LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

LLNL-TR-636155

A Variational Method for Interpolation Between Zone Centers in 2-D Geometry

E. D. Brooks III and A. Szoke

May 1, 2013

Disclaimer

Auspices Statement

This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

A Variational Method for Interpolation Between Zone Centers in 2-D Geometry*

Eugene D. Brooks III and Abraham Szőke
Lawrence Livermore National Laboratory
P.O. Box 808, Livermore, CA 94550, USA

May 1, 2013

Abstract

Radiation-hydrodynamics codes are often based on a zone centered discretization strategy for the hydrodynamics, while transport requires a higher order interpolated treatment of the material state variable to produce a correct solution in optically thick media. In this paper, we use a variational method to generalize a previously developed 1-D interpolation method to 2-D geometries. While our interpolation method was developed to handle the gradient source term for thermal photons in the Difference Formulation, it may also be useful for the generation of an accurate zone centered discretization for diffusion methods. We provide a detailed description of the 2-D interpolation method and how it relates to the 1-D method that was developed earlier. We also discuss how this interpolation strategy can be extended to 3-D, but do not work through the details.

*This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

1 Introduction

Accurate radiation transport was made practically feasible, using a single method for both optically thick and thin media, by the Symbolic Implicit Monte Carlo (SIMC) method in the Difference Formulation [1]. As an example, in 1-D slab geometry a Marshak wave was found to propagate at the correct speed even when individual zones are 100 optical depths thick with a method that also preserves accuracy in the free streaming limit.

When a zone centered discretization of the material state variable[1] is constant within each zone, excess energy flow results from the discontinuity at the interface between zones. This problem arises because the source associated with the gradient of the material state variable is concentrated at the zone interface and does not incur absorption losses. We removed this excess energy flow in 1-D using a source interpolation method.

In this paper, we address the question of extending our source interpolation method to higher dimensions. In 1-D slab geometry, our interpolation method derives its fidelity from demanding the continuity of both the material state variable and the diffusion flux across the interface between zones. It will be recognized as a straightforward discretization of the diffusion equation starting with zone centered values. We generalize the 1-D interpolation method to 2-D by reinterpreting the 1-D interpolation as the equilibrium solution of the diffusion equation given the zone centered values as boundary conditions. Using this re-interpretation, we use the variational method to produce an interpolation that satisfies the same requirement in 2-D.

In 1-D, the interpolation across half-zones is a straight line which has zero divergence. As the diffusion flux is matched on the left and right sides of zone interfaces there is zero divergence at interfaces between zones. Any divergence of the diffusion flux occurs at zone centers where it is properly associated with heating or cooling of the zone. In 2-D, an N sided zone is divided into N four-sided polygons that we refer to as corners. Optimized Stone-Adams basis functions are used to represent the material state variable within each corner, providing zero divergence on its perimeter.

[1]The material state variable for the Difference Formulation of transport is $\Phi = aT^4$, where T is the material temperature and a is the radiation constant.

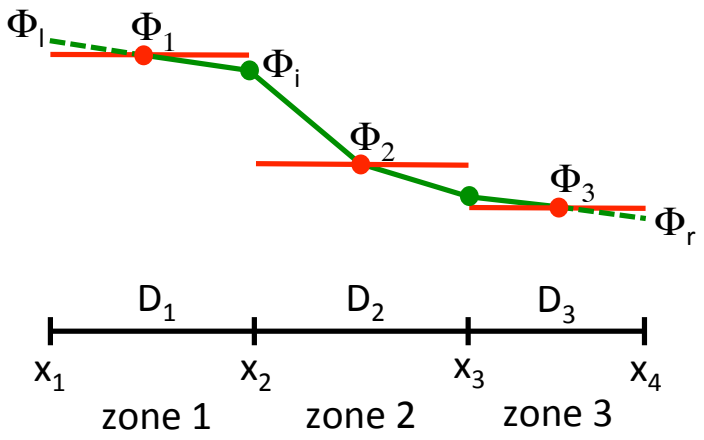

Figure 1: Interpolation between zone centers in 1-D slab geometry.

2 The 1-D Interpolation

Our interpolation of the gradient source term in the Difference Formulation remedied the excess energy flow between zones that is produced by a discretization of the material state variable that is constant in each zone. This excess energy flow causes thermal energy to propagate through optically thick media at faster than physical rates. We work through the details of this interpolation in 1-D in what follows so that it can be referred to in our discussion of the 2-D interpolation.

In Figure 1, we show the interpolation for a three zone problem in slab geometry. The original discretization, constant in a zone, is shown in red with zone center values labeled Φ_1, Φ_2, and Φ_3. We denote $\Phi = aT^4$, where T is the common temperature of the material and the reference field for the radiation, with a being the radiation constant. We want to construct an interpolation between these zone center values. Each zone may have its own diffusion coefficient, D_1, D_2, and D_3. It is the discontinuity at the interface between the zones that causes excess energy flow if the zones are optically thick. The interpolation, when it is properly done, resolves this problem.

2.1 The interpolation between two zone centers

The interpolation for an interior region between zone centers, shown with the solid green line in Figure 1, is produced by demanding that both Φ and the diffusion flux, $F = -D\nabla\Phi$ where D is the diffusion coefficient, is matched at the interface between each pair of zones.

Referring to Figure 1, we want to establish the value of Φ_i such that

$$\frac{2D_2(\Phi_2 - \Phi_i)}{x_3 - x_2} = \frac{2D_1(\Phi_i - \Phi_1)}{x_2 - x_1} \quad . \tag{1}$$

This interpolation has a change in slope at the interface between zones if the diffusion coefficients of the zones, D_1 and D_2, are not equal. This will be recognized as the usual interior discretization of the diffusion equation that establishes the energy flow between two zones during a time step. The same template is used for the interface between zones 2 and 3, leading to a change in slope at the center of zone 2, shown in the figure. The difference in flux here is associated with heating or cooling the zone.

Preparing our path forward for 2-D, it is useful to consider the variational approach [2] that arrives at the same interpolation of Eq. 1. In this case we want an approximate solution of the time independent diffusion equation between the two zone centers, using the zone center values as boundary conditions.

