ERDC Major Shared Resource Center

MSRC

ERDC MSRC/PET TR/00-04

Implementation of Adaptive Mesh Refinement into an Eulerian Hydrocode

by

David Littlefield
J. Tinsley Oden

11h00299

Work funded by the DoD High Performance Computing Modernization Program CEWES Major Shared Resource Center through

Programming Environment andTraining (PET)

Supported by Contract Number: DAHC94-96-C0002
Nichols Research Corporation

Views, opinions and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of Defense Position, policy, or decision unless so designated by other official documentation.

Implementation of Adaptive Mesh Refinement into an Eulerian Hydrocode

David L. Littlefield
J. Tinsley Oden
Texas Institute for Computational and Applied Mathematics
The University of Texas at Austin
Austin TX 78712

Abstract

In this report the implementation of an adaptive mesh refinement scheme into an existing three-dimensional Eulerian hydrocode is described. The adaptive strategy is block-based, which was required in order to leave the existing data structure intact. The focus of this work is on the development of refinement and derefinement procedures that are conservative and preserve the locations of material interfaces, as well as error indicators suitable for use in an Eulerian hydrocode.

Introduction

Adaptive mesh refinement refers to a scheme for finite difference and finite element codes wherein the size and distribution of the computational mesh is changed dynamically so that the solution complies with some specific constraint. There are many different types of constraints of interest, depending on the goals of the computation, but a common constraint is that the error be held constant over the entire computational mesh. There are many advantages to adaptive refinement schemes; most importantly is that problems are solved in the most computational time- and memory-efficient manner.

The benefits of adaptive mesh refinement have already been demonstrated in many application areas (for example, linear elasticity, gas dynamics, and acoustics), but adaptive refinement for shock wave physics codes is still a new area of research. Eulerian codes in particular may benefit significantly from adaptive mesh refinement since, typically, it is necessary to include a large number of elements in a simulation, even in regions where initially there may not be any material.

In this report, the implementation of an adaptive mesh refinement scheme into an existing three-dimensional Eulerian hydrocode is described. The code can be characterized for present purposes as being based on a discretization of the field equations for a media with multiple materials and material interfaces, approximated on a Cartesian grid, a structure typified by the well-known CTH code [1]. The adaptive strategy is block-based, which was required in order to leave the existing data structure intact. The focus of this work is on error indicators suitable for use in an Eulerian hydrocode, as well as refinement and derefinement procedures.

Overview of Adaptive Strategy

When implementing adaptive refinement into an existing code, it is very important to consider the organization and data structure of the target application code. In the present case, the data is organized in (*I,J,K*) logical blocks that correspond to the mesh used in the problem, as is shown in Figure 1. Within a block, the mesh contour lines must remain parallel to the coordinate axes and constrained nodes are not permitted. However, different values of *I, J,* and/or *K* are permitted in adjacent blocks. Thus a reasonable approach for implementation of adaptivity, which preserved the existing data structure in the code, was to limit refinement to the block level. Furthermore, in order to simplify the refinement process as well as communication between blocks, the refinement/derefinement was limited to isotropic 2:1 ratios along adjacent blocks. This is illustrated in Figure 2, where a set of communicating blocks is shown. Ghost cells are incorporated along the edges of the block, and the contents of these cells combine or split as is needed from the adjacent block. In this way, each block *sees* exactly the information it expects to see.

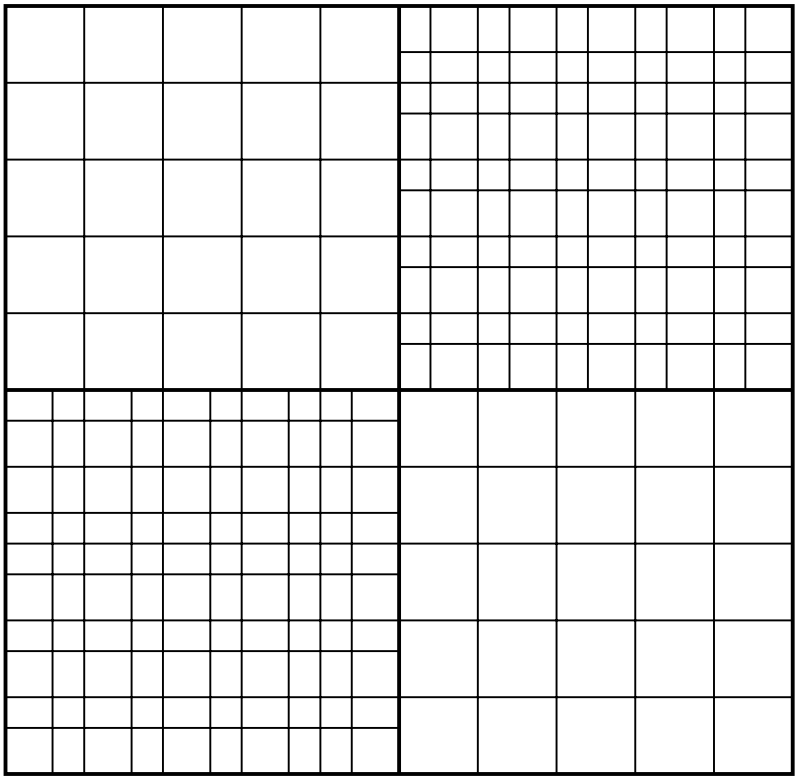

Figure 1. Organization of data in target application code

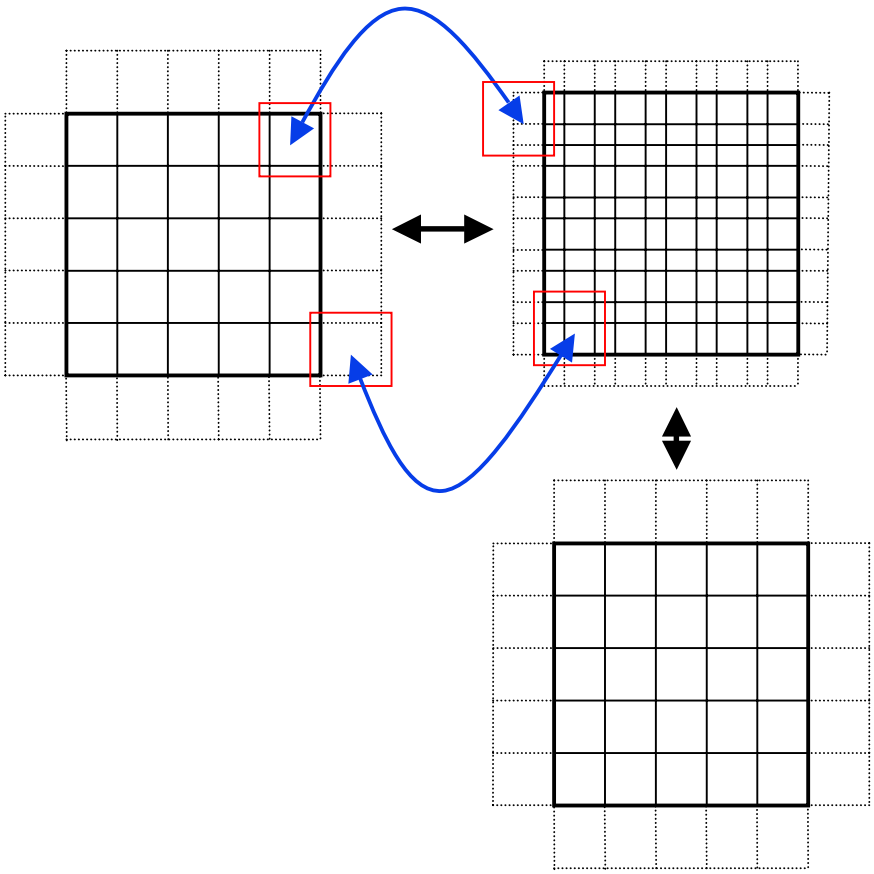

Figure 2. Block-adaptive strategy applied to target application code.

A significant part of this effort is to establish the two-way communication between blocks, as well as to make the scheme work in parallel, which is the subject of another paper [2]. The focus of this work is on the development of refinement and derefinement schemes, as well as error indicators, suitable for implementation into a three-dimensional Eulerian shock physics code.

Refinement and Derefinement

Efficient and accurate schemes for refinement and derefinement of the cell variables are a crucial element in any adaptive scheme. The refinement and derefinement strategies used here take advantage of the 1:2 and 2:1 ratios between parent and child cells, allowing these processes to be carried out rapidly and in a conservative manner.

Derefinement Procedure

The collapse of a set of eight child cells into a single parent cell is a simple process. A method for this was implemented by Crawford [2] and this scheme was used

in the present investigation. The scheme conserves mass and energy; momentum is also conserved if the velocity profiles are linear across the parent cell.

The parent cell material masses are found from the sum of the child cell material masses, i.e.

$$M_{mp} = \sum_{i=1}^{8} M_{mi}, \quad m = 1,2,....n \tag{1}$$

where M_{mp} and M_{mi} denote masses of material m in the parent and ith child cells, respectively. The parent cell total mass is the sum of the parent material masses. Values for the intensive cell material energies for the parent are determined by taking a mass average of the value for the children as

$$E_{mp} = \frac{1}{M_{mp}} \sum_{i=1}^{8} M_{mi} E_{mi}, \tag{2}$$

where E_{mp} and E_{mi} denote energies for material m in the parent and ith child cells, respectively. The parent cell total energy is then calculated as

$$E_{tp} = \frac{1}{M_p} \sum_{m=1}^{n} E_{mp} M_{mp}, \quad m = 1,2,....n \tag{3}$$

where M_p is the total mass of the parent cell, E_{mp} is the intensive cell energy for material m and E_{tp} is the intensive total cell energy for the parent cell. Currently, values for other intensive variables that are not material intensive variables (such as internal state variables) in the parent are taken from child cell 1 (refer to Figure 3).

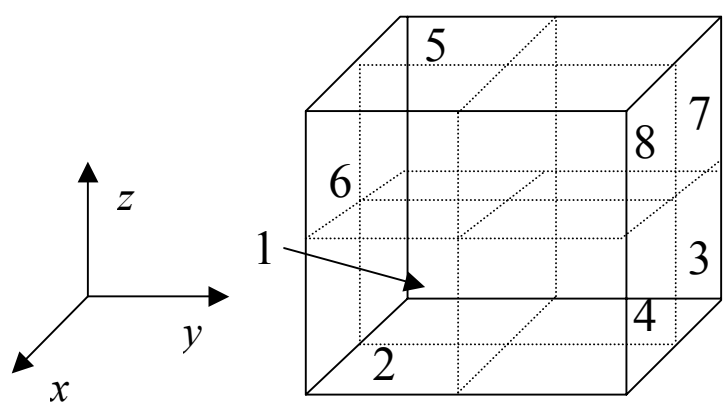

Figure 3. Orientation of child cells in the parent cell.

Material and cell pressures, temperatures, stress deviators, and artificial viscosities for the parent cell are also taken for child cell 1. Clearly this is not uniformly valid and will be modified at a later date. For example, material and cell pressures could be determined from the equation-of-state and mixed-cell thermodynamics models once the material masses and energies have been determined for the children.

The velocities in the application code are face-centered rather than cell-centered. The velocities along the face of the parents are taken as the average of the four faces of the children that share that face. For example, the x-velocity on the left face of the parent cell shown in Figure 3 is the average x-velocities from child cells 1, 3, 5 and 7. This process conserves momentum if the velocities vary linearly across the parent cell, but in general momentum conservation is not guaranteed.

Refinement Procedure

The refinement of a parent cell into eight child cells, on the other hand, is complicated by the fact that there may be material interfaces in the parent cell. The interfaces must reconstructed to properly map the cell variables to the children. There are already algorithms in place in the target application code that perform rezoning that could have been used for this; however, these algorithms rely in one-dimensional advection and may be too dispersive given the large number of times the refinement routines are used during a typical calculation. Instead, we can take advantage of the exact geometric overlaps that exist when the refinement is limited to 2 to 1 ratios, and reconstruct the material interfaces in the child cells exactly. This eliminates dispersion errors.

Review of Youngs' Method for Interface Reconstruction

A key element in the refinement process is the proper mapping of material interfaces when elements are refined. In order to understand this process, it is useful to review Youngs' algorithm for interface reconstruction [3]. The method is a systematic approach for the determination of a unique planar interface separating two materials in a cell, given the volume fractions of the materials in the cell. Conceptually, the method is simple to understand. Where it provides the greatest benefit is by minimizing the number of possible intersection conditions that must be checked when a plane of arbitrary orientation passes through a cell (there are only five when this method is applied).

The basic strategy in the Youngs' algorithm is to determine the outward unit normal vector $\boldsymbol{n}$ separating the material of interest from the other materials, and the distance d from the interface plane to a reference corner, measured along a direction parallel to $\boldsymbol{n}$. If there are only two materials in the cell and the interface plane is assumed to be planar, these two quantities uniquely define the location of the interface plane.

