A Second-Order Cell-Centered Diffusion Difference Scheme for Unstructured Hexahedral Lagrangian Meshes (U)

Michael L. Hall and Jim E. Morel
Group CIC-19, Radiation Transport Team
Los Alamos National Laboratory
Email: hall@lanl.gov

A discretization method for the diffusion equation in 3-D has been developed. The method is valid for unstructured meshes with cell-centered data. The homogeneous solution of the diffusion equation, which is linear, is preserved exactly. The method is second order accurate and conserves energy locally. Material discontinuities are handled rigorously. In the case of an orthogonal mesh, the method reduces to the standard seven-point operator. The discretization scheme results in an unsymmetric matrix with a size of roughly four times the number of cells. This matrix system can be solved using any sparse unsymmetric matrix solver. (U)

Introduction

The accurate solution of the diffusion equation is important for many varied applications. For instance, diffusion equations occur in the modeling of heat conduction, in certain formulations of fluid flow, and in radiation transport. Within the discipline of radiation transport, diffusion equations are used in single-group P_1 and Simplified Spherical Harmonics (SP_N) calculations, and in Diffusion Synthetic Acceleration (DSA) of transport iterations.

The mesh on which a diffusion problem is to be solved is often dictated by other problem constraints, such as the need to perform a Lagrangian hydrodynamics calculation in conjunction with a diffusion calculation. A mesh type which seems to be gaining prominence in three-dimensional modeling is the unstructured hexahedral mesh. This mesh consists of hexahedra and degenerate hexahedra (prisms, pyramids, tetrahedra, see Figure 1) that are connected in an arbitrary fashion. The connectivity of such a mesh must be explicitly specified. The additional complication of an unstructured mesh is balanced by the freedom to model arbitrary geometries, such as block structured meshes, and to model curved geometries with fewer distorted cells.

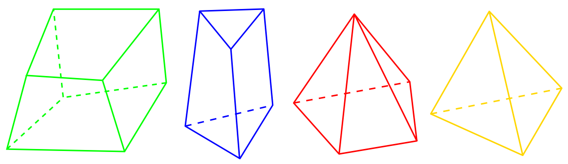

Figure 1: Cell types handled by the model in 3-D: hexahedron, prism, pyramid and tetrahedron.

This paper develops a numerical method for modeling diffusion on unstructured hexahedral meshes. The method is an extension of the method described in an earlier paper by the authors (Morel, J. E. Dendy, Hall and White, 1992). The method in the previous paper was specifically for 2-D structured (i.e. logically rectangular) meshes. The derivation given here is applicable to 1-D, 2-D, and 3-D unstructured meshes.

Method Overview

The equation to be solved is given by

$$\alpha \frac{\partial \Phi}{\partial t} - \overrightarrow{\nabla} \cdot D \overrightarrow{\nabla} \Phi + \sigma \Phi = S - \overrightarrow{\nabla} \cdot \overrightarrow{J} \,, \tag{1}$$

which can be written

$$\alpha \frac{\partial \Phi}{\partial t} + \overrightarrow{\nabla} \cdot \overrightarrow{F} + \sigma \Phi = S \tag{2}$$

$$\overrightarrow{F} = -D \overrightarrow{\nabla} \Phi + \overrightarrow{J} \,, \tag{3}$$

where

Φ	=	Intensity
$\overrightarrow{F}$	=	Flux
D	=	Diffusion Coefficient
α	=	Time Derivative Coefficient
σ	=	Removal Coefficient
S	=	Intensity Source Term
$\overrightarrow{J}$	=	Flux Source Term.

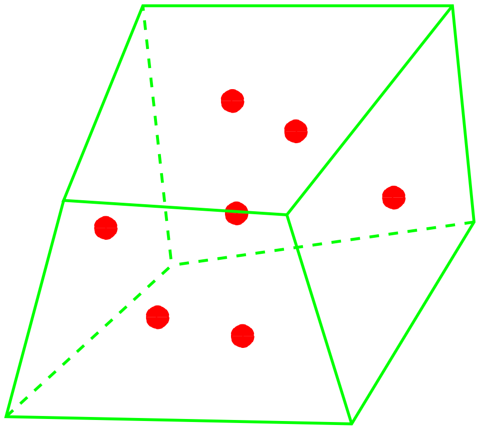

Figure 2: Location of the unknowns on the hexahedron: one at the cell center and one on each cell face.

Everything in this equation is assumed to be known, with the exception of Φ and $\overrightarrow{F}$ at the new time step. This equation has an extra term from the standard diffusion equation, the $\overrightarrow{\nabla} \cdot \overrightarrow{J}$ term, which allows it to model the P_1 and SP_N equations.

The new method shares these properties with the method described in Morel et al. (1992):

- It is cell-centered (balance equations are done over a cell).
- The method has cell-centered and face-centered unknowns, which are required to rigorously treat material discontinuities (see Figure 2).
- Homogeneous solutions (in this case, linear solutions) are preserved exactly.
- It is second-order accurate.
- The method reduces to the standard cell-centered operator (seven-point for 3-D, five-point for 2-D, three-point for 1-D) for an orthogonal mesh.
- Local energy conservation is maintained.
- Unfortunately, the method results in an unsymmetric matrix system.

In addition, the new method handles unstructured meshes and is multi-dimensional. The geometries that can be handled by the new method are listed in Table 1.

Table 1: The geometries that can be handled by the new method, all of which have an unstructured (arbitrarily connected) format.