In order to get this, we vary Φ_i in order to minimize the integral of the diffusion coefficient times the square of the gradient of Φ over the two half zones. This integral is

$$D_1\frac{(x_2 - x_1)}{2}\left(\frac{2(\Phi_i - \Phi_1)}{x_2 - x_1}\right)^2 + D_2\frac{(x_3 - x_2)}{2}\left(\frac{2(\Phi_2 - \Phi_i)}{x_3 - x_2}\right)^2 \quad . \tag{2}$$

The minimum, varying Φ_i, is obtained by demanding that the derivative of Eq. 2 with respect to Φ_i is zero. Taking the derivative, setting it to zero and simplifying, we end up with

$$\frac{2D_2(\Phi_2 - \Phi_i)}{x_3 - x_2} = \frac{2D_1(\Phi_i - \Phi_1)}{x_2 - x_1} \quad . \tag{3}$$

For 1-D, this variational approach delivers the exact solution. This amounts to a textbook demonstration of the variational method for the solution of the diffusion equation, the 2-D extension of which is neither simple, nor as exact. Without the guidance of the 1-D case we would not know where to start in 2-D.

2.2 Handling boundary conditions in 1-D

In a boundary zone we extrapolate across a half zone to a problem boundary, maintaining the slope that was established by the interior interpolation. Examples of this are shown in Figure 1 by the dashed green lines to the value at the left problem boundary, giving Φ_l, and to the value at the right problem boundary, giving Φ_r.

These zero curvature extrapolations to the problem boundaries may be constrained by the boundary conditions. If the boundary is reflecting, the slope approaching the boundary must be zero. Constraints on the zero curvature extrapolation can also arise when the boundary condition represents an incident black body. An extrapolation is not allowed to produce a discontinuity with respect to an external boundary condition that would oppose the direction of energy flow that otherwise occurs with the original constant-in-a-zone discretization.

As this curious handling of the extrapolation to the boundary is unusual, we cite an example to clarify the situation. Suppose the boundary condition on the left hand side for Figure 1 is an incident black body corresponding to Φ_{LB} with $\Phi_l > \Phi_{LB} > \Phi_1$. In this case the zero curvature extrapolation from the interior would produce a surface temperature that is hotter than the blackbody source, and the associated source term (at the surface) would cool the zone instead of heating it. To prevent this we limit the extrapolated value to Φ_{LB}.

One might also limit the extrapolation to a Milne boundary condition. A detailed discussion of boundary conditions for transport, and suitable boundary conditions for diffusion approximation for transport, is beyond the scope of this report.

3 The 2-D Interpolation

In order to create the 2-D interpolation, we must generalize the concept of 1-D half-zones to 2-D and we would like to do this in a manner that reproduces the 1-D treatment described above when the 2-D problem has the appropriate symmetry. We satisfy this requirement by using "corners", four-sided sub-divisions of an N-sided zone. The notion of corners is not new, these were used by Burton [3] and Adams [4], for different purposes.

We show this construction in Figure 2 for a pentagonal zone, to illustrate

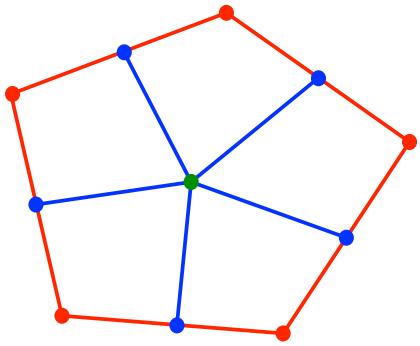

Figure 2: Division of a pentagonal zone into five four-sided corners. The original zone is shown in red. The division is accomplished by picking a center for the zone, shown in green, and connecting it to the mid-points of the edges of the zone, shown in blue.

the fact that our interpolation strategy is not restricted to four-sided zones on a logically rectangular mesh. The original pentagonal zone is shown in red with the nodes of the mesh shown as red dots. The location of the zone center is chosen somewhat arbitrarily, but must result in positive area corners, and is shown as the interior green dot. The mid-points of the sides of the original zone are marked with blue dots. The pentagonal zone is divided into five four-sided corners by the blue lines drawn from the zone center to the mid-points on the sides.

The mesh constructed from the corners of the zones comprising the original mesh is referred to as the corner mesh in what follows. The nodes of the corner mesh are the original nodes of the problem mesh, augmented by the zone centers and zone edge mid-points.

3.1 Using optimized Stone-Adams basis functions in a corner

We now need to represent our interpolation in a corner, given the values of Φ on the nodes of the corner mesh and assuming that these values interpolate linearly along the edges of the corner mesh. We use an optimized form [5] of the Stone-Adams basis functions [6] for this purpose. In this construction one selects a center for the corner, again somewhat arbitrarily, and the corner is then divided into four triangles by drawing lines from the center of the

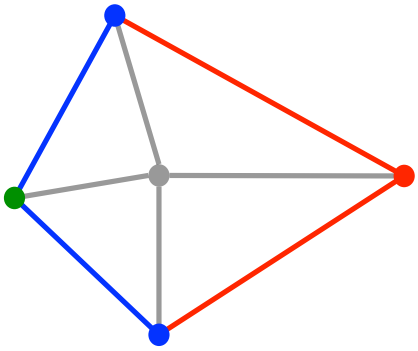

Figure 3: Subdivision of a corner into triangles. The red and blue color coding correspond to those of Figure 2 above. A center for the corner is selected, shown with the grey dot, and the corner is subdivided into triangles by connecting its nodes to the center, shown with the grey lines.

corner to the nodes[2] of the corner. This construction is shown in Figure 3.

There are four Stone-Adams basis functions in the corner, each having the value 1 at the i'th node of the corner, α_i at the center, and zero at the other nodes. The values of Φ in the interior of the triangles vary linearly, determined by the values at the center and the nodes. The value of α_i for each basis function is chosen to optimize the basis function as a representation for solutions of the Laplace equation, this being appropriate given the fact that the diffusion coefficient is constant in a corner. The value at the center of the corner is given by

$$\Phi_c = \sum_i \alpha_i \, \Phi_i \quad , \tag{4}$$

where Φ_i is the value at node i of the corner and the α_i, having the property

$$\sum_i \alpha_i = 1 \quad , \tag{5}$$

depend on the locations of the nodes [5]. The benefit of the optimization is zero divergence around the perimeter of a corner, a desired property given our goal of representing solutions of the diffusion equation with our interpolation.

[2]We will refer to the vertices of polygons as nodes because the polygons are always buried in a mesh. The terms edges and sides are also used interchangeably, depending on the context, and refer to the same thing.