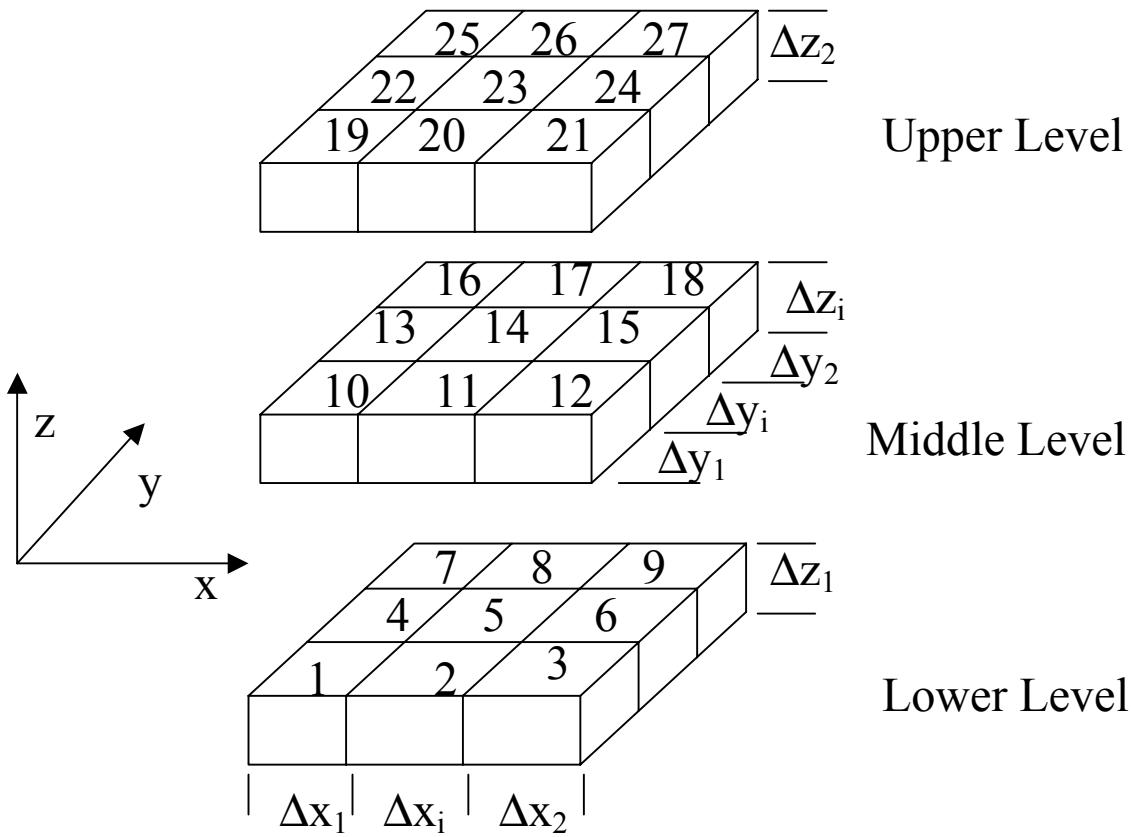

Figure 4. Schematic of a central cell (Cell 14) and its 26 surrounding cells.

The first step in the method is to determine the direction of $\boldsymbol{n}$. This is done by normalizing the direction for the maximum rate of change of the volume fraction, i.e.

$$\boldsymbol{n} = -\frac{\nabla\phi}{|\nabla\phi|}, \tag{4}$$

where ϕ is the volume fraction of the material of interest. Difference approximations are used to determine the gradient of the volume fraction. Since the direction of the normal must be independent of the coordinate system selected, a symmetric difference approximation must be used. Consider a cell and its 26 neighboring cells as is depicted in Figure 4. A symmetric difference approximation for $\partial\phi/\partial x$ in cell 14 can be expressed as

$$\frac{\partial\phi}{\partial x} = \frac{\phi_e - \phi_w}{2\Delta x_i}, \tag{5}$$

where ϕ_e and ϕ_w are determined as

$$\begin{aligned}\phi_e = [&\Delta\zeta_1(\Delta\xi_1\phi_3 + \Delta\xi_2\phi_9) + \Delta\zeta_2(\Delta\xi_1\phi_{21} + \Delta\xi_2\phi_{27}) + \\ &2(\Delta\xi_1\phi_{12} + \Delta\xi_2\phi_{18} + \Delta\zeta_1\phi_6 + \Delta\zeta_2\phi_{24}) + \\ &4\phi_{15}]/[(2 + \Delta\xi_1 + \Delta\xi_2)(2 + \Delta\zeta_1 + \Delta\zeta_2)],\end{aligned} \tag{6}$$

$$\begin{aligned}\phi_w = [&\Delta\zeta_1(\Delta\xi_1\phi_1 + \Delta\xi_2\phi_7) + \Delta\zeta_2(\Delta\xi_1\phi_{19} + \Delta\xi_2\phi_{25}) + \\ &2(\Delta\xi_1\phi_{10} + \Delta\xi_2\phi_{16} + \Delta\zeta_1\phi_4 + \Delta\zeta_2\phi_{22}) + \\ &4\phi_{13}]/[(2 + \Delta\xi_1 + \Delta\xi_2)(2 + \Delta\zeta_1 + \Delta\zeta_2)],\end{aligned} \tag{7}$$

and $\Delta\xi$ and $\Delta\zeta$ are dimensionless cell dimensions defined as

$$\Delta\xi_j = \frac{\Delta y_j}{\Delta y_i}, \qquad \Delta\zeta_j = \frac{\Delta z_j}{\Delta z_i}. \tag{8}$$

The approximation for ϕ_e, given in Eq. (6), is formed from unit cell linear interpolation of values for the total material volume (that is, the cell volume multiplied by the volume fraction) for each of the edges of cell 15 parallel to the *x*-axis, evaluated at the center of cell 15. Likewise, values for the total material volume at each of these edges is determined from unit cell linear interpolation for each of the cells sharing that edge (for example, one of these edges is formed from cells 3, 6, 12, and 15). An equivalent approximation is used to determine ϕ_w given in Eq. (7). Similar symmetric difference expressions can be also defined for the other two derivatives $\partial\phi/\partial y$ and $\partial\phi/\partial z$.

Once the unit normal vector has been defined, the interface plane can be located and the value for *d* can be found. In order to minimize the number of possible intersections that must be considered, it is useful to apply some restrictions to the calculation. First, all calculations are made with respect to a unit cube. Second, the interface determination is limited to volume fractions less than ½. If the volume fraction is greater than ½, then the interface is located based on the values of 1 - ϕ (i.e. the volume fraction for the other material) for the cell of interest and its 26 neighboring cells. Third, the interface determination is made with respect to a specific corner of the unit cube, and at a specific orientation. The corner and orientation are determined as follows. The absolute values of the components of ***n*** are ordered from smallest to largest. Let these values be designated at n_1, n_2 and n_3. Then, the interface determination is made in the 1-2-3 coordinate system having directions corresponding to the directions of n_1, n_2 and n_3. A series of axis sign changes (i.e. –*x* for *x*, -*y* for *y*, etc.) and/or axis swaps (i.e. *x* for *y*, *z* for *x*, etc.) will also transform the *x*-*y*-*z* coordinate system to the 1-2-3 coordinate system.

After applying these three restrictions, only one of five possible intersection conditions is possible. These are given in Figure 5, and include the triangle section,

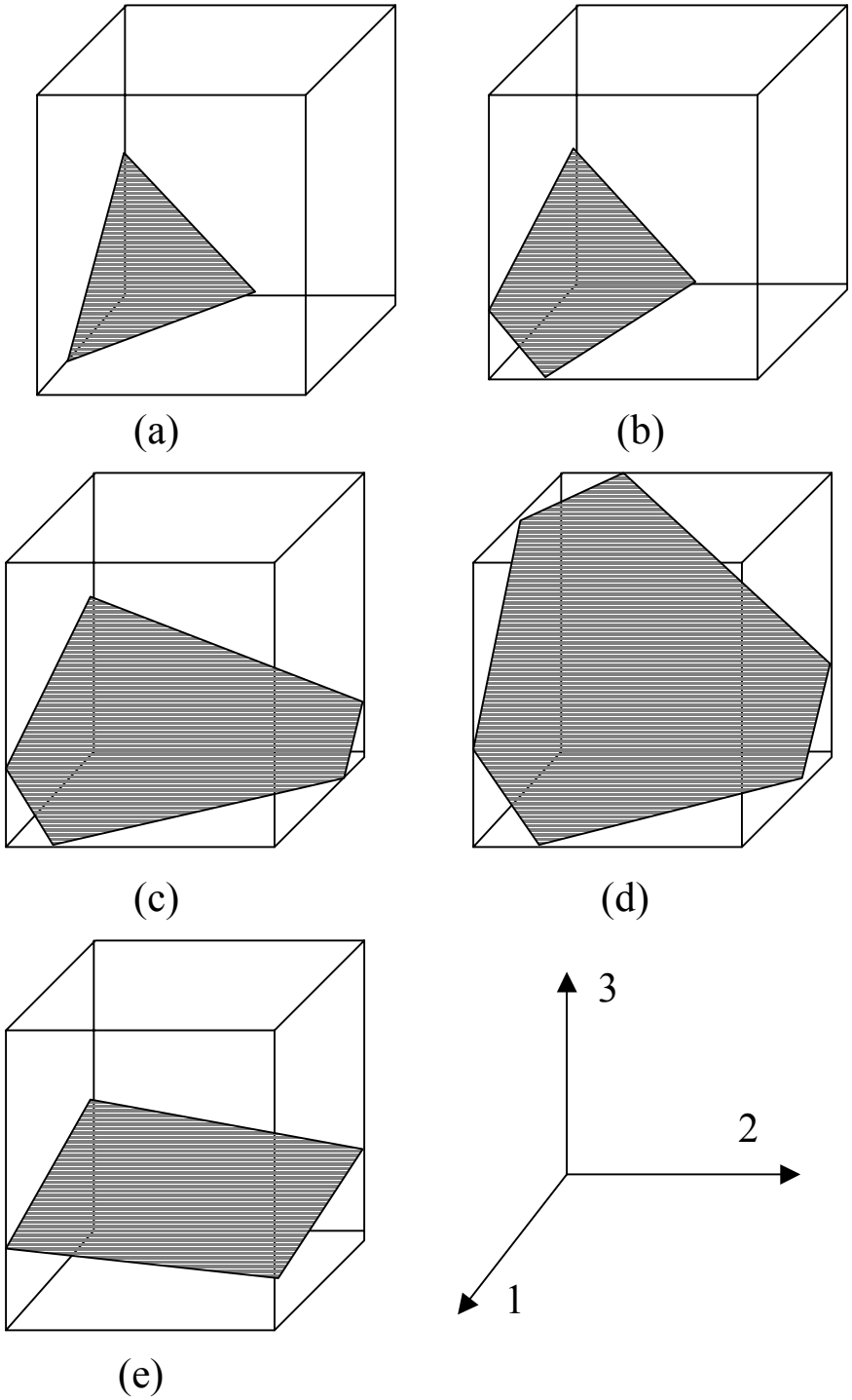

Figure 5. Possible intersection conditions for a plane intersecting a unit cube: (a) triangle section, (b) quadrilateral section A, (c) pentagonal section, (d) hexagonal section, and (e) quadrilateral section B.

quadrilateral section A, pentagonal section, hexagonal section, and the quadrilateral section B. Depending on the relative values of n_1, n_2, n_3 and ϕ, only one of these five intersection types can be produced. From this comes the interface geometry as well as a value for d. The applicable ranges for each of the interface geometries and the corresponding values for d are as follows:

Triangle section: this situation occurs for

$$6n_1n_2n_3\phi < n_1^3 \tag{9a}$$

and gives

$$d = \left(6\phi n_1n_2n_3\right)^{1/3}. \tag{9b}$$

Quadrilateral section A: this situation occurs for

$$n_1^3 \le 6n_1n_2n_3\phi < n_2^3 - \left(n_2 - n_1\right)^3 \tag{10a}$$

and gives

$$d = \frac{n_1}{2} + \left[2n_1n_2\phi - \frac{n_1^2}{12}\right]^{1/2}. \tag{10b}$$

Pentagonal section: this situation occurs for

$$n_2^3 - \left(n_2 - n_1\right)^3 \le 6n_1n_2n_3\phi < n_3^3 - \left(n_3 - n_1\right)^3 - \left(n_3 - n_2\right)^3 \quad \text{and} \quad n_1 + n_2 > n_3$$

or

$$n_2^3 - \left(n_2 - n_1\right)^3 \le 6n_1n_2n_3\phi < \left(n_1 + n_2\right)^3 - n_2^3 - n_1^3 \quad \text{and} \quad n_1 + n_2 \le n_3 \tag{11a}$$

and gives

$$d^3 + \left(d - n_1\right)^3 - \left(d - n_2\right)^3 - 6n_1n_2n_3\phi = 0. \tag{11b}$$

Hexagonal section: this situation occurs for

$$n_3^3 - \left(n_3 - n_1\right)^3 - \left(n_3 - n_2\right)^3 \le 6n_1n_2n_3\phi \quad \text{and} \quad n_1 + n_2 > n_3 \tag{12a}$$

and gives

$$d^3 + \left(d - n_1\right)^3 - \left(d - n_2\right)^3 - \left(d - n_3\right)^3 - 6n_1n_2n_3\phi = 0. \tag{12b}$$

Quadrilateral section B: this situation occurs for

$$(n_1 + n_2)^3 - n_2^3 - n_1^3 \le 6n_1n_2n_3\phi \quad \text{and} \quad n_1 + n_2 > n_3 \tag{13a}$$

and gives

$$d = \phi n_3 + \frac{1}{2}(n_1 + n_2). \tag{13b}$$

Note that in the case of the pentagonal and hexagonal sections, a cubic must be solved to determine d.

Extension of Youngs' Method for Cell Refinement

Once d and $\boldsymbol{n}$ for the parent cell are known, refinement into eight equal volume child cells can be accomplished in a manner that preserves the interface reconstruction in the children. The basic procedure is to compute d and $\boldsymbol{n}$ for each of the children, locate the position of the interface plane then determine the value for the volume fraction corresponding to this plane location.

Consider the refinement of a parent cell into eight children, in the 1-2-3 coordinate system, as is depicted in Figure 6.