Dimension	Geometries	Type of Elements
1-D	spherical, cylindrical or cartesian	line segments
2-D	cylindrical or cartesian	quadrilaterals or triangles
3-D	cartesian	hexahedra or degenerate hexahedra (tetrahedra, prisms, pyramids)

Method Derivation

Conservation Equation

The first step in the derivation of the method is the discretization of the conservation equation, Equation 2. Integrating the conservation equation over the cell volume gives

$$\int_{V_c} \alpha \frac{\partial \Phi}{\partial t}\, dV + \int_{V_c} \overrightarrow{\nabla} \cdot \overrightarrow{F}\, dV + \int_{V_c} \sigma \Phi\, dV = \int_{V_c} S\, dV \,. \tag{4}$$

Defining cell averages and applying Gauss' Theorem gives

$$\alpha_c \frac{\partial \Phi_c}{\partial t} V_c + \int_A \overrightarrow{F} \cdot \overrightarrow{dA} + \sigma_c \Phi_c V_c = S_c V_c \,. \tag{5}$$

Discretizing temporally and evaluating the flux integral gives

$$\frac{\alpha_c V_c}{\Delta t} \left(\Phi_c^{n+1} - \Phi_c^n \right) + \sum_f \overrightarrow{F_f^{n+1}} \cdot \overrightarrow{A_f} + \sigma_c \Phi_c^{n+1} V_c = S_c V_c \,, \tag{6}$$

where the sum over f represents the sum over the face values, the subscript c represents a cell-centered or cell-average value and the unsuperscripted variables are evaluated at $n + \frac{1}{2}$. Note that all of the geometries in Table 1 can be represented by this equation when the proper definitions for areas, volumes, and face sums are substituted. Note that this scheme is not limited to fully implicit differencing; for instance, a Crank-Nicholson differencing scheme may be employed.

Flux Terms

All of the terms in Equation 6 are evaluated in terms of known quantities, or in terms of the unknown Φ, except for the flux term. To effect closure, the flux vector across each face of the cell, $\overrightarrow{F_f^{n+1}}$, must be expressed in terms of Φ^{n+1}.

Evaluating the flux equation, Equation 3, at a particular face gives

$$\overrightarrow{F_f^{n+1}} = -D_{c,f}\,\overrightarrow{\nabla}\,\Phi^{n+1} + \overrightarrow{J_f}\ . \tag{7}$$

The flux source, $\overrightarrow{J_f}$, is known. The diffusion coefficient, $D_{c,f}$, is known within a cell, but may be discontinuous at the cell face. In general, one cannot evaluate a gradient by taking differences across a material interface because the gradient is discontinuous along the interface. It is often assumed that one can take differences across a material discontinuity if one uses a proper average of the two diffusion coefficients on each side of the discontinuity. However, it will be shown that this is possible only on orthogonal meshes. To avoid taking differences across material discontinuities, an intensity unknown is added to each cell face. The interface flux is then evaluated using two independent differences, with each difference being constructed solely from unknowns from a single cell. An equation for each interface intensity is obtained by requiring these independent differences to yield the same flux.

Actual expressions for the gradient in non-orthogonal coordinate systems will now be considered. The values of Φ at four non-planar points are necessary and sufficient to determine the gradient. Since the mesh is unstructured, a unique coordinate system will be defined for each cell. Any four non-planar points ($\overrightarrow{P_1}, \overrightarrow{P_2}, \overrightarrow{P_3}, \overrightarrow{P_4}$) define a local coordinate system (see Figure 3) in terms of three vectors,

$$\begin{aligned} \hat{k} &\equiv \overrightarrow{P_2} - \overrightarrow{P_1}\ , \\ \hat{l} &\equiv \overrightarrow{P_3} - \overrightarrow{P_1}\ , \\ \hat{m} &\equiv \overrightarrow{P_4} - \overrightarrow{P_1}\ . \end{aligned} \tag{8}$$

The coordinates of the four points in (k, l, m)-space are defined to be $\overrightarrow{\mathcal{P}_1} \equiv (0,0,0)^T$, $\overrightarrow{\mathcal{P}_2} \equiv (1,0,0)^T$,

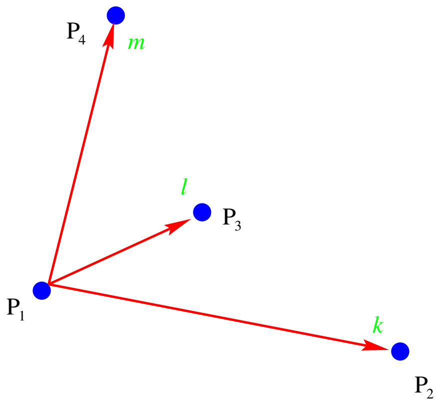

Figure 3: Coordinate system defined by four non-planar points.

$\overrightarrow{\mathcal{P}_3} \equiv (0,1,0)^T$, and $\overrightarrow{\mathcal{P}_4} \equiv (0,0,1)^T$. A Jacobian matrix converts between the (k, l, m) coordinate system and the (x, y, z) coordinate system:

$$\begin{bmatrix} P^x \\ P^y \\ P^z \end{bmatrix} - \begin{bmatrix} P_1^x \\ P_1^y \\ P_1^z \end{bmatrix} = \begin{bmatrix} \frac{\partial x}{\partial k} & \frac{\partial x}{\partial l} & \frac{\partial x}{\partial m} \\ \frac{\partial y}{\partial k} & \frac{\partial y}{\partial l} & \frac{\partial y}{\partial m} \\ \frac{\partial z}{\partial k} & \frac{\partial z}{\partial l} & \frac{\partial z}{\partial m} \end{bmatrix} \begin{bmatrix} \mathcal{P}^k \\ \mathcal{P}^l \\ \mathcal{P}^m \end{bmatrix}. \tag{9}$$

which is represented as:

$$\overrightarrow{P} - \overrightarrow{P}_1 = \mathbf{J}\ \overrightarrow{\mathcal{P}}\ . \tag{10}$$