3.2 Using the 1-D ansatz to interpolate to mid-points of zone edges

At this point we have described the geometric construction used to convert the problem mesh to a corner mesh, with values of Φ being defined everywhere by the basis functions if we know them at *all* the nodes of the corner mesh. At the moment we know the values of Φ only at the nodes of the corner mesh associated with the center of each original zone. To produce values for the other nodes of the corner mesh, we have to interpolate (or extrapolate) in a similar way as we did in our 1-D example. We will use a bootstrap process that starts with the mid-points along the interior zone edges.

In Figure 4 we show such a situation. Two four-sided zones sharing an interior edge, labeled zone 1 and zone 2, are shown with the zone edges and associated nodes of the problem mesh in red. The zones are divided into corners by selecting zone centers and connecting the mid-points of the zone edges to the centers. At this point we interpolate the value at the mid-point of the interior edge from N1 to N2, shown with the blue dot. We find a value here by conserving the normal component of the energy flow across the zone edge at this point. We are only interested in the flux in the vicinity of the mid-point along the zone edge, but it is sufficient to consider the two triangles contained by the dashed orange lines, assuming that the values of Φ at N1 and N2 are the same as that at the mid-point.

The value at the mid-point of the zone edge that satisfies our desired conditions can be found in closed form given the locations of C1, C2, N1, N2; the zone center values of Φ that are associated with C1 and C2; and the diffusion coefficient for the zones, D1 and D2. This value is given by the expression

$$\frac{\begin{array}{r} C2_\Phi D2(C1_y(N1_x - N2_x) + N1_y(N2_x - C1_x) + N2_y(C1_x - N1_x)) \\ + C1_\Phi D1(N1_y(C2_x - N2_x) + C2_y(N2_x - N1_x) + N2_y(N1_x - C2_x)) \end{array}}{\begin{array}{r} D1(C2_y(N2_x - N1_x) + C2_x(N1_y - N2_y) + N1_x N2_y - N1_y N2_x) \\ + D2(C1_y(N1_x - N2_x) + C1_x(N2_y - N1_y) + N1_y N2_x - N1_x N2_y) \end{array}} \quad , \quad (6)$$

where the subscript x refers to the x component of a location, the subscript y refers to the y component, and the subscript Φ refers to the value of Φ there. The Mathematica [7] code used to compute the expression is shown in Appendix A. Using this template, we find the value of Φ at the mid-point of all interior zone edges.

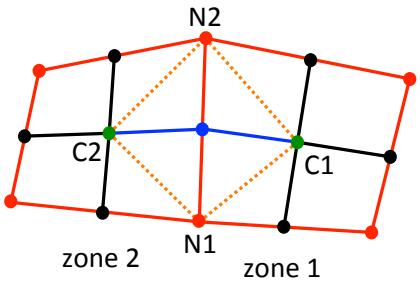

Figure 4: The geometric construction used to interpolate the value of Φ at the mid-points of interior zone edges. The zone edges and nodes are shown in red. The zone centers are shown in green. Most of the mid-points of the zone edges are shown in black, but the interior one in question is shown in blue. There is a change in direction possible for the blue line as it crosses the zone edge due to the irregular geometry. The triangles delimited by the dashed orange lines are used to bootstrap the value at the edge mid-point shown with the blue dot.

3.3 Using the variational solution for interior nodes

When the values of Φ are known at the mid-points of interior zone edges we find that the interior nodes of the problem mesh, the values of Φ for which are as yet undetermined, are surrounded by known values. This is illustrated in Figure 5 where we have labeled the interior node of the problem mesh by N. Let us concentrate on the "dual zone" with its perimeter shown in blue. This dual zone is composed of the four zone corners that touch node N of the problem mesh. Assuming a linear interpolation along a line segment connecting two nodes with known values of Φ, we realize that the value of Φ is now known at all points on the perimeter of the dual zone containing the node labeled N, providing a Dirichlet boundary condition.

We now further divide each corner comprising the dual zone into 4 triangles using the gray dashed lines. We use the optimized Stone-Adams basis functions to represent Φ everywhere in each corner, given a trial value for Φ at the node labeled N, and thereby everywhere in the dual zone. The Stone-Adams basis functions provide for linear interpolation between the nodes, agreeing with the boundary condition for the dual zone.

At this point the variational method is used to obtain the value for Φ at the node labeled N. Labeling the zones by their zone centers, CNE, CNW, CSE and CSE; the diffusion coefficients for each zone (not shown) are simi-

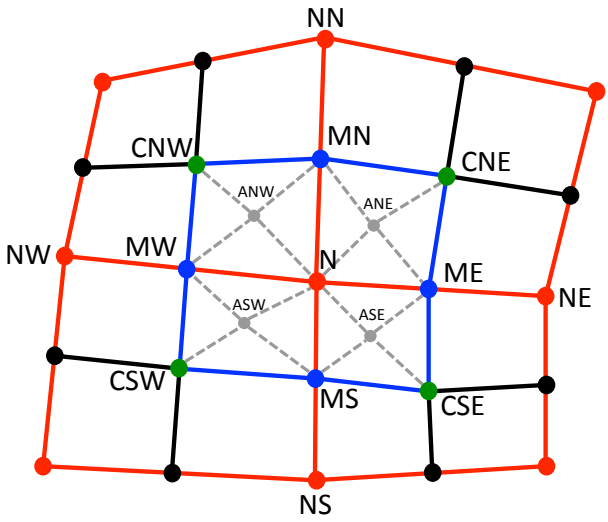

Figure 5: The template used to find the value of Φ at an interior node of the problem mesh.

larly denoted DNE, DNW, DSE and DSE. The value of Φ at the node labeled N is obtained by minimizing the integral of the square of the gradient of Φ times the diffusion coefficient over the dual zone. The gradient is a constant, as a function of space, in each triangle, so the integral is a sum over the 16 triangles. The minimum, varying the value of Φ at the node labeled N, is obtained taking the derivative of the integral with respect to the value of Φ at the node labeled N and demanding that it be zero, similarly to the 1-D example we discussed earlier. This tedious calculation is carried out using the Mathematica code shown in Appendix B. The expressions involved are too complicated to usefully display here and the C code to calculate Φ at node N in the template is generated automatically using Mathematica. The template is applied to establish the value of Φ at every interior node of the problem mesh.