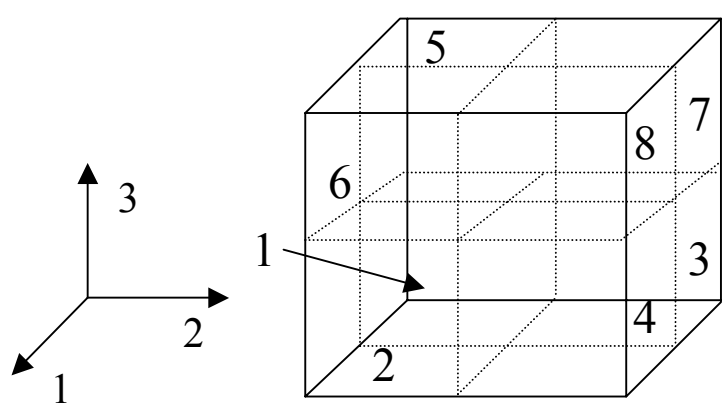

Figure 6. Schematic of refinement of a parent cell into eight equal volume child cells.

The unit normal vector to the interface plane is the same for the children as it is for the parent cell. The corner distances d_i for each of the children are easily derived and are given by

$$
\begin{aligned}
d_1 &= 2d, \\
d_2 &= 2d - n_1, \\
d_3 &= 2d - n_2, \\
d_4 &= 2d - (n_1 + n_2), \\
d_5 &= 2d - n_3, \\
d_6 &= 2d - (n_1 + n_3), \\
d_7 &= 2d - (n_2 + n_3), \\
d_8 &= 2d - (n_1 + n_2 + n_3).
\end{aligned}
\tag{14}
$$

Note that the values of d_i for the children are based values for a unit cell. Now, given the value for d and $\boldsymbol{n}$ in each of the child cells, the interface can be reconstructed and the required volume fraction can be computed. To do this, we need to apply some restrictions. First, the reference corner orientation for each child cell is set equal to the orientation for the parent cell (unless it needs to be reversed – see the third restriction). Second, a check must be made to see if any of the child cells are completely full or empty. For $d_i < 0$, the child cell is empty and the volume fraction is set to zero, and for $d_i > n_1 + n_2 + n_3$, the child cell is completely full and the volume fraction is set to one. No interface reconstruction is done in either of these situations. Third, if $d_i > n_1$, $d_i > n_2$ and $d_i > n_3$ or if $d_i > n_1$, $d_i > n_2$ and $d_i > n_1 + n_2$ then the reference corner is changed to the opposite one along the main diagonal and the volume fraction of the other material (i.e. 1 - ϕ for the material of interest) is computed. To do this, all that is required is the value of d_i be set to $n_1 + n_2 + n_3 - d_i$.

After applying these restrictions, the number of possible intersections that must be considered is again reduced to the five possibilities given in Figure 5. From these comes the value for ϕ_i for each of the children. The applicable ranges for each of the interface geometries, as well as the corresponding values for ϕ_i, are as follows:

Triangle intersection: this situation occurs for

$$d_i \le n_1 \tag{15a}$$

and gives:

$$\phi_i = d_i^3 / (6 n_1 n_2 n_3). \tag{15b}$$

Quadrilateral section A: this situation occurs for

$$d_i \le n_2 \tag{16a}$$

and gives:

$$\phi_i = \frac{1}{6n_1n_2n_3}\left[d_i^3 - (d_i - n_1)^3\right]. \tag{16b}$$

Pentagonal section: this situation occurs for

$$d_i \le n_3 \quad \text{and} \quad d_i \ge n_1 + n_2 \tag{17a}$$

and gives:

$$\phi_i = \frac{1}{6n_1n_2n_3}\left[d_i^3 - (d_i - n_1)^3 - (d_i - n_2)^3\right]. \tag{17b}$$

Hexagonal section: this situation occurs for

$$d_i > n_3 \quad \text{and} \quad d_i \ge n_1 + n_2 \tag{18a}$$

and gives

$$\phi_i = \frac{1}{6n_1n_2n_3}\left[d_i^3 - (d_i - n_1)^3 - (d_i - n_2)^3 - (d_i - n_3)^3\right]. \tag{18b}$$

Quadrilateral section B: this situation occurs for

$$d_i \le n_3 \quad \text{and} \quad d_i < n_1 + n_2 \tag{19a}$$

and gives

$$\phi_i = \frac{1}{n_3}\left[d_i - (n_1 + n_2)\right]. \tag{19b}$$

Material Ordering

An implicit assumption in the interface reconstruction is that there is only one interface separating two materials in a cell. Reconstruction of multiple planar interfaces separating materials (for example, a crack tip) is not modeled using this reconstruction technique. As such, when the method is extended for multi-material cells with n materials (including voids), it must be assumed that there are $n - 1$ interfaces. Then, in order to properly locate the interfaces, the material ordering must be prescribed. Youngs' original work required that the user specify the material ordering. This was later extended by Bell [3] who implemented a scheme for automatic ordering. However, Bell's method requires the advection direction to be specified, and in the case of refinement there is no identifiable advection direction.

Using an idea similar in spirit to the one used by Bell, an automatic material ordering scheme is implemented into the cell refinement algorithm. The ordering scheme outlined is applied to the parent cell in the refinement process. Consider a cell with three materials as is shown in Figure 7. For the particular situation shown, it is clear that for the center cell the best choice for material ordering is 2-1-3. An algorithm that embodies this decision process can be described as follows:

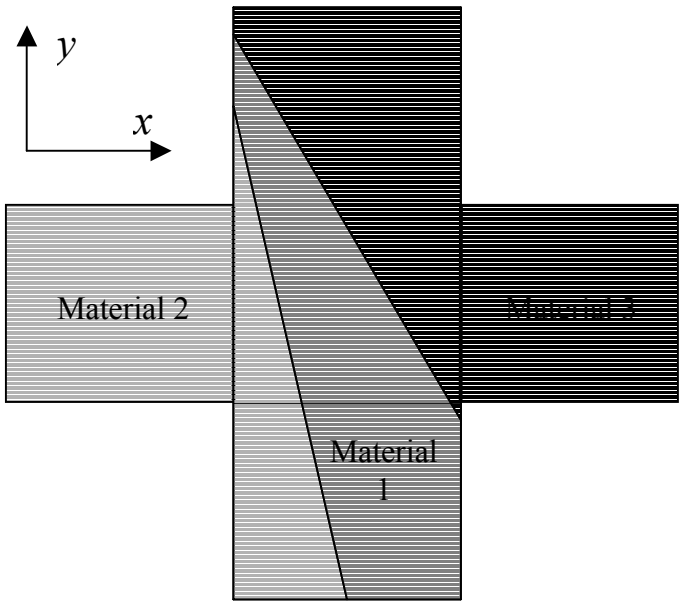

Figure 7. Typical cell and surrounding cells with three materials.

(1) Select a coordinate direction. Evaluate

$$\Delta\phi_m = \phi_{ahead,m} - \phi_{behind,m} \quad m = 1,2,....n \tag{20}$$

for each material in the cell, using the symmetric difference approximations for ϕ_{ahead} and ϕ_{behind} as was presented in Eqs. (6) – (7) (i.e. ϕ_{ahead} is ϕ_e and ϕ_{behind} is ϕ_w for the x-direction).

(2) Evaluate

$$\Delta\Phi = \max\{\Delta\phi_m\} - \min\{\Delta\phi_m\}. \tag{21}$$

(3) Repeat (1) and (2) for each of the coordinate directions. The coordinate direction used for the final ordering is the one resulting in the largest value for $\Delta\Phi$.

(4) After the coordinate direction has been selected, the materials are placed into three categories, and the material ordering is based on the values for $\Delta\phi_m$ for each of the materials in a category. The categories are:
   a) Category 1: Some of this material ahead, none behind
   b) Category 2: Some of this material ahead and behind (or none ahead and behind)
   c) Category 3: Some of this material behind, none ahead

   Category 1 materials are ordered first, based on their values of $\Delta\phi_m$, from largest to smallest, followed by Category 2 and 3 materials, which are ordered the same way.

(5) Materials with equal values of $\Delta\phi_m$ in any one category are ordered randomly.

With regards to treatment of fragments (materials with none of a particular material ahead or behind), this procedure will naturally tend to collect fragments in the center of the parent cell. This seems to be a desirable feature since the refinement process will naturally introduce cell boundaries across the fragments, making it possible to advect these materials without introducing *ad-hoc* advection velocities. Numerical experimentation is needed to verify how well this scheme works for fragments.

Refinement of Cell Variables

Once the volume fractions have been properly mapped from the parent cell to the children, the cell variables can then be mapped. The mapping employed here assures conservation of mass, momentum and internal energy between the parent and its children.

Assuming equal densities in the parents and children, the cell masses for each material in the children are

$$M_{mc} = \frac{M_{mp}}{8}\frac{\phi_{mc}}{\phi_{mp}}, \quad m = 1,2,....n \tag{22}$$

where M_m is the mass of material m, and the subscripts c and p denote child and parent, respectively. Values for the intensive cell material energies for the children are inherited from the parent. The intensive cell total energy is then calculated as

$$E_{tc} = \frac{1}{M_c}\sum_{m=1}^{n} E_{mc}M_{mc}, \tag{23}$$

where M_c is the total mass of the child cell (i.e. the sum of the material masses), E_{mc} is the intensive cell energy for material m and E_{tc} is the intensive total cell energy for the child cell. Values for other intensive variables in the children that are not material intensive variables (for example, internal state variables) are directly inherited from the parents. Currently, the material and cell pressures, temperatures, stress deviators, and artificial viscosities for the children are also directly inherited from the parent. Clearly this is not an optimal scheme. For example, material and cell pressures could be determined from the equation-of-state and mixed-cell thermodynamics models once the material masses and energies have been determined for the children. These modifications will be performed at a later date.

The velocities are face-centered instead of cell-centered and require special treatment to insure momentum conservation. Here we need to determine the velocities of the faces in each of the three coordinate directions at the center of the parent cell. Consider the parent cell subdivided into the eight children shown in Figure 6. The velocity in the 1-direction for the center face becomes:

$$V_{1m} = \frac{1}{M_p}\left[M_p\left(V_{1l} + V_{1r}\right) - V_{1l}\left(M_1 + M_3 + M_5 + M_7\right) - V_{1r}\left(M_2 + M_4 + M_6 + M_8\right)\right], \tag{24}$$

where M_p is the total mass of the parent cell, M_i the total mass of a child cell, V_{1l} the 1-velocity of the *left* face (i.e. the outside face shared by child cells 1, 3, 5 and 7), V_{1r} the 1-velocity of the *right* face (i.e. the outside face shared by child cells 2, 4, 6 and 8), and V_{1m} the 1-velocity of the center face. In a similar manner, the velocities of the center faces in the 2- and 3-directions are computed.

Implementation

FORTRAN routines were written to perform the refinement process described in the previous sections. Using a simple driver program, the routines were all tested. An example of the output from the testing is given in Figure 8. Shown in Figure 8a are volume fractions for three materials in an 8x8 two-dimensional block. In Figure 8b, the volume fractions for the refined cells corresponding to the lower left-hand quadrant of this block are shown. Note that the volume fraction for Material 1 in refined cell (4,4) appears to be quite large. This distortion is a correct result and is caused by the fact that the interface edges do not match up along cells adjacent to parent cell (2,2) (which is used to determine the volume fractions in child cell (4,4)). Recall that the interface reconstruction method described requires that the interfaces be planar and does not require that the planes match up along the edges of adjacent cells.

Material 1

8	1.0000	0.2500	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
7	1.0000	0.2500	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
6	1.0000	0.2500	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
5	1.0000	0.2500	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
4	1.0000	0.2500	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
3	1.0000	0.2500	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
2	0.2500	0.2500	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**

Material 2

8	0.0000	0.2500	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
7	0.0000	0.2500	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
6	0.0000	0.2500	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
5	0.0000	0.2500	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
4	0.0000	0.2500	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
3	0.0000	0.2500	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
2	0.2500	0.2500	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
1	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**

Material 3

8	0.0000	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
7	0.0000	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
6	0.0000	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
5	0.0000	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
4	0.0000	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
3	0.0000	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
2	0.5000	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**

Figure 8a. Example of volume fractions for an 8x8 parent cell.

Material 1

8	1.0000	1.0000	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000
7	1.0000	1.0000	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000
6	1.0000	1.0000	0.6154	0.0000	0.0000	0.0000	0.0000	0.0000
5	1.0000	1.0000	0.3846	0.0000	0.0000	0.0000	0.0000	0.0000
4	0.6154	0.3846	0.8284	0.0858	0.0000	0.0000	0.0000	0.0000
3	0.0000	0.0000	0.0858	0.0000	0.0000	0.0000	0.0000	0.0000
2	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**

Material 2

8	0.0000	0.0000	0.0000	0.5000	1.0000	1.0000	1.0000	1.0000
7	0.0000	0.0000	0.0000	0.5000	1.0000	1.0000	1.0000	1.0000
6	0.0000	0.0000	0.0000	0.4667	1.0000	1.0000	1.0000	1.0000
5	0.0000	0.0000	0.0000	0.5333	1.0000	1.0000	1.0000	1.0000
4	0.0000	0.0000	0.0000	0.0858	1.0000	1.0000	1.0000	1.0000
3	0.4667	0.5333	0.0858	0.8284	1.0000	1.0000	1.0000	1.0000
2	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
1	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**

Material 3

8	0.0000	0.0000	0.5000	0.5000	0.0000	0.0000	0.0000	0.0000
7	0.0000	0.0000	0.5000	0.5000	0.0000	0.0000	0.0000	0.0000
6	0.0000	0.0000	0.3846	0.5333	0.0000	0.0000	0.0000	0.0000
5	0.0000	0.0000	0.6154	0.4667	0.0000	0.0000	0.0000	0.0000
4	0.3846	0.6154	0.1716	0.8284	0.0000	0.0000	0.0000	0.0000
3	0.5333	0.4667	0.8284	0.1716	0.0000	0.0000	0.0000	0.0000
2	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**

Figure 8b. Computed volume fractions for the 8x8 child cells corresponding to the lower left-hand quadrant.