Note that an equally valid inverse transformation from the (x, y, z) coordinate system to the (k, l, m) coordinate system could have been used, with a Jacobian matrix equal to $\mathbf{J}^{-1}$. However, since the four points are located along the axes in (k, l, m)-space, but not in (x, y, z)-space, it is easier to take the derivatives needed for the forward Jacobian than the reverse Jacobian:

$$\begin{aligned} \mathbf{J} &= \left[\left(\overrightarrow{P_2} - \overrightarrow{P_1}\right) \ \left(\overrightarrow{P_3} - \overrightarrow{P_1}\right) \ \left(\overrightarrow{P_4} - \overrightarrow{P_1}\right) \right] \\ &= \left[\ \hat{k} \quad \hat{l} \quad \hat{m} \ \right]. \end{aligned} \tag{11}$$

Returning to the consideration of the gradient term and expanding the k, l and m derivatives of Φ using the chain rule yields

$$\begin{bmatrix} \frac{\partial \Phi}{\partial k} \\ \frac{\partial \Phi}{\partial l} \\ \frac{\partial \Phi}{\partial m} \end{bmatrix} = \begin{bmatrix} \frac{\partial x}{\partial k} & \frac{\partial y}{\partial k} & \frac{\partial z}{\partial k} \\ \frac{\partial x}{\partial l} & \frac{\partial y}{\partial l} & \frac{\partial z}{\partial l} \\ \frac{\partial x}{\partial m} & \frac{\partial y}{\partial m} & \frac{\partial z}{\partial m} \end{bmatrix} \begin{bmatrix} \frac{\partial \Phi}{\partial x} \\ \frac{\partial \Phi}{\partial y} \\ \frac{\partial \Phi}{\partial z} \end{bmatrix}$$

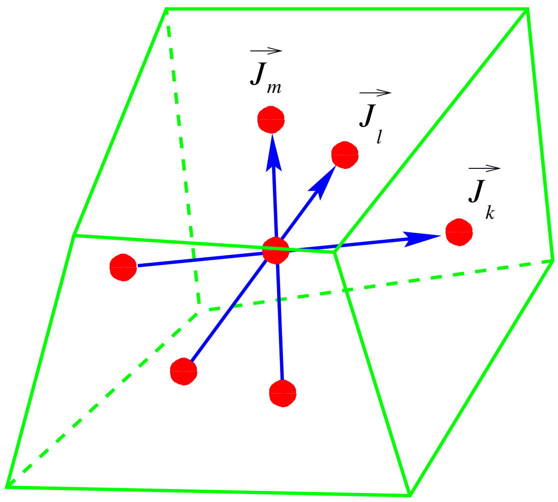

Figure 4: Coordinate system defined by three intersecting lines.

$$= \quad \mathbf{J}^T \overrightarrow{\nabla} \Phi \, , \tag{12}$$

or, solving for $\overrightarrow{\nabla} \Phi$ and inserting the derivative definitions,

$$\overrightarrow{\nabla} \Phi = \mathbf{J}^{-T} \begin{bmatrix} \frac{\partial \Phi}{\partial k} \\ \frac{\partial \Phi}{\partial l} \\ \frac{\partial \Phi}{\partial m} \end{bmatrix} \approx \mathbf{J}^{-T} \begin{bmatrix} \Phi_2 - \Phi_1 \\ \Phi_3 - \Phi_1 \\ \Phi_4 - \Phi_1 \end{bmatrix} . \tag{13}$$

This method of representing gradients is exact for linear functions, but only approximate for higher order functions.

Four points are not the only way to determine a gradient, for instance, six points that form three lines intersecting in a single point can also be used. A six points gradient is actually used to determine the fluxes on the faces of the cell. If a point (and therefore an unknown Φ) is placed in the center of each face, the three lines formed by connecting opposing faces all intersect at the cell center. A single Jacobian matrix per cell is then sufficient to determine the necessary gradients.

If the vectors connecting the face centers of opposite faces are denoted $\overrightarrow{J_k}$, $\overrightarrow{J_l}$, and $\overrightarrow{J_m}$ for the k, l, and m directions (see Figure 4), then the Jacobian matrix is given by

$$\mathbf{J} = \left[\begin{array}{ccc} \overrightarrow{J_k} & \overrightarrow{J_l} & \overrightarrow{J_m} \end{array} \right] , \tag{14}$$

and the inverse transpose matrix is

$$\mathbf{J}^{-T} = \frac{1}{|\mathbf{J}|} \left[\begin{array}{ccc} \left(\overrightarrow{J_l} \times \overrightarrow{J_m} \right) & \left(\overrightarrow{J_m} \times \overrightarrow{J_k} \right) & \left(\overrightarrow{J_k} \times \overrightarrow{J_l} \right) \end{array} \right] . \tag{15}$$

The values for the k, l and m derivatives of Φ have yet to be defined. These are defined in terms of the seven unknown Φ's in each cell. If the origin is placed at the center of the cell, then the locations of the unknown Φ's in (k, l, m)-space are

$$\begin{aligned} \Phi_c &\Rightarrow (0,0,0) \, , \\ \Phi_{+k} &\Rightarrow (\frac{1}{2},0,0) \, , \\ \Phi_{-k} &\Rightarrow (-\frac{1}{2},0,0) \, , \\ \Phi_{+l} &\Rightarrow (0,\frac{1}{2},0) \, , \\ \Phi_{-l} &\Rightarrow (0,-\frac{1}{2},0) \, , \\ \Phi_{+m} &\Rightarrow (0,0,\frac{1}{2}) \, , \text{ and} \\ \Phi_{-m} &\Rightarrow (0,0,-\frac{1}{2}) \, , \end{aligned} \tag{16}$$