3.4 Handling boundary conditions in 2-D

We have calculated so far the values of Φ for all interior nodes of the corner mesh. We now need to find Φ for the nodes on the problem boundaries. The situation for the extrapolation to a problem boundary is more complicated

for 2-D than it was for 1-D. A relatively simple nine zone (three by three) problem mesh is shown in Figure 6. The nodes of the problem mesh are indicated with red symbols, with the edges of zones being the straight red lines that connect them. Zone centers are indicated by the green dots, and the zones are divided into corners by the lines connecting the zone centers to the mid-points of the zone edges. These lines are blue where the method of Section 3.2 was used to establish the value of Φ at the mid-point of a zone edge, and define the perimeter of the dual zones where the method of Section 3.3 was used to establish the value of Φ at the enclosed interior node of the problem mesh, shown as a red dot.

The extrapolation to the exterior nodes on the problem boundary shown in Figure 6 as red pentagons is straightforward. The zone edge from the interior node of the problem mesh, shown as a red dot, through the mid-point, shown as a blue dot, to the node on the problem boundary is a straight line. An extrapolation along this line, using the previously determined values at the mid-point and interior node, provides a value for the node on the problem boundary.

Extrapolation along the path from an interior mid-point of a zone edge, shown as a blue dot, through a zone center, shown as a green dot, to an exterior mid-point, shown as a blue box, is not so straightforward because the path is not a straight line. An example is shown by the path marked MI, C and ME on the left side of Figure 6. To handle the change in direction we use the template shown in Figure 7. The mid-point of the exterior edge is labeled ME, the zone center labeled C, and the mid-point of the interior zone edge is labeled MI. We also assume that the zone edges with mid-points labeled MU and MD are interior zone edges, so we know the values of Φ at points C, MI, MU and MD. For the purpose of clarifying our extrapolation, we have added the dashed blue lines that produce four triangles which touch the zone center. The gradient of Φ is known for the two triangles on the right hand side in Figure 7, assuming a plane that passes through the points. Two values for Φ for the exterior node ME are produced by demanding the continuity of the gradient across the line from C to MU, and the continuity of the gradient across the line from MD to C. We take the average of these two values to produce the value used for the node ME. This template is used to produce a value of Φ for all boundary nodes of the corner mesh that are not problem corners, or adjacent to problem corners, marked by red and green squares in Figure 6.

The value of Φ at the nodes of the corner mesh adjacent to the problem

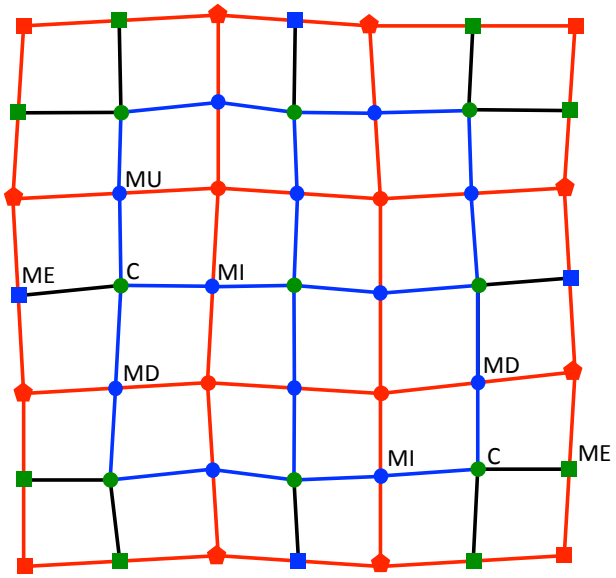

Figure 6: A simple mesh including boundary nodes. The problem mesh is shown in red. The red lines connecting red symbols are the edges of zones, with the green dots marking the zone centers. The interior dual zones, surrounding interior nodes of the problem mesh, are shown in blue. The boundary nodes of the corner mesh have different symbols and colors, indicating the method used to determine Φ as described in the text. The application of the full template of Figure 7 is shown on the left side of the mesh. The application of half of the template is shown at the lower right.

corners, marked by green squares in Figure 6, is obtained by applying half of the template shown in Figure 7 because of the unknown value on the other side of the problem corner. One of several examples is shown at the lower right of Figure 6. Finally, the value of Φ at the problem corners, marked by red squares, is obtained by extrapolating along each edge to the corner and averaging the result.

We would like to point out that one can do better with the extrapolations to boundary edges than we have implemented, by following the principle of minimizing curvature (the component of the change in slope perpendicular to the edge) more rigorously. Instead of averaging two planar extrapolations, referring to Figure 7, we can vary the value of Φ at point ME to minimize the length weighted curvature along the line segments from MD to C and C to

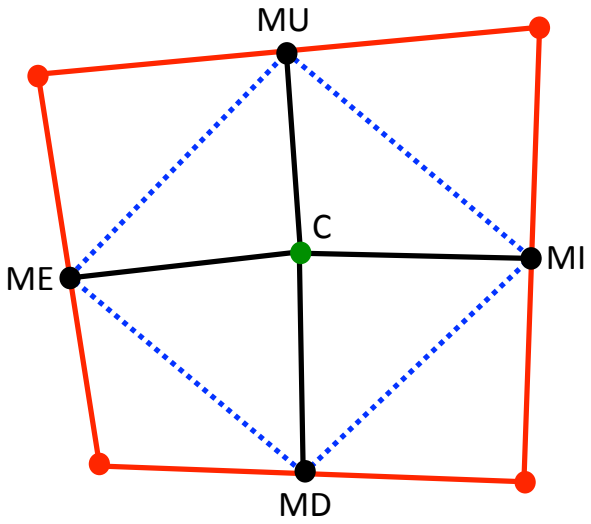

Figure 7: Template for extrapolation through a zone center to a mid-point on a problem boundary. ME labels the exterior mid-point, while MI labels the interior one. The values of Φ at C, MD, MI and MU are known.

MU. Similarly, the values of Φ at the exterior corners of the problem mesh, shown with red squares in Figure 6, along with the values at the adjacent mid-points of the exterior zone edges that end at the corner, shown with green squares, can be solved for simultaneously, again by minimizing curvature on the appropriate line segments.

As was the case for 1-D, these extrapolations can be limited by the boundary conditions for the problem. If the boundary is a symmetry boundary or is reflecting, the extrapolation is constrained to have zero slope as it approaches the boundary. If the boundary condition for the problem has a value for Φ, representing an incident black body source (not to be confused with a Dirichlet boundary condition), the extrapolation is limited to the value at the boundary if an overshoot (or undershoot) produced by the extrapolation, relative to the boundary condition, would generate heat flow that opposes that which would occur for a constant in a zone discretization for Φ.