A listing of these routines is included in the appendix. This includes amrzr.f (the top-level routine that does the refinement, modified from Crawford's original version), vofsplit.f (the top-level routine that splits the volume fractions), morder.f (determines the material ordering), gradvof.f (determines the direction of the unit normal vectors), and partvof.f (partitions the volume fractions).

Error Estimation

A key component of any adaptive scheme is the ability to estimate the error in the calculation accurately and efficiently. Decisions are made regarding refinement and/or derefinement based on values computed for the error estimates.

There are numerous techniques that have been used successfully as indicators for the error in numerical solutions of differential equations. *A posteriori* error estimation methods for differential equations have been under continual development for about the last two decades. A summary of the current state-of-the-art in posteriori error estimation can be found in [4]. However, very few methods have been examined and tested for highly nonlinear equations, which exist in shock physics and large deformation applications. Recently, several error indicators were implemented into the Lagrangian code EPIC [5] to test their ability in predicting the regions in a calculation where refinement should be performed [6]. Included in the tests were an RVS indicator (a feature indicator that evaluates a scalar velocity difference at the nodes), a flux jump indicator (a feature indicator that examines the magnitude of the stress jumps across inter-element boundaries), and a gradient recovery indicator [7]. A qualitative assessment was made and none of the error indicators was completely successful in predicting regions where refinement and/or derefinement should probably be performed. As a result, a study was performed to derive a new set of indicators tailored specifically to the differential equations solved in these applications.

The complexity of the governing equations, when coupled with the speed of time integration in the explicit calculation used for these applications, prohibits detailed estimates of the error in the solution. However, residuals associated with each of the governing equations can be estimated very quickly, and do not represent a significant computational burden over and above the time integration itself. Then, hopefully the residuals will provide meaningful information for use in driving an adaptive process. This is the approach considered here. Furthermore, although the numerical integration procedure used in the target application code is based on finite volume, the formulation developed here for the residual estimates is based on finite element methods, using the solution provided by the code as a finite element approximation to the exact solution. In this way we can build on the residual estimation procedures that have already been developed for the finite element method.

Preliminaries

We shall derive here a residual-based, explicit collection of a posteriori error estimators that will provide for independent control and adaptive meshing of functions related to the approximation error in the continuity, momentum and energy equations.

We begin by considering the flow domain Ω which is a bounded domain in $\Re^3$. The primitive variables of interest are the mass density ρ, the velocity field $\mathbf{v}$, and the internal energy e which are functions of the position $\boldsymbol{x} \in \Omega$ and time $t \in [0,\infty)$. Since it is known that the solutions of the governing conservation laws are not generally differentiable everywhere in Ω x $(0,T]$, it is necessary to consider weak forms of these equations. For this purpose, we introduce the spaces $\boldsymbol{P}$, $\boldsymbol{V}$, and $\boldsymbol{E}$ of admissible densities, velocities and internal energy fields defined as that the following integral formulations are meaningful:

Find $\rho \in \boldsymbol{P}$, $\mathbf{v} \in \boldsymbol{V}$, and $e \in \boldsymbol{E}$ such that

$$\int_{\Omega} \left(\partial_t \rho q - \rho \mathbf{v} \cdot \nabla q\right) d\mathbf{x} = \int_{\Gamma_\rho} \sigma_\rho q \, ds, \quad \forall q \in \boldsymbol{P} \tag{25}$$

$$\int_{\Omega} \left[\rho\left(\partial_t \mathbf{v} + \mathbf{v} \cdot \nabla \mathbf{v}\right) \cdot \mathbf{u} + \tau : \nabla \mathbf{u} - p \operatorname{div} \mathbf{u}\right] d\mathbf{x} = \int_{\Omega} \mathbf{f} \cdot \mathbf{u} \, d\mathbf{x} + \int_{\Gamma_v} \boldsymbol{\sigma}_v \cdot \mathbf{u} \, ds, \quad \forall \mathbf{v} \in \boldsymbol{V} \tag{26}$$

and

$$\int_{\Omega} \left[\rho\left(\partial_t e + \mathbf{v} \cdot \nabla e\right) g + \left(\tau \cdot \mathbf{v} - p\mathbf{v}\right) \cdot \nabla g\right] d\mathbf{x} = \int_{\Gamma_e} \sigma_e g \, ds. \quad \forall e \in \boldsymbol{E} \tag{27}$$

Here Γ_ρ, Γ_v, and Γ_e are inflow boundaries through which prescribed mass, momentum and energy fluxes σ_ρ, $\boldsymbol{\sigma}_v$, and σ_e are given; τ is the deviatoric Cauchy stress, a function of gradients of $\mathbf{v}$ and possibly the pressure p, which are specified by the appropriate constitutive equations and equations of state; and $\mathbf{f}$ is a specified body force per unit volume. The spaces of admissible functions are then, for example,

$$\boldsymbol{P} = \left\{q = q(x): \quad q\partial_t \rho - \rho \mathbf{v} \cdot \nabla q \in L^1(\Omega) \quad \text{for all} \quad t > 0, \quad q = 0 \text{ on } \partial\Omega \setminus \Gamma_\rho\right\}, \tag{28}$$

etc. It is easily verified that if the solutions to Eqs. (25) – (27) exist and are sufficiently smooth, then they are also solutions to the classical differential conservation laws given in most discussions of Eulerian mechanics.

Next, we consider a family of partitions $\{P_h\}$ of Ω into the elements Ω_k,

$$\overline{\Omega} = \bigcup_{k=1}^{N} \overline{\Omega}_k, \quad \Omega_k \cap \Omega_j = 0, \quad k \neq j$$

$$N = N(P_h)$$

h being a mesh parameter; e.g. $h = \max_{1 \le k \le N} h_k$, $h_k = \text{dia}\,(\Omega_k)$. We approximate the functions in the spaces $\boldsymbol{P}$, $\boldsymbol{V}$, and $\boldsymbol{E}$ with piecewise polynomials so as to produce the finite dimensional subspaces

$$\boldsymbol{P}^h \subset \boldsymbol{P}, \quad \boldsymbol{V}^h \subset \boldsymbol{V}, \quad \boldsymbol{E}^h \subset \boldsymbol{E}.$$

In subsequent calculations, $\boldsymbol{P}^h$ consists of piecewise constants, $\boldsymbol{V}^h$ of piecewise linear functions, and $\boldsymbol{E}^h$ of piecewise constants. We can now formulate the semi-discrete approximations of the weak forms of the conservation laws given in Eqs. (25) – (27). For example, the semi-discrete approximation of (25) is:

Find $\rho_h \in \boldsymbol{P}^h$ such that

$$\int_{\Omega} \left(\partial_t \rho_h q_h - \rho_h \mathbf{v_h} \cdot \nabla q_h\right) d\mathbf{x} = \int_{\Gamma_\rho} \sigma_\rho q_h \, ds. \quad \forall q_h \in \boldsymbol{P}^h \tag{29}$$

Similar semi-discrete approximations of Eqs. (26) and (27) are constructed, but not stated for brevity.

Error Residual Relation

Let us focus on the continuity equation for clarity. Similar results will follow immediately for the momentum and energy equations.

Let $(\rho, \mathbf{v}, e)(t)$ be the exact solutions of Eqs. (25) – (27) at time t and $(\rho_h, \mathbf{v}_h, e_h)(t)$ be the corresponding semi-discrete approximation. Then the approximation errors at time t are the functions

$$\left.\begin{aligned} \varepsilon_\rho &= \rho - \rho_h \,, \\ \boldsymbol{\varepsilon}_{\mathrm{v}} &= \mathbf{v} - \mathbf{v}_h \,, \end{aligned}\right\} \qquad \begin{aligned} \varepsilon_e &= e - e_h \,. \end{aligned} \tag{30}$$

Returning to Eq. (25), we replace ρ and $\mathbf{v}$ by $\rho_h + \varepsilon_\rho$ and $\mathbf{v}_h + \boldsymbol{\varepsilon}_{\mathrm{v}}$, respectively, to obtain

$$\int_{\Omega} \left[q \partial_t \varepsilon_\rho - \left(\rho_h \boldsymbol{\varepsilon}_{\mathrm{v}} + \varepsilon_\rho \mathbf{v}_h + \varepsilon_\rho \boldsymbol{\varepsilon}_{\mathrm{v}}\right) \cdot \nabla q \right] d\mathbf{x} = R_\rho(q), \quad \forall q \in \boldsymbol{P} \tag{31}$$

where R_ρ is the *total mass residual functional*:

$$R_\rho(q) = \int_{\Gamma_\rho} \sigma_\rho q \, ds - \int_\Omega \left(q \partial_t \rho_h - \rho_h \mathbf{v}_h \cdot \nabla q \right) d\mathbf{x} \,. \tag{32}$$

Note that

$$R_\rho(q_h) = 0. \quad \forall q_h \in \boldsymbol{P}^h \tag{33}$$

It is sometimes convenient to express R_ρ as the sum of residual contributions from each element:

$$R_\rho(q) = \sum_{k=1}^{N} \left\{ \int_{\Omega_k} r_\rho q \, dx + \int_{\partial\Omega_k} r_{b\rho} q \, ds \right\}, \tag{34}$$

where r_ρ is the interior residual defined as

$$r_\rho(\mathbf{x},t) = -\partial_t \rho_h(\mathbf{x},t) - \nabla \cdot (\rho_h \mathbf{v}_h)(\mathbf{x},t), \quad \mathbf{x} \in \Omega_k, \quad t \geq 0 \tag{35}$$

and $r_{b\rho}$ is the boundary residual defined as

$$r_{b\rho}(\mathbf{x},t) = \begin{cases} 0 & \text{on} \quad \Gamma_\rho \\ \mathbf{n}(\mathbf{x},t) \cdot \mathbf{v}_h(\mathbf{x},t) \, [\rho_h(\mathbf{x},t)] & \text{on} \quad \partial\Omega_k \setminus \Gamma_\rho \end{cases}, \quad \mathbf{x} \in \Omega_k, \quad t \geq 0 \tag{36}$$

where $[\rho_h]$ denotes the jump in ρ_h across inter-element boundaries. Similar expressions for the *total momentum and energy residual functionals*, $R_{\mathrm{v}}(\mathbf{u})$ and $R_e(g)$, follow from this result. For $R_{\mathrm{v}}(\mathbf{u})$ we have:

$$R_{\mathrm{v}}(\mathbf{u}) = \sum_{k=1}^{N} \left\{ \int_{\Omega_k} \mathbf{r}_{\mathrm{v}} \cdot \mathbf{u} \, d\mathbf{x} + \int_{\partial\Omega_k} \mathbf{r}_{b\mathrm{v}} \cdot \mathbf{u} \, ds \right\}, \tag{37}$$

where $\mathbf{r}_{\mathrm{v}}$ is the interior residual defined as

$$\mathbf{r}_{\mathrm{v}}(\mathbf{x},t) = -\rho_h \left(\partial_t \mathbf{v}_h - \mathbf{v}_h \nabla \cdot \mathbf{v}_h \right) - \nabla \cdot \tau + \nabla p + \mathbf{f}, \quad \mathbf{x} \in \Omega_k, \quad t \geq 0 \tag{38}$$

and $r_{b\mathrm{v}}$ is the boundary residual defined as

$$\mathbf{r}_{b\mathrm{v}} = \begin{cases} 0 & \text{on} \quad \Gamma_{\mathrm{v}} \\ -[-p_h\mathbf{n} + \tau_h \cdot \mathbf{n}] & \text{on} \quad \partial\Omega_k \setminus \Gamma_{\mathrm{v}} \end{cases}. \quad \mathbf{x} \in \Omega_k, \quad t \geq 0 \tag{39}$$

Likewise, for $R_e(g)$ we have:

$$R_e(g) = \sum_{k=1}^{N} \left\{ \int_{\Omega_k} r_e g \, d\mathbf{x} + \int_{\partial\Omega_k} r_{be} g \, ds \right\}, \tag{40}$$

where r_e is the interior residual defined as

$$r_e(\mathbf{x},t) = -\rho_h(\partial_t e_h + \mathbf{v} \cdot \nabla e) - \nabla \cdot (\tau_h \cdot \mathbf{v}_h - p_h \mathbf{v}_h), \quad \mathbf{x} \in \Omega_k, \quad t \geq 0 \tag{41}$$

and r_{be} is the boundary residual defined as

$$r_{be}(\mathbf{x},t) = \begin{cases} 0 & \text{on} \quad \Gamma_e \\ -[(\tau_h \cdot \mathbf{v}_h - p_h \mathbf{v}_h) \cdot \mathbf{n}] & \text{on} \quad \partial\Omega_k \setminus \Gamma_e \end{cases}. \quad \mathbf{x} \in \Omega_k, \quad t \geq 0 \tag{42}$$

Returning now to Eq. (31), we denote the *discrete convected mass flux error* as the vector field

$$\boldsymbol{\omega}_h = -\rho \boldsymbol{\varepsilon}_{\mathrm{v}} - \varepsilon_\rho \mathbf{v}_h - \varepsilon_\rho \boldsymbol{\varepsilon}_{\mathrm{v}} . \tag{43}$$

Using this relation, we can write the total mass residual functional as

$$\int_{\Omega} \left(q \partial_t \varepsilon_\rho + \boldsymbol{\omega}_h \cdot \nabla q \right) d\mathbf{x} = R_\rho(q). \quad \forall q \in P \tag{44}$$