where the faces are denoted $+k$, $-k$, $+l$, $-l$, $+m$ and $-m$. Along any particular direction there are three ways to determine a derivative. For example, in the k direction, the k derivative of Φ can be determined via

$$\begin{aligned} \frac{\partial \Phi}{\partial k} &= \frac{\Phi_{+k} - \Phi_{-k}}{\frac{1}{2} - \left(-\frac{1}{2}\right)} = \Phi_{+k} - \Phi_{-k} \, , \\ \frac{\partial \Phi}{\partial k} &= \frac{\Phi_{+k} - \Phi_c}{\frac{1}{2} - 0} = 2\left(\Phi_{+k} - \Phi_c\right) \, , \text{ or} \\ \frac{\partial \Phi}{\partial k} &= \frac{\Phi_c - \Phi_{-k}}{0 - \left(-\frac{1}{2}\right)} = 2\left(\Phi_c - \Phi_{-k}\right) \, . \end{aligned} \tag{17}$$

The first definition uses a full-cell difference, whereas the second and third definitions use a half-cell difference. Each of these definitions will be used depending on which direction and which face is being considered.

The gradient must be determined for each face of the cell. Each face must have definitions for all of the k, l and m derivatives. For a given face, the direction which is perpendicular to the face is called the major direction, because it is the only direction for which the derivative is non-zero if the cell is orthogonal, and it is usually the main contributor to the flux across the face. The other directions are called the minor directions for that face.

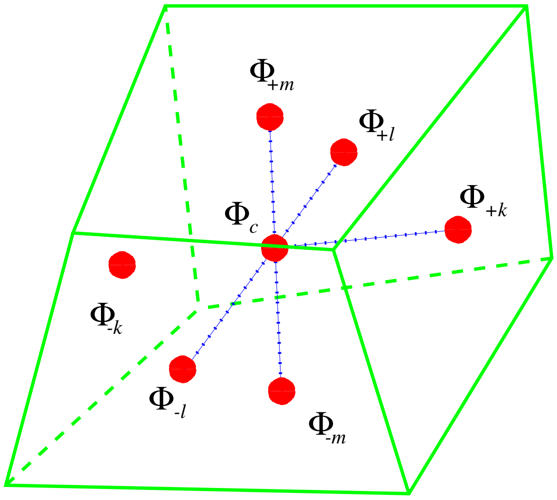

Figure 5: Stencil for the $+k$ face gradient, shown with a dotted blue line.

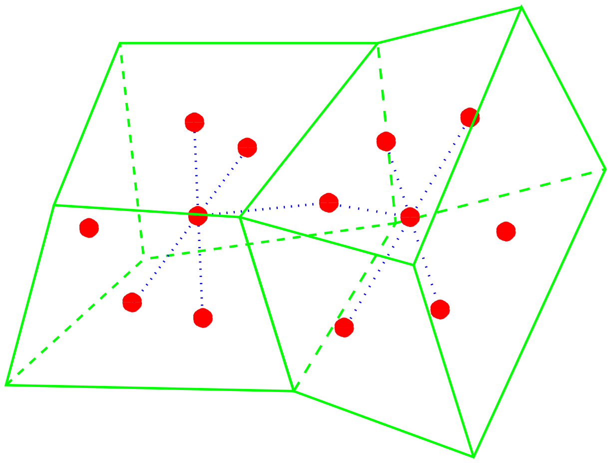

Figure 6: Stencil for the face equation, shown with a dotted blue line.

To compute the gradient for a face, full-cell derivatives are used for the minor directions. The half-cell derivative which involves the face in question is used for the major direction. For example, for the $+k$ face the major direction is the k direction, and the minor directions are the l and m directions. The gradient for the $+k$ face is represented by the cell value for the $\mathbf{J}^{-T}$ matrix multiplied by the k, l and m derivative vector for that face:

$$\overrightarrow{F_{+k}^{n+1}} = -D_{c,+k}\,\mathbf{J}_c^{-T}\begin{bmatrix} 2\left(\Phi_{+k}^{n+1}-\Phi_c^{n+1}\right) \\ \Phi_{+l}^{n+1}-\Phi_{-l}^{n+1} \\ \Phi_{+m}^{n+1}-\Phi_{-m}^{n+1}\end{bmatrix} + \overrightarrow{J_{c,+k}} \tag{18}$$

Figure 5 shows the stencil for each face gradient that is given by this method. This completes the discretization of the conservation equation, Equation 2. When all of the face gradients have been defined and substituted in, the conservation equation has terms for all of the seven unknowns within the cell and has no terms involving any unknowns that are not within the cell.

Cell Face Equations

After discretizing the conservation equation, there is an equation for every cell in the problem, but roughly three extra unknowns per cell have been added. Closure is achieved by applying a continuity of flux condition at each cell face:

$$-\overrightarrow{F_{c1,f}^{n+1}}\cdot\overrightarrow{A_{c1,f}} - \overrightarrow{F_{c2,f}^{n+1}}\cdot\overrightarrow{A_{c2,f}} = 0 \tag{19}$$

where $c1$ and $c2$ are the two cells that share the face f. If, for example, face f is a $-m$ face in $c1$ and is a $+k$ face in $c2$, this equation can be written out as

$$\begin{aligned} & D_{c1,-m}\,\mathbf{J}_{c1}^{-T}\begin{bmatrix} \Phi_{+k}^{n+1}-\Phi_{-k}^{n+1} \\ \Phi_{+l}^{n+1}-\Phi_{-l}^{n+1} \\ 2\left(\Phi_{c1}^{n+1}-\Phi_{-m}^{n+1}\right)\end{bmatrix}\cdot\overrightarrow{A_{c1,-m}} \\ & + D_{c2,+k}\,\mathbf{J}_{c2}^{-T}\begin{bmatrix} 2\left(\Phi_{+k}^{n+1}-\Phi_{c2}^{n+1}\right) \\ \Phi_{+l}^{n+1}-\Phi_{-l}^{n+1} \\ \Phi_{+m}^{n+1}-\Phi_{-m}^{n+1}\end{bmatrix}\cdot\overrightarrow{A_{c2,+k}} \\ & - \overrightarrow{J_{c1,-m}}\cdot\overrightarrow{A_{c1,-m}} - \overrightarrow{J_{c2,+k}}\cdot\overrightarrow{A_{c2,+k}} = 0\,. \end{aligned} \tag{20}$$

The 11-point stencil for the cell face equations is shown in Figure 6.