4 Generalizing to 3-D

Although we have not worked out the generalization of this interpolation method to 3-D, we would like to point out that the bootstrap strategy that we have employed for 2-D, first using a 1-D like method to get the values of Φ at the mid-points of the interior zone edges, then using a variational method for the rest, has an analog in 3-D. One uses a 1-D like method to get the values at the centers of zone faces, then the 2-D variational method to get values at the mid-points of zone edges, and finally a 3-D variational method to get the values at the nodes of the problem mesh. This would be a very tedious, but straightforward, undertaking that is left as an exercise for the reader.

5 Discussion

We have described the 2-D extension of a previously developed 1-D interpolation between zone centers that has been developed for the gradient source term of the Difference Formulation of transport. Our interpolation method is a bootstrap approach where an interpolation similar to the previous 1-D method is used to find the values at mid-points of interior zone edges, and then a variational method is used to find the values at interior nodes of the problem mesh. Values on the problem boundary are established with extrapolations that attempt to minimize curvature, subject to constraints imposed by the problem boundary conditions.

The method takes advantage of an optimized form of the Stone-Adams piecewise linear basis functions. The optimization provides for zero divergence on the perimeter of a corner while providing a constant gradient in each of four triangles that a corner is divided into. The constant gradient provides easy sampling of the gradient source term associated with the Difference Formulation. The description of the 2-D implementation of the Difference Formulation and its effectiveness in solving transport problems involving both optically thin and thick media will be described in a separate report.

In addition to its use in computing the gradient source term in the Difference Formulation, we speculate that our interpolation might be used to produce an optimized zone centered discretization of the diffusion equation. This might be accomplished by solving for the flux between zones as a func-

tion of unknown zone centered values.

Appendix A

Mathematica code used to produce the "1-D like" solution at the mid-point of a zone edge between two zone centers.

```
AppendTo[$Echo, "stdout"]
Off[General::spell]
Off[General::spell1]
SetOptions[$Output,PageWidth->174]

(* Find the value interpolated to the interior zone edge from two
zone centers, assuming linear treatment within a zone and conserving
the perpendicular component of the diffusion flux across the zone edge.
A plane is constructed by assuming the same value of z (phi) along
the zone edge is constant, assigning this unknown value to nodes N1
and N2. *)

(* Given three points, (X1,Y1,Z3), (X2,Y2,Z3), (X3,Y3,Z3);
the equation of a plane is AA x + BB y + CC z + DD = 0 . *)

AA[X1_,Y1_,Z1_,X2_,Y2_,Z2_,X3_,Y3_,Z3_] = Y1(Z2 - Z3) + Y2(Z3 - Z1) + Y3(Z1 - Z2)
BB[X1_,Y1_,Z1_,X2_,Y2_,Z2_,X3_,Y3_,Z3_] = Z1(X2 - X3) + Z2(X3 - X1) + Z3(X1 - X2)
CC[X1_,Y1_,Z1_,X2_,Y2_,Z2_,X3_,Y3_,Z3_] = X1(Y2 - Y3) + X2(Y3 - Y1) + X3(Y1 - Y2)
DD[X1_,Y1_,Z1_,X2_,Y2_,Z2_,X3_,Y3_,Z3_] = -X1(Y2 Z3 - Y3 Z2) -X2(Y3 Z1 - Y1 Z3) -X3(Y1 Z2 - Y2 Z1)

(* Z = Expand[(AA x + BB y + DD)/(-CC)] *)

(* The components of the gradient of Z(x,y). *)
dZdx[X1_,Y1_,Z1_,X2_,Y2_,Z2_,X3_,Y3_,Z3_] = -AA[X1,Y1,Z1,X2,Y2,Z2,X3,Y3,Z3]/CC[X1,Y1,Z1,X2,Y2,Z2,X3,Y3,Z3]
dZdy[X1_,Y1_,Z1_,X2_,Y2_,Z2_,X3_,Y3_,Z3_] = -BB[X1,Y1,Z1,X2,Y2,Z2,X3,Y3,Z3]/CC[X1,Y1,Z1,X2,Y2,Z2,X3,Y3,Z3]

(* Our triangle to the right of the edge has the corners N1, C1 and N2 going
around clockwise.  N1 and N2 are the coordinates of the nodes while C1 is the
coordinate of the zone center.  The triangle to the left of the edge is given
by N2, C2 and N1, again going around clockwise. *)

GradC1x = dZdx[N1x,N1y,p,C1x,C1y,C1p,N2x,N2y,p]
GradC1y = dZdy[N1x,N1y,p,C1x,C1y,C1p,N2x,N2y,p]

GradC2x = dZdx[N2x,N2y,p,C2x,C2y,C2p,N1x,N1y,p]
GradC2y = dZdy[N2x,N2y,p,C2x,C2y,C2p,N1x,N1y,p]

(* Construct the normal to the zone edge.  We
ignore the normalization because we are ultimately
only interested in the continuity of the
perpendicular component of the gradient. *)

Nx = - (N2y - N1y)
Ny = (N2x - N1x)

(* Finally, the normal component of the flux is preserved. *)
```

```
right = D1 (GradC1x Nx + GradC1y Ny)
left = D2 (GradC2x Nx + GradC2y Ny)

sol = Solve[right - left == 0, p]

s = FullSimplify[Part[Part[Part[sol,1],1],2]]

(* CForm is used to produce the expression used in the code. *)
CForm[s] >> "bluedots.h"

(* Mathematica does not always simplify fully.
Some manipulation of the variable names solves the problem. *)
numerator = Part[s,2]
numerator = Expand[numerator]
numerator = Collect[numerator, {D1 C1p,D2 C2p}]
numerator = Simplify[numerator]
numerator = ReplaceAll[numerator,{N2y -> aN2y}]
numerator = Simplify[numerator]
numerator = ReplaceAll[numerator,{aN2y -> N2y}]

denominator = 1/Part[s,1]
denominator = Expand[denominator]
denominator = Collect[denominator, {D1,D2}]
denominator = Simplify[denominator]
denominator = ReplaceAll[denominator,{N1x -> aN1x}]
denominator = Simplify[denominator]
denominator = ReplaceAll[denominator,{aN1x -> N1x}]

(* The expression for phi appearing in the paper is: *)
numerator / denominator

Exit
```