Error Analysis

To proceed further, we need additional notation and properties. We denote $H^s(\Omega)$, $s \geq 0$, as the Sobolev space of functions with generalized derivatives of order s in $L^2(\Omega)$ (see [8]). In particular, $H^1(\Omega)$ is equipped with the norm

$$\|\mathrm{v}\|_{H^1(\Omega)} = \left\{ \int_{\Omega} \left(\nabla \mathrm{v} \cdot \nabla \mathrm{v} + \mathrm{v}^2 \right) d\mathbf{x} \right\}^{1/2} \tag{45}$$

and the semi-norm

$$|\mathrm{v}|_{H^1(\Omega)} = \left\{ \int_\Omega \nabla \mathrm{v} \cdot \nabla \mathrm{v} \, d\mathbf{x} \right\}^{1/2} . \tag{46}$$

Clearly, $H^0(\Omega) = L^2(\Omega)$, and we denote

$$\|\mathrm{v}\|_{H^0(\Omega)} = \left\{ \int_\Omega \mathrm{v}^2 \, d\mathbf{x} \right\}^{1/2} . \tag{47}$$

Let $\Omega_k \in P_h$ (P_h being a partition of Ω). It is known that if P_h belongs to a regular family of partitions, and $\boldsymbol{V}^h \in \boldsymbol{P}^h$ is a polynomial approximation of $\mathrm{v} \in H^l(\Omega_h)$ of order p, then

$$\left\| \mathrm{v} - \mathrm{v}^h \right\|_{H^r(\Omega_k)} \le C h_k^{p+1-r} |\mathrm{v}|_{H^{p+1}(\Omega_h)} \tag{48}$$

where h_k = dia (Ω_k) and C is a constant independent of v and p. In particular,

$$\left\| \mathrm{v} - \mathrm{v}^h \right\|_{L^2(\Omega_k)} \le C h_k |\mathrm{v}|_{H^1(\Omega_k)} \tag{49}$$

and, on the boundary of Ω_k, it can be shown that

$$\left\| \mathrm{v} - \mathrm{v}^h \right\|_{L^2(\partial\Omega_k)} \le C h_k^{1/2} |\mathrm{v}|_{H^1(\Omega_k)} . \tag{50}$$

Returning to Eqs. (33) and (34), we observe that

$$\begin{aligned} R_\rho(q) &= R_\rho(q - q_h) \\ &= \sum_{k=1}^{N} \left\{ \int_{\Omega_k} r_\rho(q - q_h) \, d\mathbf{x} + \int_{\partial\Omega_k} r_{b\rho}(q - q_h) \, ds \right\} \\ &\le \sum_{k=1}^{N} \left\{ \|r_\rho\|_{L^2(\Omega_k)} \|q - q_h\|_{L^2(\Omega_k)} + \|r_{b\rho}\|_{L^2(\partial\Omega_k)} \|q - q_h\|_{L^2(\partial\Omega_k)} \right\} \\ &\le C \sum_{k=1}^{N} \left\{ h_k \|r_\rho\|_{L^2(\Omega_k)} + \frac{1}{2} h_k^{1/2} \|r_{b\rho}\|_{L^2(\partial\Omega_k)} \right\} \|q\|_{H^1(\Omega_k)} , \end{aligned} \tag{51}$$

where we have used the Cauchy-Schwarz inequality and (49) and (50), and C is a constant independent of ρ, $\mathbf{v}$, h_k and q. Denoting $\eta_{\rho,k}$ as the *local, explicit residual mass density error* indicator for element Ω_k,

$$\eta_{\rho,k} = h_k \left\| r_\rho \right\|_{L^2(\Omega_k)} + \frac{1}{2} h_k^{1/2} \left\| r_{b\rho} \right\|_{L^2(\partial\Omega_k)} \tag{52}$$

and again using the Cauchy-Schwarz inequality, we have

$$R_\rho(q) \le C \left\{ \sum_{k=1}^{N} \eta_{\rho,k}^2 \right\}^{1/2} \left\| q \right\|_{H^1(\Omega)} . \tag{53}$$

Likewise, we can derive similar expressions for the local, explicit residual momentum and energy errors $\eta_{\mathrm{v},k}$ and $\eta_{e,k}$ given as

$$\eta_{\mathrm{v},k} = h_k \left\| \mathbf{r}_{\mathrm{v}} \right\|_{L^2(\Omega_k)} + \frac{1}{2} h_k^{1/2} \left\| \mathbf{r}_{b\mathrm{v}} \right\|_{L^2(\partial\Omega_k)} \tag{54}$$

and

$$\eta_{e,k} = h_k \left\| r_e \right\|_{L^2(\Omega_k)} + \frac{1}{2} h_k^{1/2} \left\| r_{be} \right\|_{L^2(\partial\Omega_k)} . \tag{55}$$

Application of the Cauchy-Schwarz inequality then yields

$$R_{\mathrm{v}}(\mathbf{u}) \le C \left\{ \sum_{k=1}^{N} \eta_{\mathrm{v},k}^2 \right\}^{1/2} \left\| \mathbf{u} \right\|_{H^1(\Omega)} \tag{56}$$

and

$$R_e(g) \le C \left\{ \sum_{k=1}^{N} \eta_{e,k}^2 \right\}^{1/2} \left\| g \right\|_{H^1(\Omega)} . \tag{57}$$

Returning again to the mass density error, we can use the result in Eq. (53) to write Eq. (44) as

$$\int_\Omega \left(q \partial_t \varepsilon_\rho + \boldsymbol{\omega}_h \cdot \nabla q \right) d\mathbf{x} \le C \left\{ \sum_{k=1}^{N} \eta_{\rho,k}^2 \right\}^{1/2} \left\| q \right\|_{H^1(\Omega)} . \tag{58}$$

We next introduce the *elliptic mass density error projection* $\varphi_\rho \in H^1(\Omega)$ defined as the unique function satisfying

$$\int_\Omega \left(\nabla \varphi_\rho \cdot \nabla q + \varphi_\rho q\right) d\mathbf{x} = R_\rho(q)$$

$$= \int_\Omega \left(q \partial_t \varepsilon_\rho + \boldsymbol{\omega}_h \cdot \nabla q\right) d\mathbf{x} . \quad \forall q \in \boldsymbol{P} \tag{59}$$

The left-hand side of Eq. (59) is recognized as the H^1 inner product of φ_ρ and q; thus

$$\left(\varphi_\rho, q\right)_{H^1(\Omega)} \le C \left\{ \sum_{k=1}^{N} \eta_{\rho,k}^2 \right\}^{1/2} \|q\|_{H^1(\Omega)} , \tag{60}$$

and

$$\|\varphi_\rho\|_{H^1(\Omega)} = \sup_{q \in H^1(\Omega)\backslash 0} \frac{\left(\varphi_\rho, q\right)_{H^1(\Omega)}}{\|q\|_{H^1(\Omega)}} \le C \left\{ \sum_{k=1}^{N} \eta_{\rho,k}^2 \right\}^{1/2} . \tag{61}$$

Likewise, for the momentum and energy equations we can define the *elliptic momentum and energy error projections*, $\boldsymbol{\varphi}_\mathrm{v}$ and φ_e, as

$$\int_\Omega \left(\nabla \boldsymbol{\varphi}_\mathrm{v} : \nabla \mathbf{u} + \boldsymbol{\varphi}_\mathrm{v} \cdot \mathbf{u}\right) d\mathbf{x} = R_\mathrm{v}(\mathbf{u}) \quad \forall \mathbf{u} \in \boldsymbol{V} \tag{62}$$

and

$$\int_\Omega \left(\nabla \varphi_e \cdot \nabla g + \varphi_e g\right) d\mathbf{x} = R_e(g) \quad \forall g \in \boldsymbol{E} \tag{63}$$

respectively, to obtain the results

$$\|\boldsymbol{\varphi}_\mathrm{v}\|_{H^1(\Omega)} = \sup_{\mathbf{u} \in H^1(\Omega)\backslash 0} \frac{\left(\boldsymbol{\varphi}_v, \mathbf{u}\right)_{H^1(\Omega)}}{\|\mathbf{u}\|_{H^1(\Omega)}} \le C \left\{ \sum_{k=1}^{N} \eta_{\mathrm{v},k}^2 \right\}^{1/2} \tag{64}$$

and

$$\|\varphi_e\|_{H^1(\Omega)} = \sup_{g \in H^1(\Omega)\backslash 0} \frac{(\varphi_e, g)_{H^1(\Omega)}}{\|g\|_{H^1(\Omega)}} \le C \left\{ \sum_{k=1}^{N} \eta_{e,k}^2 \right\}^{1/2} . \tag{65}$$

Thus, we conclude that the error measures $\eta_{\rho,k}$ represent contributions for each element Ω_k to a global bound on the H^l norm (the energy norm) of φ_ρ, an elliptic smoothing of the mass flux error. Similarly, the error measures $\eta_{\mathrm{v},k}$ and $\eta_{e,k}$ represent contributions to the global bounds on $\boldsymbol{\varphi}_\mathrm{v}$ and φ_e, elliptic smoothings of the momentum and energy flux errors.

One-Dimensional Simulations

In order to test the validity of the residuals for use as error indicators, some one-dimensional calculations of shock propagation were performed. Shown in Figure 9 is a

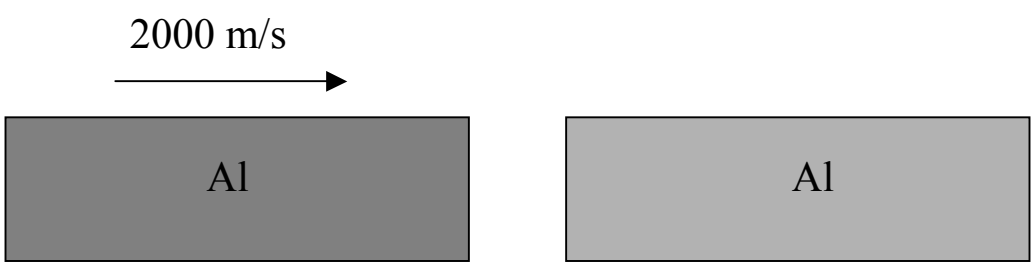

Figure 9. Setup for one-dimensional shock propagation test problem.

description of the problem. An aluminum bar strikes a stationary aluminum bar at 2000 m/s. This causes shock waves to be propagated forward into the stationary bar and backward into the moving bar. Figure 10 shows results from a one-dimensional Lagrangian simulation of the impact encounter. Shown are calculated velocity profiles at various times after impact, along with the exact solution.

Agreement between the solutions is good; albeit the numerical solution exhibits some ringing near the shock boundaries; a consequence of artificial viscosity. Artificial viscosity is not used in the analytical solution.

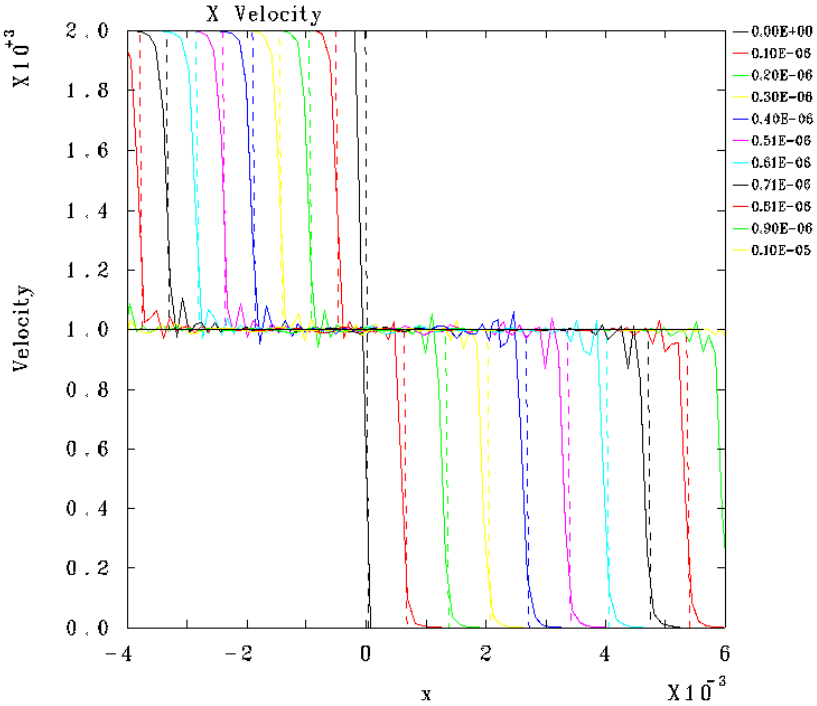

Figure 10. Velocity versus position at several times.

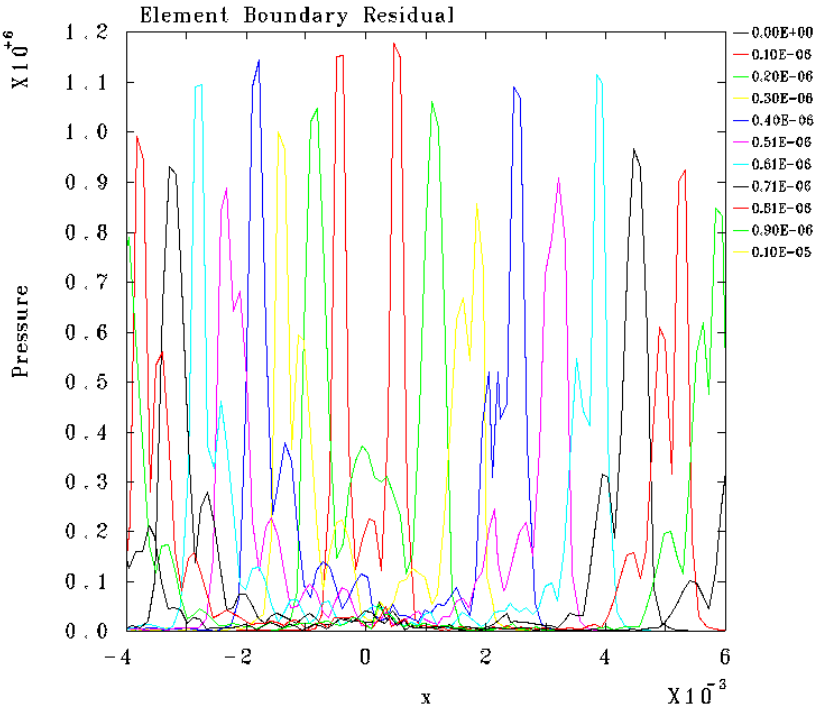

Figure 11. $1/2h_k^{1/2}\|r_{bv}\|$ versus position at several times.