Boundary Conditions

Because the conservation equation only involves variables within a cell, the boundary conditions only affect the cell face equations. On the boundaries, instead of a cell face equation, a Robin boundary condition is specified,

$$\beta_1\,\Phi_f^{n+1} - \beta_2\,\overrightarrow{F_{c,f}^{n+1}}\cdot\overrightarrow{A_{c,f}} = \beta_3\,\Phi_{bc}\,, \tag{21}$$

where β_1, β_2 and β_3 can be specified to match the following common boundary conditions:

- homogeneous: $\Phi_f^{n+1} = 0$,
- reflective: $-\overrightarrow{F_{c,f}^{n+1}} \cdot \hat{n}_{c,f} = 0$,
- vacuum: $\frac{1}{2}\Phi_f^{n+1} - \overrightarrow{F_{c,f}^{n+1}} \cdot \hat{n}_{c,f} = 0$,
- Dirichlet: $\Phi_f^{n+1} = \Phi_{bc}$,
- Neumann: $-\overrightarrow{F_{c,f}^{n+1}} \cdot \hat{n}_{c,f} = -\Phi_{bc}$, or
- source boundary conditions: $\frac{1}{2}\Phi_f^{n+1} - \overrightarrow{F_{c,f}^{n+1}} \cdot \hat{n}_{c,f} = \frac{1}{2}\Phi_{bc}$.

Orthogonal Reduction

In the case of an orthogonal mesh, the face area vector for each face is parallel with the major direction for that face, and perpendicular to the minor directions for that face. The dot product of the flux vector with the area vector for a given face is therefore only related to the flux in the major direction and not the minor directions.

There is no loss of generality in assuming that the k, l, and m directions are aligned with the x, y, and z directions. With this assumption, the $\mathbf{J}^{-T}$ matrix reduces to the following:

$$\mathbf{J}^{-T} = \frac{1}{J_k^x J_l^y J_m^z} \begin{bmatrix} J_l^y J_m^z & 0 & 0 \\ 0 & J_k^x J_m^z & 0 \\ 0 & 0 & J_k^x J_l^y \end{bmatrix} . \quad (22)$$

The flux-area dot product for each face then becomes directly proportional to the difference in intensity values between the face and the cell center. For example, the flux dot product for the $+k$ face in the case of an orthogonal mesh is given by[1]

$$\overrightarrow{F_{c1,+k}^{n+1}} \cdot \overrightarrow{A_{c1,+k}} \quad (23)$$

$$= -D_{c1,+k}\, \mathbf{J}_{c1}^{-T} \begin{bmatrix} \frac{\partial \Phi}{\partial k} \\ \frac{\partial \Phi}{\partial l} \\ \frac{\partial \Phi}{\partial m} \end{bmatrix} \cdot \begin{bmatrix} A_{+k}^x \\ 0 \\ 0 \end{bmatrix}$$

$$= -\frac{D_{c1,+k}}{J_k^x J_l^y J_m^z} \begin{bmatrix} 2J_l^y J_m^z \left(\Phi_{+k}^{n+1} - \Phi_{c1}^{n+1}\right) \\ J_k^x J_m^z \left(\Phi_{+l}^{n+1} - \Phi_{-l}^{n+1}\right) \\ J_k^x J_l^y \left(\Phi_{+m}^{n+1} - \Phi_{-m}^{n+1}\right) \end{bmatrix} \cdot \begin{bmatrix} A_{+k}^x \\ 0 \\ 0 \end{bmatrix}$$

$$= -\frac{D_{c1,+k} A_{c1,+k}^x}{J_{c1,k}^x/2} \left(\Phi_{+k}^{n+1} - \Phi_{c1}^{n+1}\right) .$$

Similarly, for the $-k$ face of cell $c2$,

$$\overrightarrow{F_{c2,-k}^{n+1}} \cdot \overrightarrow{A_{c2,-k}} = -\frac{D_{c2,-k} A_{c2,-k}^x}{J_{c2,k}^x/2} \left(\Phi_{c2}^{n+1} - \Phi_{-k}^{n+1}\right) . \quad (24)$$

If cell $c1$ and cell $c2$ share the $+k/-k$ face, then these two equations may be substituted into the cell face equation, Equation 19. Noting that $\overrightarrow{A_{c2,-k}} = -\overrightarrow{A_{c1,-k}}$,

$$\frac{D_{c1,+k}}{J_{c1,k}^x/2} \left(\Phi_{+k}^{n+1} - \Phi_{c1}^{n+1}\right) = \frac{D_{c2,-k}}{J_{c2,k}^x/2} \left(\Phi_{c2}^{n+1} - \Phi_{+k}^{n+1}\right) . \quad (25)$$

This equation can be solved for Φ_{+k}^{n+1} and then substituted back into Equation 23 to yield an equation for the flux which only involves the cell center unknowns:

$$\overrightarrow{F_{c1,+k}^{n+1}} \cdot \overrightarrow{A_{c1,+k}} = \quad (26)$$
$$-A_{c1,+k}^x \left[\frac{J_{c1,k}^x/2}{D_{c1,+k}} + \frac{J_{c2,k}^x/2}{D_{c2,-k}}\right]^{-1} \left(\Phi_{c2}^{n+1} - \Phi_{c1}^{n+1}\right) .$$