Appendix B

Mathematica code to produce the variational solution for the node of the problem mesh in the center of a dual zone. The resulting expression for the derivative is separated into 16 terms linear in Φ and 16 constant terms. The C code for the expressions is included in the Difference Formulation transport package to add up the terms linear in Φ, and the constant ones, separately, and then perform the divide to produce the value of Φ for the node.

```
AppendTo[$Echo, "stdout"]
Off[General::spell]
Off[General::spell1]
SetOptions[$Output,PageWidth->174]

(* Interpolate to the node between four zone centers,
given values at the zone centers and the midpoints of
the zone edges.  Each corner of a zone (a sub-quad defined
by zone centers, nodes, and midpoints of zone edges) is
```

```
treated with a piecewise linear basis function using four
triangles and a center that is the average of the locations
of its corners. *)

(* Signed Area of a triangle given by three points in the (x,y) plane.
The area is positive if the ordering of the points is counter clockwise.
The area of a triangle is half the cross product that computes the
area of the corresponding parallelogram. *)
AREATRI[x1_,y1_,x2_,y2_,x3_,y3_] = ((x2-x1)(y3-y1)-(y2-y1)(x3-x1))/2

(* Given three points, (X1,Y1,Z3), (X2,Y2,Z3), (X3,Y3,Z3);
the equation of a plane is AA x + BB y + CC z + DD = 0 . *)
AA[X1_,Y1_,Z1_,X2_,Y2_,Z2_,X3_,Y3_,Z3_] = Y1(Z2 - Z3) + Y2(Z3 - Z1) + Y3(Z1 - Z2)
BB[X1_,Y1_,Z1_,X2_,Y2_,Z2_,X3_,Y3_,Z3_] = Z1(X2 - X3) + Z2(X3 - X1) + Z3(X1 - X2)
CC[X1_,Y1_,Z1_,X2_,Y2_,Z2_,X3_,Y3_,Z3_] = X1(Y2 - Y3) + X2(Y3 - Y1) + X3(Y1 - Y2)
DD[X1_,Y1_,Z1_,X2_,Y2_,Z2_,X3_,Y3_,Z3_] = -X1(Y2 Z3 - Y3 Z2) -X2(Y3 Z1 - Y1 Z3) -X3(Y1 Z2 - Y2 Z1)

(* Z = Expand[(AA x + BB y + DD)/(-CC)]
The components of the gradient of Z(x,y). *)
dZdx[X1_,Y1_,Z1_,X2_,Y2_,Z2_,X3_,Y3_,Z3_] = -AA[X1,Y1,Z1,X2,Y2,Z2,X3,Y3,Z3]/CC[X1,Y1,Z1,X2,Y2,Z2,X3,Y3,Z3]
dZdy[X1_,Y1_,Z1_,X2_,Y2_,Z2_,X3_,Y3_,Z3_] = -BB[X1,Y1,Z1,X2,Y2,Z2,X3,Y3,Z3]/CC[X1,Y1,Z1,X2,Y2,Z2,X3,Y3,Z3]

(* Integral of the square of the gradient over the triangle. *)
INTGRADSQTRI[X1_,Y1_,Z1_,X2_,Y2_,Z2_,X3_,Y3_,Z3_] = AREATRI[X1,Y1,X2,Y2,X3,Y3]\
* (dZdx[X1,Y1,Z1,X2,Y2,Z2,X3,Y3,Z3]^2 + dZdy[X1,Y1,Z1,X2,Y2,Z2,X3,Y3,Z3]^2)

term = FullSimplify[ExpandAll[INTGRADSQTRI[X1,Y1,Z1,X2,Y2,Z2,X3,Y3,Z3]]]

(* We set the location of the center of each corner as the average of the
locations of the nodes.  The value of phi at the center of each corner is
given by the sum of the appropriate value of alpha times the value at the
node of the corner.  The alpha values are computed in the program using
the results of the paper on optimized piecewise linear basis functions. *)

ASWx = (CSWx+MSx+Nx+MWx)/4
ASWy = (CSWy+MSy+Ny+MWy)/4
ASWz = ASW1 CSWz + ASW2 MSz + ASW3 Nz + ASW4 MWz

ASEx = (MSx+CSEx+MEx+Nx)/4
ASEy = (MSy+CSEy+MEy+Ny)/4
ASEz = ASE1 MSz + ASE2 CSEz + ASE3 MEz + ASE4 Nz

ANEx = (Nx+MEx+CNEx+MNx)/4
ANEy = (Ny+MEy+CNEy+MNy)/4
ANEz = ANE1 Nz + ANE2 MEz + ANE3 CNEz + ANE4 MNz

ANWx = (MWx+Nx+MNx+CNWx)/4
ANWy = (MWy+Ny+MNy+CNWy)/4
ANWz = ANW1 MWz + ANW2 Nz + ANW3 MNz + ANW4 CNWz

Term1 = FullSimplify[ExpandAll[DSW INTGRADSQTRI[CSWx,CSWy,CSWz,MSx,MSy,MSz,ASWx,ASWy,ASWz]]]
dTerm1 = FullSimplify[D[Term1,Nz]]
dTerm1c = FullSimplify[Coefficient[dTerm1,Nz,0]]
dTerm1Nz = FullSimplify[Coefficient[dTerm1,Nz,1]]
CForm[ReplaceAll[dTerm1c,x_^2->pow2[x]]] >> dTerms/dTerm1c
CForm[ReplaceAll[dTerm1Nz,x_^2->pow2[x]]] >> dTerms/dTerm1Nz
If[FullSimplify[Expand[dTerm1 - dTerm1c - Nz * dTerm1Nz]] == 0,1,Exit[]]
```

```
Term2 = FullSimplify[ExpandAll[DSW INTGRADSQTRI[MSx,MSy,MSz,Nx,Ny,Nz,ASWx,ASWy,ASWz]]]
dTerm2 = FullSimplify[D[Term2,Nz]]
dTerm2c = FullSimplify[Coefficient[dTerm2,Nz,0]]
dTerm2Nz = FullSimplify[Coefficient[dTerm2,Nz,1]]
CForm[ReplaceAll[dTerm2c,x_^2->pow2[x]]] >> dTerms/dTerm2c
CForm[ReplaceAll[dTerm2Nz,x_^2->pow2[x]]] >> dTerms/dTerm2Nz
If[FullSimplify[Expand[dTerm2 - dTerm2c - Nz * dTerm2Nz]] == 0,1,Exit[]]

Term3 = FullSimplify[ExpandAll[DSW INTGRADSQTRI[Nx,Ny,Nz,MWx,MWy,MWz,ASWx,ASWy,ASWz]]]
dTerm3 = FullSimplify[D[Term3,Nz]]
dTerm3c = FullSimplify[Coefficient[dTerm3,Nz,0]]
dTerm3Nz = FullSimplify[Coefficient[dTerm3,Nz,1]]
CForm[ReplaceAll[dTerm3c,x_^2->pow2[x]]] >> dTerms/dTerm3c
CForm[ReplaceAll[dTerm3Nz,x_^2->pow2[x]]] >> dTerms/dTerm3Nz
If[FullSimplify[Expand[dTerm3 - dTerm3c - Nz * dTerm3Nz]] == 0,1,Exit[]]

Term4 = FullSimplify[ExpandAll[DSW INTGRADSQTRI[MWx,MWy,MWz,CSWx,CSWy,CSWz,ASWx,ASWy,ASWz]]]
dTerm4 = FullSimplify[D[Term4,Nz]]
dTerm4c = FullSimplify[Coefficient[dTerm4,Nz,0]]
dTerm4Nz = FullSimplify[Coefficient[dTerm4,Nz,1]]
CForm[ReplaceAll[dTerm4c,x_^2->pow2[x]]] >> dTerms/dTerm4c
CForm[ReplaceAll[dTerm4Nz,x_^2->pow2[x]]] >> dTerms/dTerm4Nz
If[FullSimplify[Expand[dTerm4 - dTerm4c - Nz * dTerm4Nz]] == 0,1,Exit[]]

Term5 = FullSimplify[ExpandAll[DSE INTGRADSQTRI[MSx,MSy,MSz,CSEx,CSEy,CSEz,ASEx,ASEy,ASEz]]]
dTerm5 = FullSimplify[D[Term5,Nz]]
dTerm5c = FullSimplify[Coefficient[dTerm5,Nz,0]]