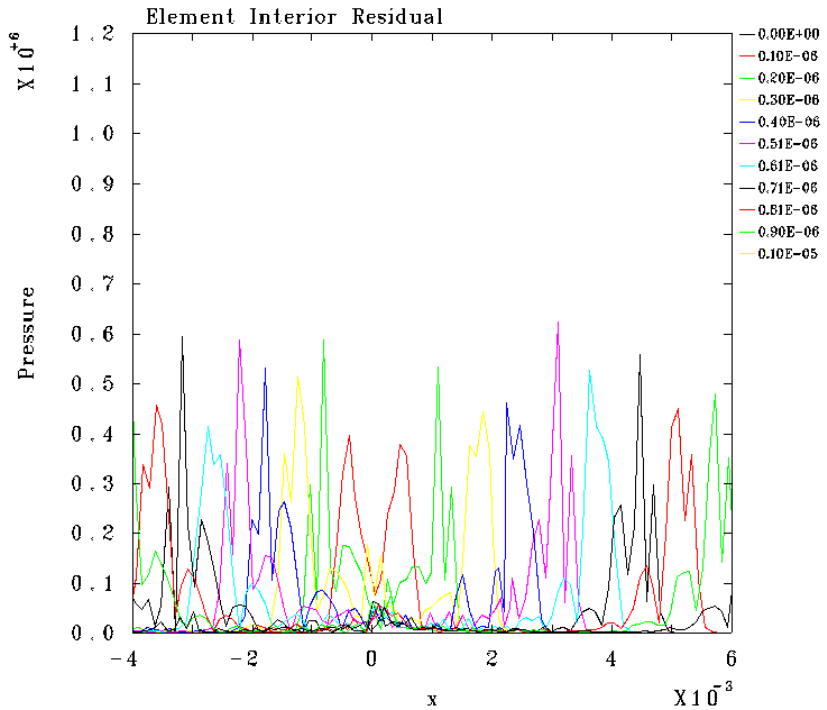

Figure 12. $h_k||r_v||$ versus position at several times.

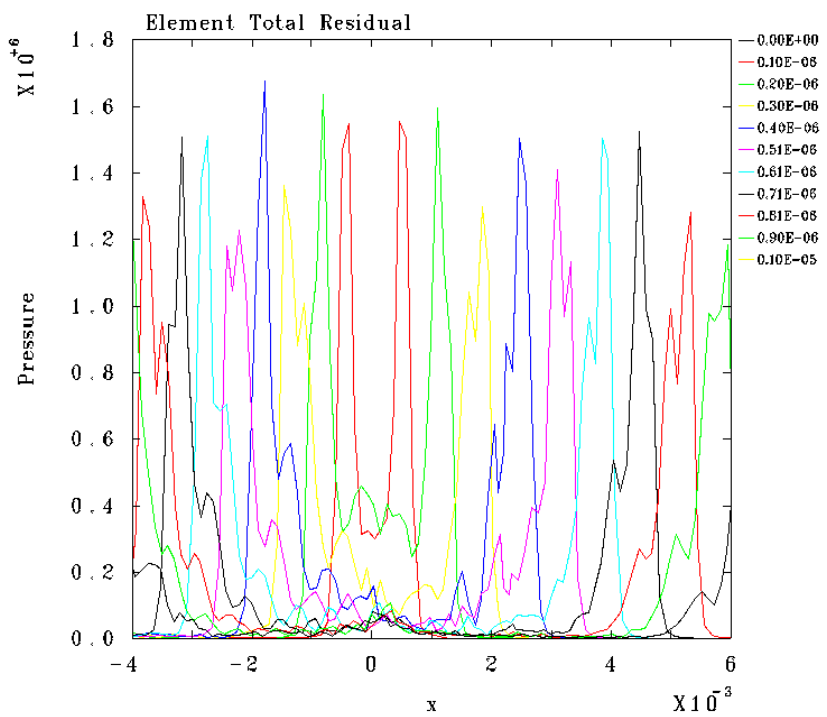

Figure 13. $\eta_{v,k}$ versus position at several times.

Norms of the residuals calculated from the solutions shown in Figure 10 are shown in Figures 11 – 13. In Figure 11, the norm of the boundary residual from the momentum equation, multiplied by $1/2h_k^{1/2}$, is shown. The boundary residual is largest at the shock interface. Figure 12 shows the interior residual from the momentum equation multiplied by h_k. The interior residual is also large near the shock boundary, but is smaller in magnitude than the boundary residual. The sum of these two residuals, which is the error indicator given in Eq. (54), is given in Figure 13. Comparison of this result to the solutions shown in Figure 10 suggests that $\eta_{v,k}$ is a probably suitable for driving an adaptive process.

Two-Dimensional Simulations

In the two-dimensional case it is usually not possible to derive an analytical solution for problems of practical significance. Nevertheless, it is possible to make qualitative judgments about the performance of error indicators in identifying regions in the problem where the error in the solution is probably large (for example, across shock waves or in regions of large deformations).

A two-dimensional Lagrangian simulation of a generic Taylor anvil impact was performed to test the utility of the residuals for use as an indication of the error. Shown in Figure 14a is the problem setup. The steel cylinder, 20 mm in diameter and 40 mm long, strikes a rigid boundary at $x = 0$ with an initial velocity of 1000 m/s. The simulation was run assuming axial symmetry. Interaction with the rigid boundary produces a mushroom-shaped cylinder at subsequent times after impact, which is illustrated in Figures 14b – 14d.

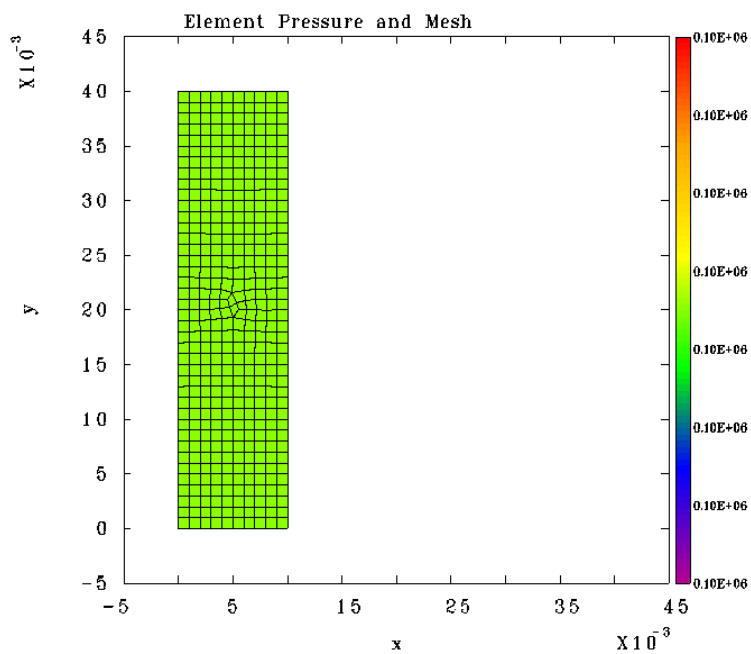

Figure 14a. Problem setup.

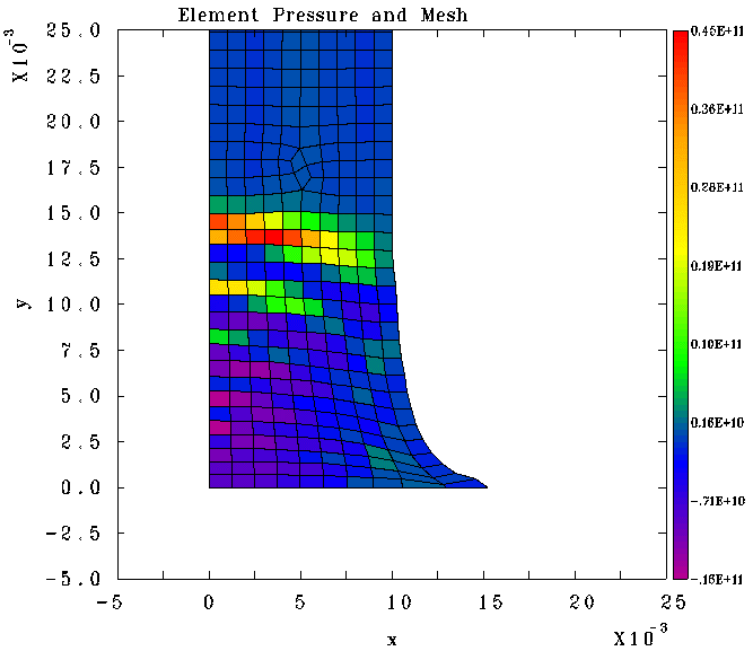

Figure 14b. Pressure contours at 3.0 μs.

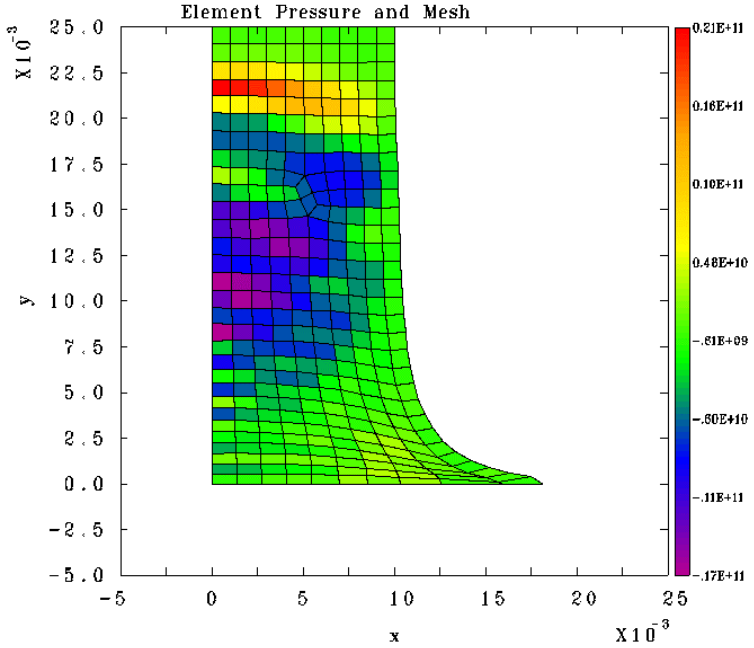

Figure 14c. Pressure at 4.9 μs.

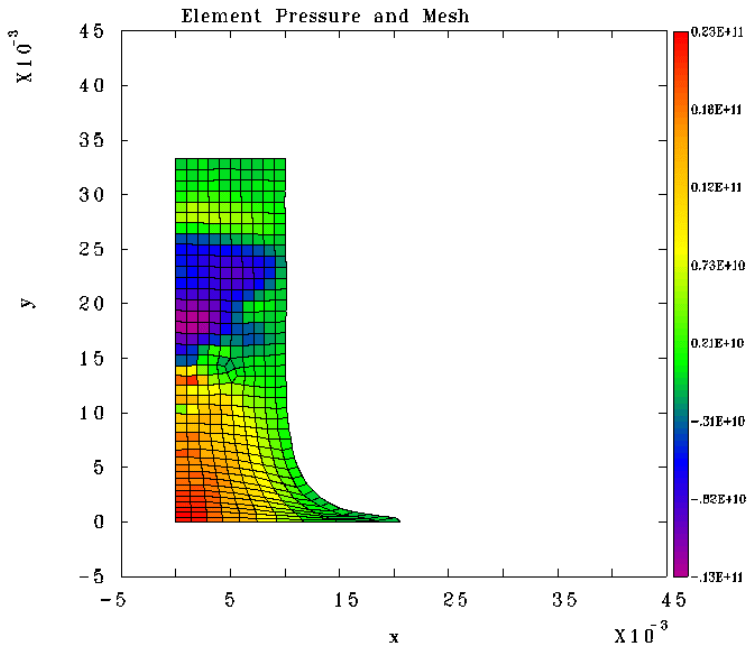

Figure 14d. Pressure at 6.7 μs.

Values for the boundary residual associated with the momentum equation, multiplied $1/2h_k^{1/2}$, are shown in Figures 15a-c. Clearly, the boundary residuals are large across the shock wave propagating back towards the back of the cylinder. However, the boundary residual is not large near the mushrooming head of the cylinder where the largest deformations are occurring. This is probably also a region of large error, and is not identified by the boundary residual.

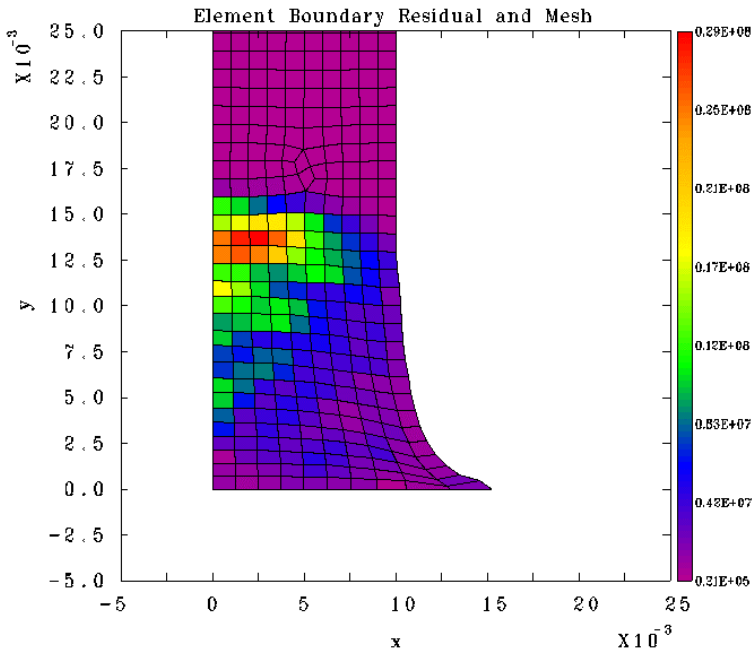

Figure 15a. $1/2h_k^{1/2}||r_{bv}||$ *at* 3.0 μs.