This equation shows that the effective diffusion coefficient on an orthogonal mesh is

$$D_{\mathit{eff},f} = \left[\frac{\Delta x_{c1}}{D_{c1,f}} + \frac{\Delta x_{c2}}{D_{c2,f}}\right]^{-1} (\Delta x_{c1} + \Delta x_{c2}) , \quad (27)$$

where J_k^x has been denoted with the more familiar form of Δx. The flux expression then becomes

$$\overrightarrow{F_{c1,+k}^{n+1}} \cdot \overrightarrow{A_{c1,+k}} = \quad (28)$$
$$-A_{c1,+k}^x \frac{D_{\mathit{eff},f}}{\Delta x_{c1} + \Delta x_{c2}} \left(\Phi_{c2}^{n+1} - \Phi_{c1}^{n+1}\right) ,$$

which agrees exactly with the standard seven-point diffusion operator. It is only possible to reduce the flux expression to the cell center variables, using an effective diffusion coefficient to represent the material discontinuity, if the mesh is orthogonal. Otherwise, the gradients on each side of the interface must be represented separately, and the equation set cannot be reduced to a cell center difference using local operations.

[1] The standard seven-point diffusion operator does not account for a flux source term, $\overrightarrow{J}$, so that is omitted from the current discussion.

Algebraic Solution

The discretization scheme results in an algebraic system with $(4n_c + n_b/2)$ unknowns, where n_c is the number of cells in the problems and n_b is the number of boundary faces.[2] This is roughly four times the number of unknowns that a method like the standard seven-point orthogonal operator produces, but the new method cannot be expressed in terms of only the cell center variables without obtaining a dense diffusion matrix. There are no methods which use only cell center unknowns that are second-order accurate on skewed meshes. The matrix system for this method is unsymmetric, which necessitates an unsymmetric solver. There is a maximum of 11 non-zero elements in any row, due to the cell face equation stencil.

When a Krylov subspace solver is used, a specialized preconditioning system has been developed to speed convergence. The preconditioner consists of a low-order version of the new method itself, which is derived by setting the minor direction terms for each face to zero (see section entitled "Flux Terms"). This results in a matrix which can be reduced to a system involving the cell center unknowns only. In addition to having four times fewer unknowns, this system has a maximum of seven non-zero elements per row and is symmetric. After solution with a symmetric solver (which tends to be faster than an unsymmetric solver), the face unknowns are obtained via back-substitution.

The Krylov subspace solvers used for the solution of the main system are GMRES and TFQMR, among others. The low-order preconditioning system is solved using the Conjugate Gradients method, which is in turn preconditioned using SSOR. Alternately, an incomplete direct method called the unstructured multi-frontal method may be used. This solver gives very exact answers rapidly for small systems, but ultimately loses with the respect to the Krylov solvers when large time-dependent problems with loose tolerances are solved.

[2] In general, for multiple dimensions the scheme has $((n_d + 1)\, n_c + n_b/2)$ unknowns, where n_d is the number of dimensions.

Implementation: The Augustus Code Package

The method outlined in this paper has been implemented into a code package called Augustus. It is currently written in Fortran-77, but it will be re-written in Fortran-90 in the future. Current platforms (Operating Systems) include Sun (SunOS and Solaris), SGI (IRIX), HP (HP-UX), and IBM (AIX). On-line documentation can be found at `http://www.lanl.gov/Augustus`.

The Augustus code package is a true multi-dimensional package that can model all of the geometries described in Table 1. The package has been installed into the ALEGRA (SNLA) hydrodynamics code (Summers et al., 1996; Budge et al., 1994; Robinson et al., 1996), and the Telluride (LANL) low-speed flow code (Kothe et al., 1997; Reddy et al., 1997). Augustus is also used as the diffusion kernel for the Spartan SP_N radiation transport code, developed by the authors.

The Augustus code package uses two algebraic solver packages. The main package, used for Krylov subspace methods, is JTpack, developed by John Turner at LANL (Turner, 1997). The auxiliary package, used for the unstructured multi-frontal method, is UMFPACK (Davis and Duff, 1995).

The Augustus code package has been completed, and there is active development of new features. The package may be obtained by contacting the authors.

Results

Second-Order Demonstration

In order to demonstrate that the method is second-order accurate, a problem with an analytic quartic[3] solution is solved. The problem which is chosen is described in detail in Morel et al. (1992). The problem domain is a cube, with a random mesh obtained by moving (in 3-D) the interior points of an orthogonal mesh by a random fraction of 20% of the inter-nodal distance, in a random direction. There are reflective boundaries on four sides, and vacuum boundaries on two opposite sides. The properties

[3] The method is exact for linear solutions, so no improvement is achieved by refining the mesh in that situation.

Table 2: Results from the Second-Order Accuracy Test.

New Method

Problem Size (cells)	$\frac{\Vert\Phi_{\text{exact}}-\Phi\Vert_2}{\Vert\Phi_{\text{exact}}\Vert_2}$	Error Ratio
$5 \times 5 \times 5$	$1.0248{\times}10^{-2}$	
$10 \times 10 \times 10$	$2.6190{\times}10^{-3}$	3.91
$20 \times 20 \times 20$	$6.6082{\times}10^{-4}$	3.96
$40 \times 40 \times 40$	$1.6530{\times}10^{-4}$	4.00

Orthogonal Seven-Point Solution

Problem Size (cells)	$\frac{\Vert\Phi_{\text{exact}}-\Phi\Vert_2}{\Vert\Phi_{\text{exact}}\Vert_2}$	Error Ratio
$5 \times 5 \times 5$	$1.0202{\times}10^{-2}$	
$10 \times 10 \times 10$	$2.6205{\times}10^{-3}$	3.92
$20 \times 20 \times 20$	$6.5952{\times}10^{-4}$	3.97
$40 \times 40 \times 40$	$1.6515{\times}10^{-4}$	3.99

are constant spatially and temporally, and there is a spatially-varying source which is proportional to x^2 in each cell. With these conditions, the steady-state analytic answer is a quartic of the form $\Phi(x, y, z) = a + bx + cx^4$.