dTerm5Nz = FullSimplify[Coefficient[dTerm5,Nz,1]]
CForm[ReplaceAll[dTerm5c,x_^2->pow2[x]]] >> dTerms/dTerm5c
CForm[ReplaceAll[dTerm5Nz,x_^2->pow2[x]]] >> dTerms/dTerm5Nz
If[FullSimplify[Expand[dTerm5 - dTerm5c - Nz * dTerm5Nz]] == 0,1,Exit[]]

Term6 = FullSimplify[ExpandAll[DSE INTGRADSQTRI[CSEx,CSEy,CSEz,MEx,MEy,MEz,ASEx,ASEy,ASEz]]]
dTerm6 = FullSimplify[D[Term6,Nz]]
dTerm6c = FullSimplify[Coefficient[dTerm6,Nz,0]]
dTerm6Nz = FullSimplify[Coefficient[dTerm6,Nz,1]]
CForm[ReplaceAll[dTerm6c,x_^2->pow2[x]]] >> dTerms/dTerm6c
CForm[ReplaceAll[dTerm6Nz,x_^2->pow2[x]]] >> dTerms/dTerm6Nz
If[FullSimplify[Expand[dTerm6 - dTerm6c - Nz * dTerm6Nz]] == 0,1,Exit[]]

Term7 = FullSimplify[ExpandAll[DSE INTGRADSQTRI[MEx,MEy,MEz,Nx,Ny,Nz,ASEx,ASEy,ASEz]]]
dTerm7 = FullSimplify[D[Term7,Nz]]
dTerm7c = FullSimplify[Coefficient[dTerm7,Nz,0]]
dTerm7Nz = FullSimplify[Coefficient[dTerm7,Nz,1]]
CForm[ReplaceAll[dTerm7c,x_^2->pow2[x]]] >> dTerms/dTerm7c
CForm[ReplaceAll[dTerm7Nz,x_^2->pow2[x]]] >> dTerms/dTerm7Nz
If[FullSimplify[Expand[dTerm7 - dTerm7c - Nz * dTerm7Nz]] == 0,1,Exit[]]

Term8 = FullSimplify[ExpandAll[DSE INTGRADSQTRI[Nx,Ny,Nz,MSx,MSy,MSz,ASEx,ASEy,ASEz]]]
dTerm8 = FullSimplify[D[Term8,Nz]]
dTerm8c = FullSimplify[Coefficient[dTerm8,Nz,0]]
dTerm8Nz = FullSimplify[Coefficient[dTerm8,Nz,1]]
CForm[ReplaceAll[dTerm8c,x_^2->pow2[x]]] >> dTerms/dTerm8c
CForm[ReplaceAll[dTerm8Nz,x_^2->pow2[x]]] >> dTerms/dTerm8Nz
If[FullSimplify[Expand[dTerm8 - dTerm8c - Nz * dTerm8Nz]] == 0,1,Exit[]]
```

```
Term9 = FullSimplify[ExpandAll[DNE INTGRADSQTRI[Nx,Ny,Nz,MEx,MEy,MEz,ANEx,ANEy,ANEz]]]
dTerm9 = FullSimplify[D[Term9,Nz]]
dTerm9c = FullSimplify[Coefficient[dTerm9,Nz,0]]
dTerm9Nz = FullSimplify[Coefficient[dTerm9,Nz,1]]
CForm[ReplaceAll[dTerm9c,x_^2->pow2[x]]] >> dTerms/dTerm9c
CForm[ReplaceAll[dTerm9Nz,x_^2->pow2[x]]] >> dTerms/dTerm9Nz
If[FullSimplify[Expand[dTerm9 - dTerm9c - Nz * dTerm9Nz]] == 0,1,Exit[]]


Term10 = FullSimplify[ExpandAll[DNE INTGRADSQTRI[MEx,MEy,MEz,CNEx,CNEy,CNEz,ANEx,ANEy,ANEz]]]
dTerm10 = FullSimplify[D[Term10,Nz]]
dTerm10c = FullSimplify[Coefficient[dTerm10,Nz,0]]
dTerm10Nz = FullSimplify[Coefficient[dTerm10,Nz,1]]
CForm[ReplaceAll[dTerm10c,x_^2->pow2[x]]] >> dTerms/dTerm10c
CForm[ReplaceAll[dTerm10Nz,x_^2->pow2[x]]] >> dTerms/dTerm10Nz
If[FullSimplify[Expand[dTerm10 - dTerm10c - Nz * dTerm10Nz]] == 0,1,Exit[]]

Term11 = FullSimplify[ExpandAll[DNE INTGRADSQTRI[CNEx,CNEy,CNEz,MNx,MNy,MNz,ANEx,ANEy,ANEz]]]
dTerm11 = FullSimplify[D[Term11,Nz]]
dTerm11c = FullSimplify[Coefficient[dTerm11,Nz,0]]
dTerm11Nz = FullSimplify[Coefficient[dTerm11,Nz,1]]
CForm[ReplaceAll[dTerm11c,x_^2->pow2[x]]] >> dTerms/dTerm11c
CForm[ReplaceAll[dTerm11Nz,x_^2->pow2[x]]] >> dTerms/dTerm11Nz
If[FullSimplify[Expand[dTerm11 - dTerm11c - Nz * dTerm11Nz]] == 0,1,Exit[]]

Term12 = FullSimplify[ExpandAll[DNE INTGRADSQTRI[MNx,MNy,MNz,Nx,Ny,Nz,ANEx,ANEy,ANEz]]]
dTerm12 = FullSimplify[D[Term12,Nz]]
dTerm12c = FullSimplify[Coefficient[dTerm12,Nz,0]]
dTerm12Nz = FullSimplify[Coefficient[dTerm12,Nz,1]]
CForm[ReplaceAll[dTerm12c,x_^2->pow2[x]]] >> dTerms/dTerm12c
CForm[ReplaceAll[dTerm12Nz,x_^2->pow2[x]]] >> dTerms/dTerm12Nz
If[FullSimplify[Expand[dTerm12 - dTerm12c - Nz * dTerm12Nz]] == 0,1,Exit[]]

Term13 = FullSimplify[ExpandAll[DNW INTGRADSQTRI[MWx,MWy,MWz,Nx,Ny,Nz,ANWx,ANWy,ANWz]]]
dTerm13 = FullSimplify[D[Term13,Nz]]
dTerm13c = FullSimplify[Coefficient[dTerm13,Nz,0]]
dTerm13Nz = FullSimplify[Coefficient[dTerm13,Nz,1]]
CForm[ReplaceAll[dTerm13c,x_^2->pow2[x]]] >> dTerms/dTerm13c
CForm[ReplaceAll[dTerm13Nz,x_^2->pow2[x]]] >> dTerms/dTerm13Nz
If[FullSimplify[Expand[dTerm13 - dTerm13c - Nz * dTerm13Nz]] == 0,1,Exit[]]

Term14 = FullSimplify[ExpandAll[DNW INTGRADSQTRI[Nx,Ny,Nz,MNx,MNy,MNz,ANWx,ANWy,ANWz]]]
dTerm14 = FullSimplify[D[Term14,Nz]]
dTerm14c = FullSimplify[Coefficient[dTerm14,Nz,0]]
dTerm14Nz = FullSimplify[Coefficient[dTerm14,Nz,1]]
CForm[ReplaceAll[dTerm14c,x_^2->pow2[x]]] >> dTerms/dTerm14c
CForm[ReplaceAll[dTerm14Nz,x_^2->pow2[x]]] >> dTerms/dTerm14Nz
If[FullSimplify[Expand[dTerm14 - dTerm14c - Nz * dTerm14Nz]] == 0,1,Exit[]]

Term15 = FullSimplify[ExpandAll[DNW INTGRADSQTRI[MNx,MNy,MNz,CNWx,CNWy,CNWz,ANWx,ANWy,ANWz]]]
dTerm15 = FullSimplify[D[Term15,Nz]]
dTerm15c = FullSimplify[Coefficient[dTerm15,Nz,0]]
dTerm15Nz = FullSimplify[Coefficient[dTerm15,Nz,1]]
CForm[ReplaceAll[dTerm15c,x_^2->pow2[x]]] >> dTerms/dTerm15c
CForm[ReplaceAll[dTerm15Nz,x_^2->pow2[x]]] >> dTerms/dTerm15Nz
If[FullSimplify[Expand[dTerm15 - dTerm15c - Nz * dTerm15Nz]] == 0,1,Exit[]]
```

```
Term16 = FullSimplify[ExpandAll[DNW INTGRADSQTRI[CNWx,CNWy,CNWz,MWx,MWy,MWz,ANWx,ANWy,ANWz]]]
dTerm16 = FullSimplify[D[Term16,Nz]]
dTerm16c = FullSimplify[Coefficient[dTerm16,Nz,0]]
dTerm16Nz = FullSimplify[Coefficient[dTerm16,Nz,1]]
CForm[ReplaceAll[dTerm16c,x_^2->pow2[x]]] >> dTerms/dTerm16c
CForm[ReplaceAll[dTerm16Nz,x_^2->pow2[x]]] >> dTerms/dTerm16Nz
If[FullSimplify[Expand[dTerm16 - dTerm16c - Nz * dTerm16Nz]] == 0,1,Exit[]]
```