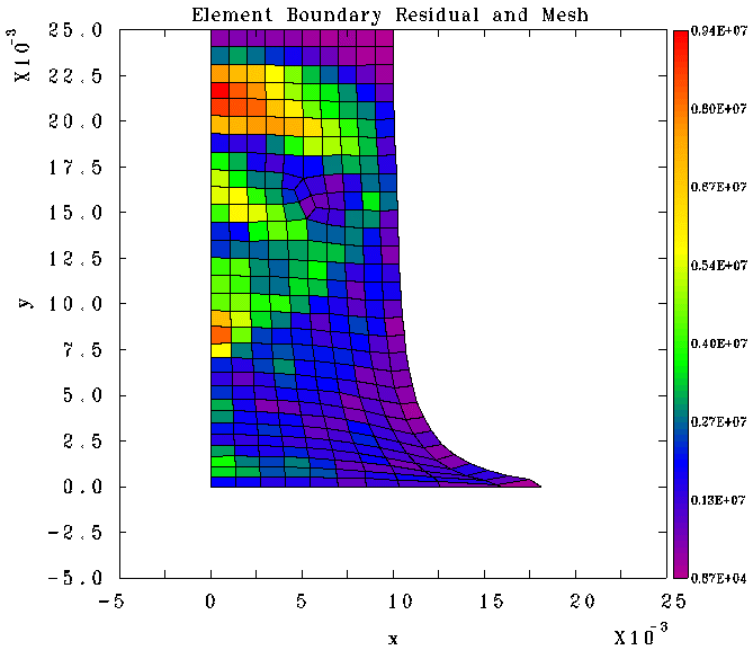

Figure 15b. $1/2h_k^{1/2}||r_{bv}||$ *at* 4.9 μs.

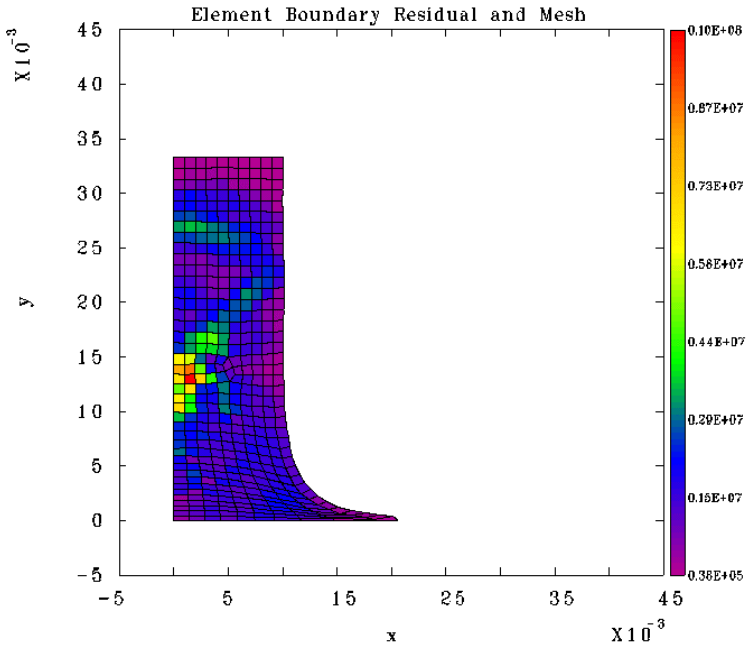

Figure 15c. $1/2h_k^{1/2}||r_{bv}||$ *at* 6.7 μs.

Corresponding values for the interior residual, multiplied by h_k, are shown in Figures 16a-c. As was the case for the boundary residuals, the interior residual is large across the shock wave. However, the magnitudes of the interior residuals are smaller, similar to what was observed in the one-dimensional case. Moreover, it is interesting to note that unlike the boundary residuals, the interior residual becomes large near the edge of the mushrooming head in the cylinder. The sum of the boundary and interior residuals, which is the error indicator given in Eq. (54), is shown in Figures 17a-c. The indicator appears to yield a reasonable representation of regions in this problem where adaptive meshes are probably needed: across the shock wave as well as in regions of large deformation.

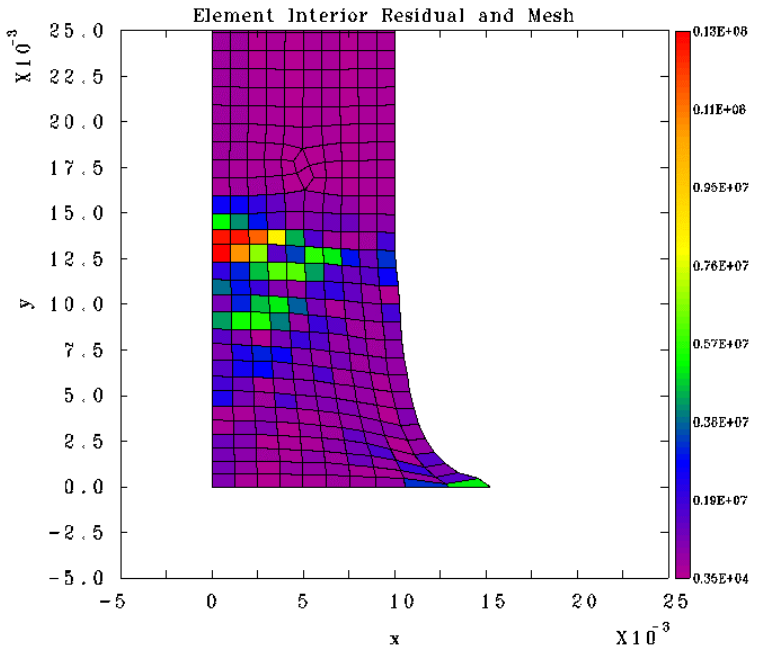

Figure 16a. $h_k\|r_v\|$ ***at*** **3.0 μs.**

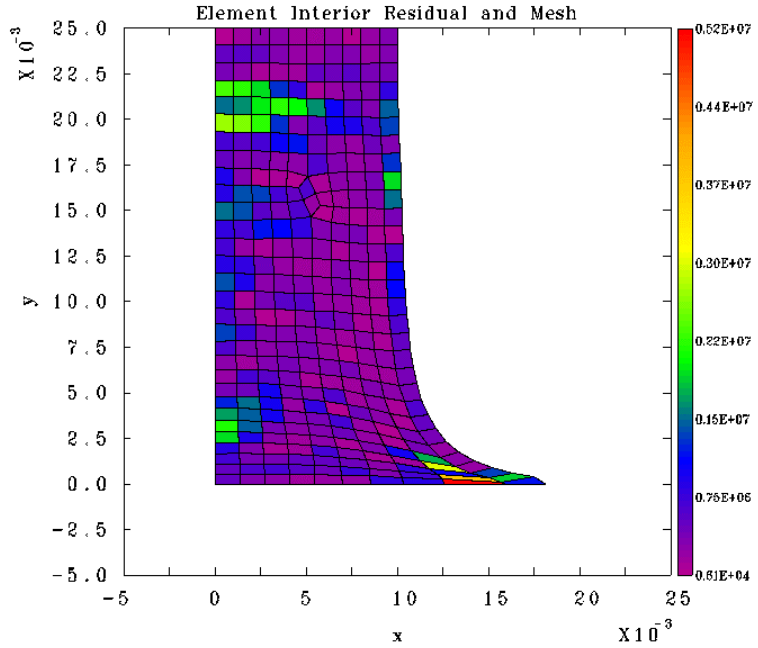

Figure 16b. $h_k\|r_v\|$ ***at*** **4.9 μs.**

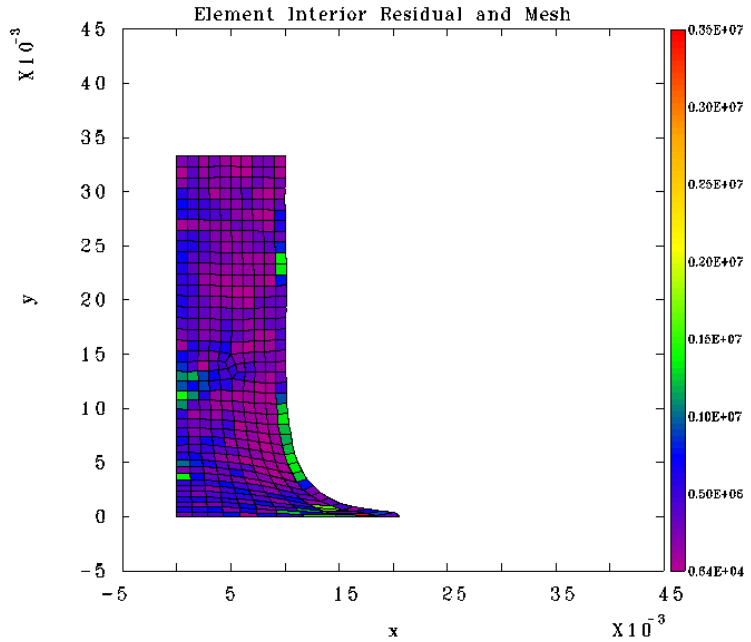

Figure 16c. $h_k\|r_v\|$ *at* 6.7 μs.

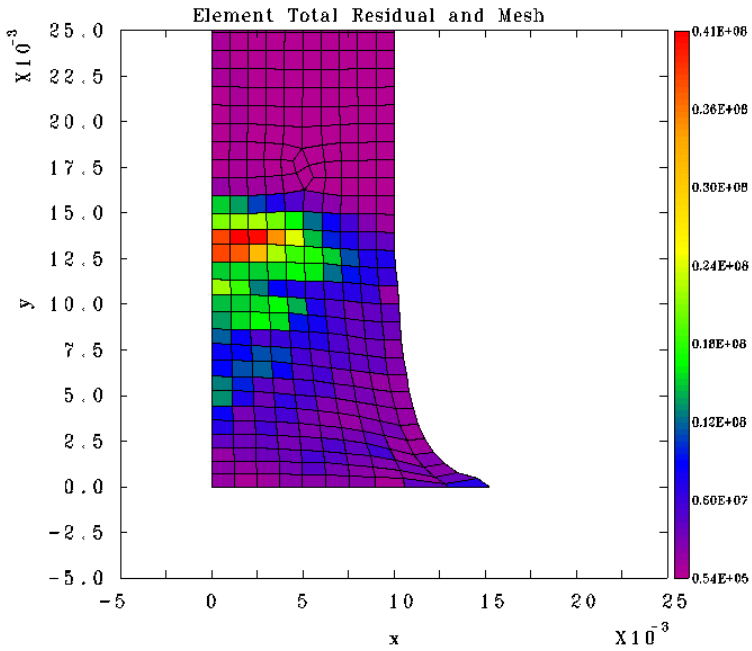

Figure 17a. $\eta_{v,k}$ at 3.0 μs.

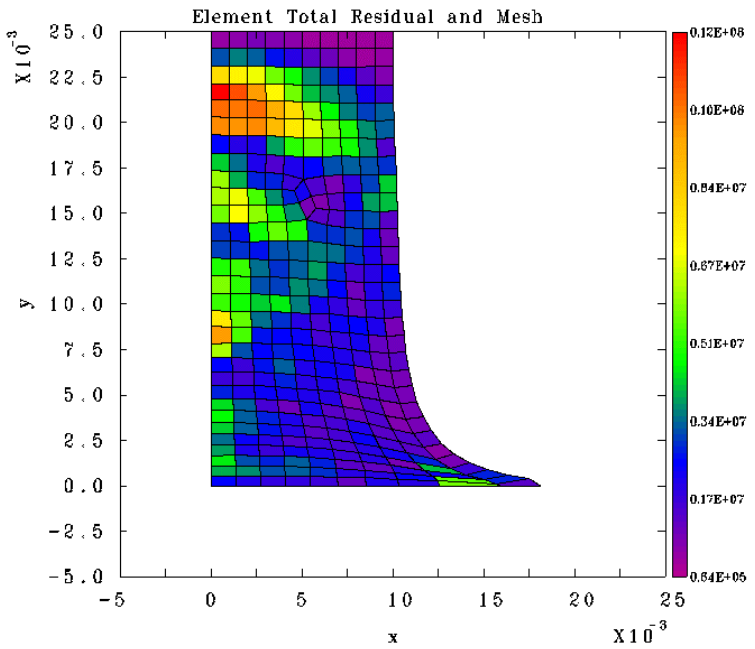

Figure 17b. $\eta_{v,k}$ at 4.9 µs.