The results from running this problem for different mesh sizes are given in Table 2. It can be seen that the error is reduced by a factor of four each time the mesh spacing is reduced by a factor of two, which indicates a second-order accurate method. The results from the orthogonal seven-point operator (on an orthogonal mesh) show similar behavior.

Homogeneous Solution Problem

The method will exactly preserve the homogeneous solution to the diffusion equation, which is a linear solution, even if the mesh is highly skewed. To show this, a problem with a linear solution has been solved. The problem domain is a cube, with reflective boundaries on four sides and source and vacuum boundary conditions on opposite sides. The physical properties are constant spatially and temporally, and

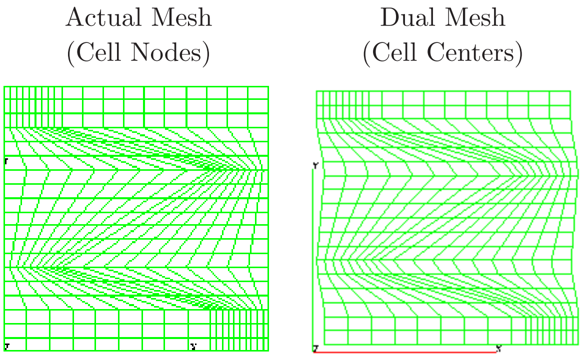

Figure 7: Differences between the actual mesh and the dual mesh that is used by the plotting package, shown on one face of the cube.

there are no removal or source terms. The steady-state analytic solution is linear in one dimension.

The mesh for this problem is $20{\times}20{\times}20$, which results in 8000 nodes, 6859 cells, and 28519 unknowns. The mesh spacing is one that has been developed by the authors and is termed a "3-D Kershaw" mesh. The basis of this mesh is a 2-D mesh that was described in Kershaw (1981), which had constant spacing in one dimension and varied spacing in the second dimension. The 3-D Kershaw mesh has constant spacing in one dimension and varied spacing in the second and third dimensions, which creates a mesh that is very skewed in 3-D.

Figures 8, 9, and 10 were made using GMV, a program written by Frank Ortega at LANL (Ortega, 1995). Unfortunately, this program is best suited for node-centered data, rather than cell-centered data. This problem was partially circumvented by treating the cell centers as node centers in a dual mesh (see Figure 7). Due to the skewed nature of the mesh, the cell centers are not flush with the edges of the cube and give the illusion of a wavy cube boundary, which is not the case.

Before running the 3-D Kershaw mesh problem, the same problem was run on an orthogonal mesh. A contour plot of the steady-state results is shown in Figure 8. The analytical solution is linear in x, and the method reproduces this exactly, as is seen from the straight contour lines. This was expected because the method reduces to the standard seven-point operator in the case of an orthogonal mesh.

Figure 9 shows a contour plot of the steady-state results for the 3-D Kershaw mesh. The contour lines

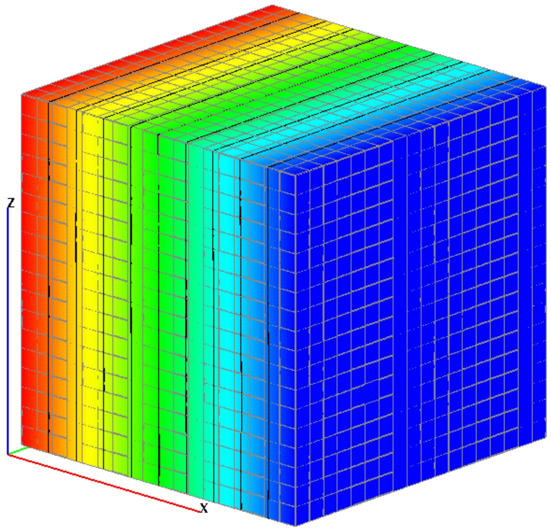

Figure 8: Contour plot of the steady-state solution to the homogeneous problem on an orthogonal mesh.

remain linear, even though the mesh is highly skewed. A random cutplane through the cube (see Figure 10) shows that the contour lines are linear on the interior of the cube and highlights the skewed nature of the mesh. Indeed, calculations exhibit linearity of the solution down to machine precision.

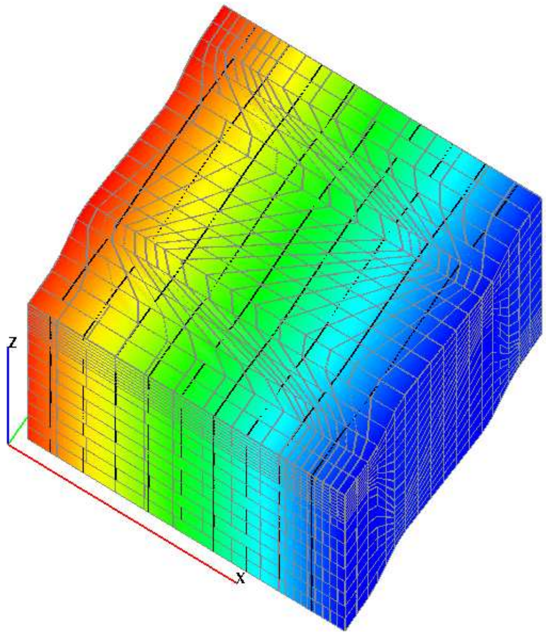

Figure 9: Contour plot of the steady-state solution to the homogeneous problem on a 3-D Kershaw mesh.