References

[1] T.C. Luu, E.D. Brooks III, and A. Szoke, "Generalized reference fields and source interpolation for the difference formulation of radiation transport," J. Comp. Phys. 229 (2010) 1626-1642.

[2] B. A. Finlayson, "Variational Principles for Heat Transfer," in: Numerical Properties and Methodologies in Heat Transfer, ed. T. M. Shih, Hemisphere Publishing Corporation, New York, ISBN 0-89116-309-3, 1983, pp 17-31.

[3] D. E. Burton, "Conservation of Energy, Momentum, and Angular Momentum in Lagrangian Staggered-Grid Hydrodynamics," Lawrence Livermore National Laboratory, UCRL-JC-105926 (1991).

[4] M. L. Adams, "A new transport discretization scheme for arbitrary spatial meshes in XY geometry," Lawrence Livermore National Laboratory, UCRL-JC-105974 (1991).

[5] E.D. Brooks III and A. Szoke, "Optimal Piecewise Linear Basis Functions in Two Dimensions," Lawrence Livermore National Laboratory, LLNL-TR-410412, 2009, https://library-ext.llnl.gov.

[6] H. G. Stone and M. L. Adams, "A Piecewise Linear Finite Element Basis with Application to Particle Transport," Transactions of American Nuclear Society Winter Meeting, Washington, D.C., November 17-21, 2002, Vol. 87, pp. 130-133 (2002).

[7] S. Wolfram, "The Mathematica Book," 5th ed., Wolfram Media, 2003.