3 + Sqrt[3])*kappa - 1/2
BLD3[rho_, kappa_] :=
 3*rho*kappa - 3/(3 + Sqrt[3])*rho -
  3/(3 + Sqrt[3])*kappa + (1 - Sqrt[3]/2)
```

```
BLD4[rho_, kappa_] := -3*rho*kappa +
  3/(3 + Sqrt[3])*rho + (3 + Sqrt[3])/2*kappa - 1/2
M = Integrate[
  BLD2[rho, kappa]*BLD2[rho, kappa]*j, {rho, 0, 1}, {kappa, 0, 1}]


mr0 = 0;
mz0 = 0;
mr1 = 0.25;
mz3 = 0.25;


alphatau = 0.845025867;
psiMinusHalf = 0.1591549;
M*alphatau*psiMinusHalf
```

C.9 LinearForm Scattering/Fixed Source Matrix Entries for RZ BLD Gauss-Legendre on Quadrilaterals

```
(*LinearForm scattering/fixed sources for RZ BLD Gauss-Legendre on \
quadrilaterals*)Clear[r0, r1, r2, r3, z0, z1, z2, z3, mr0, mr1, mr2, \
mr3, mz0, mz1, mz2, mz3, Source]
(*Define physical integration point coordinates to compare to MFEM \
output:r0=(1+(-Sqrt[1/3]))/(1-(-1))*(0.25-0);
r1=(1+(Sqrt[1/3]))/(1-(-1))*(0.25-0);
z0=(1+(-Sqrt[1/3]))/(1-(-1))*(0.25-0);
z3=(1+(Sqrt[1/3]))/(1-(-1))*(0.25-0);*)
(*These make the mesh \
rectangular:*)
```

```
z2 := z3
z1 := z0
r2 := r1
r3 := r0
(*Quadrilateral transformation back to reference square-more general \
than rectangle transofrmation-but NOT curved mesh*)

r[rho_, kappa_] := (r1 - r0) rho + (r3 - r0) kappa + (r2 + r0 - r1 -
     r3) rho kappa + r0
z[rho_, kappa_] := (z1 - z0) rho + (z3 - z0) kappa + (z2 + z0 - z1 -
     z3) rho kappa + z0

(*Mesh nodes are needed to conserve cell volume.Cannot define mr0=0 \
nor mz0=0 here.Perform integration and then set the values.*)
\
(*Comment out to get the general mesh element mr1=0.25;
mz3=0.25;*)
mr2 := mr1
mr3 := mr0
mz1 := mz0
mz2 := mz3
mr[rho_, kappa_] := (mr1 - mr0) rho + (mr3 - mr0) kappa + (mr2 + mr0 -
      mr1 - mr3) rho kappa + mr0
mz[rho_, kappa_] := (mz1 - mz0) rho + (mz3 - mz0) kappa + (mz2 + mz0 -
      mz1 - mz3) rho kappa + mz0
a = {mr[rho, kappa], mz[rho, kappa]};
b = {rho, kappa};
```

```
(*Determinant of the Jacobian-for determining area of the physical \
element.*)
j = Det[D[a, {b}]];
(*Basis functions.The coefficients are from MATLAB script \
"GaussLegendreQuadrilateral.m".*)

BLD1[rho_, kappa_] :=
 3*rho*kappa - (3 + Sqrt[3])/2*rho - (3 + Sqrt[3])/2*
   kappa + (1 + Sqrt[3]/2)
BLD2[rho_, kappa_] := -3*rho*kappa + (3 + Sqrt[3])/2*rho +
  3/(3 + Sqrt[3])*kappa - 1/2
BLD3[rho_, kappa_] :=
 3*rho*kappa - 3/(3 + Sqrt[3])*rho -
  3/(3 + Sqrt[3])*kappa + (1 - Sqrt[3]/2)
BLD4[rho_, kappa_] := -3*rho*kappa +
  3/(3 + Sqrt[3])*rho + (3 + Sqrt[3])/2*kappa - 1/2
M = Integrate[
  Source*mr[rho, kappa]*BLD2[rho, kappa]*j, {rho, 0, 1}, {kappa, 0, 1}]
(*Source=0.7/(2*Pi);*)
Source = (0.3*1)/(2*Pi);
mr0 = 0;
mz0 = 0;
mr1 = 0.25;
mz3 = 0.25;
M
```