Summary

A discretization method for the diffusion equation in 3-D has been developed. The method is valid for unstructured meshes with cell-centered data. The homogeneous solution of the diffusion equation, which is linear, is preserved exactly. The method is second order accurate and conserves energy locally. Material discontinuities are handled rigorously. In the case of an orthogonal mesh, the method reduces to the standard seven-point operator.

The discretization scheme results in an unsymmetric matrix with a size of roughly four times the number of cells. This matrix system can be solved using any sparse unsymmetric matrix solver.

A code package named Augustus has been developed to implement the method. This package models all of the geometries listed in Table 1. The matrix is solved using either Krylov subspace methods or an unstructured multi-frontal method.

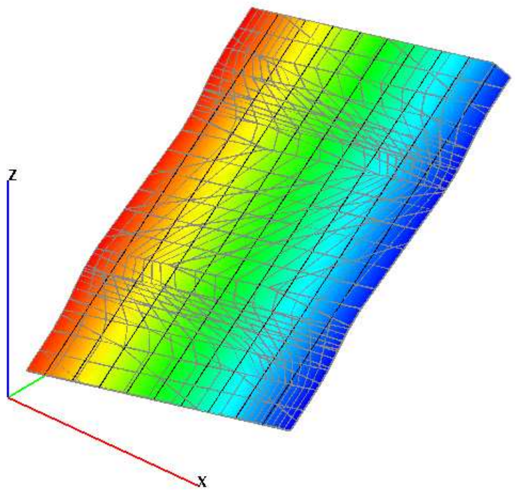

Figure 10: Contour plot on a random cutplane of the steady-state solution to the homogeneous problem on a 3-D Kershaw mesh.

Future and Concurrent Work

The authors have also developed the SPARTAN code package, which models the SP_N radiation transport equations using the AUGUSTUS code package as a diffusion kernel. It also models all of the geometries listed in Table 1.

A symmetric diffusion discretization method based on Morel et al. (1992) has been developed, but unfortunately it is only applicable to 2-D cartesian geometries. A support operator diffusion discretization method has also been developed. This method gives a symmetric matrix and looks promising for multiple geometries.

References

Budge, K. G., J. S. Peery, M. K. Wong and T. G. Trucano (1994), RAYHYD: An ICF Target Simulation Code Written in C++ (U), *in* "Proceedings of the Nuclear Explosive Code Developers Conference", October 1994. Online information on ALEGRA is available at `http://sherpa.sandia.gov/9231home/alegra/`.

Davis, Timothy A. and Iain S. Duff (1995), "A Combined Unifrontal/Multifrontal Method for Unsymmetric Sparse Matrices", *submitted to ACM Transactions on Mathematical Software*, 1995. TR-95-020. Online information on UMFPACK is available at `http://www.cis.ufl.edu/~davis/`.

Kershaw, David S. (1981), "Differencing of the Diffusion Equation in Lagrangian Hydrodynamic Codes", *Journal of Computational Physics* **39**:375–395, 1981.

Kothe, D. B., R. C. Ferrell, J. A. Turner and S. J. Mosso (1997), A High Resolution Finite Volume Method for Efficient Parallel Simulation of Casting Processes on Unstructured Meshes, *in* "Proceedings of the 8th SIAM Conference on Parallel Processing for Scientific Computing", Minneapolis, MN, March 14–17 1997. LA-UR–97-30. Online information on TELLURIDE is available at `http://www.lanl.gov/telluride/`.

Morel, J. E., Jr. J. E. Dendy, Michael L. Hall and Stephen W. White (1992), "A Cell-Centered Lagrangian-Mesh Diffusion Differencing Scheme", *Journal of Computational Physics* **103**(2):286–299, December 1992. LA-UR–90-3582.

Ortega, Frank A. (1995), General Mesh Viewer (GMV) User's Manual, Technical Report LA-UR–95-2986, Los Alamos National Laboratory. Online information on GMV is available at `http://www-xdiv.lanl.gov/XCM/gmv/GMVHome.html`.

Reddy, A. V., D. B. Kothe, C. Beckermann, R. C. Ferrell and K. L. Lam (1997), High Resolution Finite Volume Parallel Simulations of Mold Filling and Binary Alloy Solidification on Unstructured 3-D Meshes, *in* "Proceedings of SP97: The Fourth Decennial International Conference on Solidification Processing", The University of Sheffield, UK, July 7–10 1997. LA-UR–97-136. Online information on TELLURIDE is available at `http://www.lanl.gov/telluride/`.

Robinson, A. C., A. V. Farnsworth, S. T. Montgomery, J. S. Peery and K. O. Merewether (1996), Neutron Generator Power Supply Modeling in EMMA, *in* "Proceedings of the Ninth Nuclear Explosives Code Developer's Conference", San Diego, CA, October 22–25 1996. Online information on ALEGRA is available at `http://sherpa.sandia.gov/9231home/alegra/`.

Summers, R. M., J. S. Peery, M. W. Wong, E. S. Hertel, T. G. Trucano and L. C. Chhabildas (1996), "Recent Progress in ALEGRA Development and Application to Ballistic Impacts", *to be published in International Journal of Impact Engineering* **20**, 1996. Presented at the 1996 Hypervelocity Impact Symposium, Freiburg, Germany, Oct. 7–10, 1996. Online information on ALEGRA is available at `http://sherpa.sandia.gov/9231home/alegra/`.

Turner, John A. (1997), JTPACK User's Manual, Technical Report LA-UR–97-2, Los Alamos National Laboratory. Online information on JTPACK is available at `http://www.turner-family.com/John/LANL/JTpack.html`.