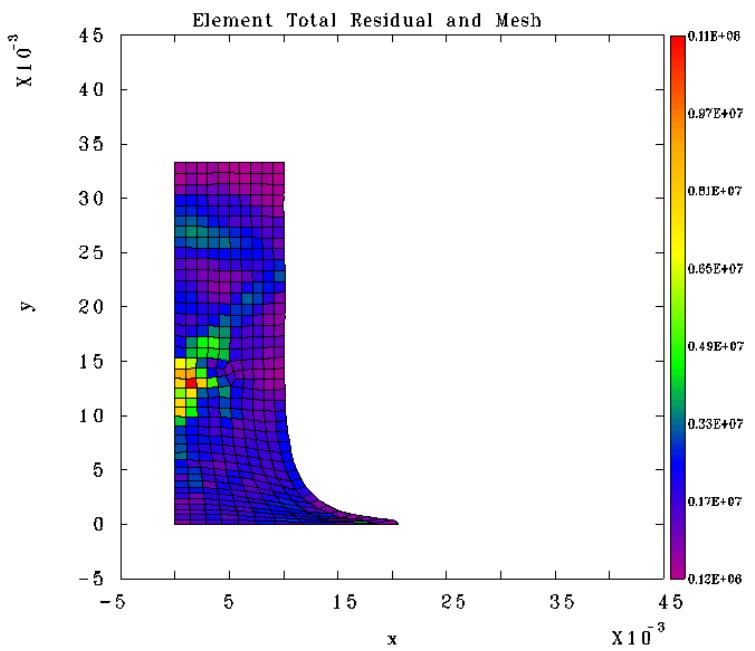

Figure 17c. $\eta_{v,k}$ at 6.7 µs.

Conclusions

Algorithms necessary for implementation of adaptive mesh refinement into a three-dimensional Eulerian hydrocode have been described in this work. An existing technique for interface reconstruction was extended to yield the proper volume fractions for mixed-material cells during refinement. Preliminary results for error indicators are promising, but further enhancements and testing are still needed.

Acknowledgements

This work was supported by the DoD High Performance Computing Modernization Program ERDC Major Shared Resource Center through Programming Environment and Training (PET). The authors would also like to acknowledge Dr. Rick Weed for his suggestions and comments.

References

[1] Crawford, D., "Adaptive mesh refinement in CTH", U. S. Army Symposium on Solid Mechanics, Myrtle Beach SC, April 11-14 (1999).

[2] Youngs, D., "An Interface Tracking Method for a 3D Eulerian Hydrodynamics Code", Atomic Weapons Research Establishment Report No. AWRE/44/92/35, April (1987).

[3] Bell, R. and Hertel, E., "An Improved Material Interface Reconstruction Algorithm for Eulerian Codes", Sandia National Laboratories Report No. SAND92-1716, September (1992).

[4] Ainsworth, M. And Oden, J. T., "A Posteriori Error Estimation in Finite Element Analysis", *Comp. Methods Appl. Mech. Engng*. **142**(1-2), pp. 1–88, (1997).

[5] Johnson, G. R. and Stryk, R. A., "User Instructions for the EPIC-3 Code", Alliant Techsystems, Inc. (1994).

[6] Patra, A. K. and Littlefield, D. L., "A Report on the Development of Some Adaptive Methods for Computational Impact Mechanics", Institute for Advanced Technology Report, in preparation.

[7] Zienkiewicz, O. C. and Zhu, J. C., "The Superconvergent Patch Recovery and A Posteriori Error Estimates, Part I: The Recovery Technique", *Int. J. Numer. Methods Engng.*, **33**, pp. 1331-1364, (1992).

[8] Adams, R. A., S*obolev Spaces*, Academic Press, New York, (1975).

Appendix

Program listing of refinement routines

Modified Routine AMRZR.F

```
      SUBROUTINE AMRZR(
C       input
     &                  KPLANE,IQQ,NSDD,IIQ,JJQ,
     &                  I1,I2,J1,J2,
C       input
     &                  PRES,QX,QY,QZ,
     &                  TEMP,XMAST,VVOID,DE,CE,XMASM,
     &                  ENRGM,VMAT,VX,VY,VZ,SIJ,EXVAR,
     &                  DVOLX,DVOLY,DVOLZ,DIV,
     |                  VMATU,VMATL,VVOIDU,VVOIDL,VZU,
C
C  scratch (dimensioned IMAX)
     |           NMP,
C  scratch (dimensioned IMAX*(NUMMAT+1))
     |           IMP,  MATORD,  VOLSUM,
C  scratch (dimensioned IMAX*3*NUMMAT)
     |           GNRM,
C  scratch (dimensioned NUMMAT+1)
     |           PHIE,  PHIW,  PHIN,  PHIS,  PHIU,  PHID,
     |           DPHI,  ICAT1,  ICAT2,  ICAT3,
C  scratch (dimensioned 8*(NUMMAT+1))
     |           VOLSUMS,
C  scratch (dimensioned 8)
     |           PDISS,  PDISRS,  VOLS,  IPOS,
C
C       output
     &  PRSM,PRSP,QXM,QXP,QYM,QYP,QZM,QZP,TMPM,TMPP,XMSTM,XMSTP,
     &  VOIDM,VOIDP,DEM,DEP,CEM,CEP,XMSMM,XMSMP,XMEM,XMEP,XMVM,XMVP,
     &  VXM,VXP,VYM,VYP,VZMM,VZM,VZP,SIJM,SIJP,EXVM,EXVP,
     &  DVOLXM,DVOLXP,DVOLYM,DVOLYP,DVOLZM,DVOLZP,DIVM,DIVP)
C
C     This routine splits a plane (PRES, QX, etc.) and places
C     the results in planes M and P. This is primarily a placeholder
C     routine using first-order techniques.
C
C     Called by:
C     Database manager: calling routine
C
C  input
C     kplane                    - plane number
C     iqq                       - flag to indicate variables present
C     nsdd                      - number of stress deviators
C     iiq                       - flag to indicate location of results
C                                 = 1 in I=1,IMAX/2
C                                 = 2 in I=IMAX/2+1,IMAX
C     jjq                       - flag to indicate location of results
C                                 = 1 in J=1,JMAX/2
C                                 = 2 in J=JMAX/2+1,JMAX
C     i1,i2                     - define the starting and stopping
C                                 i indices for the conversion
C     j1,j2                     - define the starting and stopping
C                                 j indices for the conversion
C input
C     pres(imax,jmax,0:nummat) - cell pressure (material pressures)
C     qx(imax,jmax)            - x-artificial viscosity
C     qy(imax,jmax)            - y-artificial viscosity
C     qz(imax,jmax)            - z-artificial viscosity
C     temp(imax,jmax,0:nummat) - cell temperature array
C     xmast(imax,jmax)         - total mass array
C     vvoid(imax,jmax)         - void volume fraction array
C     vvoidu(imax,jmax,nummat) - void volume fraction array (kplane+1)
C     vvoidl(imax,jmax,nummat) - void volume fraction array (kplane-1)
```

```
C     de(imax,jmax)            - delta energy array
C     ce(imax,jmax)            - total energy array
C     xmasm(imax,jmax,nummat)  - material mass array
C     enrgm(imax,jmax,nummat)  - material energy array
C     vmat(imax,jmax,nummat)   - material volume fraction array
C     vmatu(imax,jmax,nummat)  - material volume fraction array (kplane+1)
C     vmatl(imax,jmax,nummat)  - material volume fraction array (kplane-1)
C     vx(imax,jmax)            - x velocity array
C     vy(imax,jmax)            - y velocity array
C     vz(imax,jmax)            - z velocity array
C     sij(imax,jmax,nsdd)      - stress deviator array
C     exvar(imax,jmax,nnexv)   - internal state variable array
C     dvol(imax,jmax)          - advection volume (remap step)
C     div(imax,jmax)           - change in volume (remap step)
C
C output
C     prsm(imax,jmax,0:nummat) - pressure at z(kplane)
C     prsp(imax,jmax,0:nummat) - pressure at z(kplane+1)
C     qxm(imax,jmax)           - x-artificial viscosity at z(kplane)
C     qxp(imax,jmax)           - x-artificial viscosity at z(kplane+1)
C     qym(imax,jmax)           - y-artificial viscosity at z(kplane)
C     qyp(imax,jmax)           - y-artificial viscosity at z(kplane+1)
C     qzm(imax,jmax)           - z-artificial viscosity at z(kplane)
C     qzp(imax,jmax)           - z-artificial viscosity at z(kplane+1)
C     tmpm(imax,jmax,0:nummat) - temperature at z(kplane)
C     tmpp(imax,jmax,0:nummat) - temperature at z(kplane+1)
C     xmstm(imax,jmax)         - total mass at z(kplane)
C     xmstp(imax,jmax)         - total mass at z(kplane+1)
C     voidm(imax,jmax)         - void volume fraction at z(kplane)
C     voidp(imax,jmax)         - void volume fraction at z(kplane+1)
C     cem(imax,jmax)           - total cell energy at z(kplane)
C     cep(imax,jmax)           - total cell energy at z(kplane+1)
C     dem(imax,jmax)           - delta cell energy at z(kplane)
C     dep(imax,jmax)           - delta cell energy at z(kplane+1)
C     sijm(imax,jmax,nsdd)     - stress deviators at z(kplane)
C     sijp(imax,jmax,nsdd)     - stress deviators at z(kplane+1)
C     exvm(imax,jmax,nnexv)    - internal state variables at z(kplane)
C     exvp(imax,jmax,nnexv)    - internal state variables at z(kplane+1)
C     vxm(imax,jmax)           - x velocity at z(kplane)
C     vxp(imax,jmax)           - x velocity at z(kplane+1)
C     vym(imax,jmax)           - y velocity at z(kplane)
C     vyp(imax,jmax)           - y velocity at z(kplane+1)
C     vzmm(imax,jmax)          - z velocity at z(kplane-1)
C     vzm(imax,jmax)           - z velocity at z(kplane)
C     vzp(imax,jmax)           - z velocity at z(kplane+1)
C     xmsmm(imax,jmax,nummat)  - material masses at z(kplane)
C     xmsmp(imax,jmax,nummat)  - material masses at z(kplane+1)
C     xmvm(imax,jmax,nummat)   - material volume fractions at z(kplane)
C     xmvp(imax,jmax,nummat)   - material volume fractions at z(kplane+1)
C     xmem(imax,jmax,nummat)   - material energies at z(kplane)
C     xmep(imax,jmax,nummat)   - material energies at z(kplane+1)
C     dvolm(imax,jmax)         - advection volume (remap step)
C     dvolp(imax,jmax)         - advection volume (remap step)
C     divm(imax,jmax)          - change in volume (remap step)
C     divp(imax,jmax)          - change in volume (remap step)
C
C   Author:  Dave Crawford
C   Written:  01/27/99
C
#include "impdoubl.h"
#include "maxmat.h"
#include "comint.h"
#include "dbcsdm.h"
```

```
#include "dbcdim.h"
#include "dbcxyz.h"
#include "dbcvol.h"
#include "ceunxv.h"
#include "tempdef.h"
#include "mpcthn.h"
#include "msgtyp.h"
#include "vofrnd.h"
C
C     input arrays
C
      DIMENSION  PRES(IMAX,JMAX,0:NUMMAT),QX(IMAX,JMAX),QY(IMAX,JMAX),
     &           QZ(IMAX,JMAX),TEMP(IMAX,JMAX,0:NUMMAT),
     &  XMAST(IMAX,JMAX),VVOID(IMAX,JMAX),DE(IMAX,JMAX),
     &  CE(IMAX,JMAX),XMASM(IMAX,JMAX,NUMMAT),
     &  ENRGM(IMAX,JMAX,NUMMAT),VMAT(IMAX,JMAX,NUMMAT),
     &  VX(IMAX,JMAX),VY(IMAX,JMAX),VZ(IMAX,JMAX),
     &  SIJ(IMAX,JMAX,NSDD),EXVAR(IMAX,JMAX,NNEXV),DVOLX(IMAX,JMAX),
     &  DVOLY(IMAX,JMAX),DVOLZ(IMAX,JMAX),DIV(IMAX,JMAX)
C
      DIMENSION VVOIDL(IMAX,JMAX),VVOIDU(IMAX,JMAX),
     &     VMATL(IMAX,JMAX,NUMMAT),VMATU(IMAX,JMAX,NUMMAT),
     &     VZU(IMAX,JMAX)
C
C     output arrays
C
      DIMENSION  PRSM(IMAX,JMAX,0:NUMMAT),PRSP(IMAX,JMAX,0:NUMMAT),
     &           QXM(IMAX,JMAX),QXP(IMAX,JMAX),QYM(IMAX,JMAX),
     &  QYP(IMAX,JMAX),QZM(IMAX,JMAX),QZP(IMAX,JMAX),
     &  TMPM(IMAX,JMAX,0:NUMMAT),TMPP(IMAX,JMAX,0:NUMMAT),
     &  XMSTM(IMAX,JMAX),XMSTP(IMAX,JMAX),VOIDM(IMAX,JMAX),
     &  VOIDP(IMAX,JMAX),CEM(IMAX,JMAX),CEP(IMAX,JMAX),
     &  DEM(IMAX,JMAX),DEP(IMAX,JMAX),SIJM(IMAX,JMAX,NSDD),
     &  SIJP(IMAX,JMAX,NSDD),EXVM(IMAX,JMAX,NNEXV),
     &  EXVP(IMAX,JMAX,NNEXV),VXM(IMAX,JMAX),VXP(IMAX,JMAX),
     &  VZMM(IMAX,JMAX),VYM(IMAX,JMAX),
     &  VYP(IMAX,JMAX),VZM(IMAX,JMAX),VZP(IMAX,JMAX),
     &  XMSMM(IMAX,JMAX,NUMMAT),XMSMP(IMAX,JMAX,NUMMAT),
     &  XMVM(IMAX,JMAX,NUMMAT),XMVP(IMAX,JMAX,NUMMAT),
     &  XMEM(IMAX,JMAX,NUMMAT),XMEP(IMAX,JMAX,NUMMAT),
     &  DVOLXM(IMAX,JMAX),DVOLXP(IMAX,JMAX),
     &  DVOLYM(IMAX,JMAX),DVOLYP(IMAX,JMAX),
     &  DVOLZM(IMAX,JMAX),DVOLZP(IMAX,JMAX),
     &  DIVM(IMAX,JMAX),DIVP(IMAX,JMAX)
C
C   scratch arrays
C
      DIMENSION NMP(IMAX),IMP(IMAX,NUMMAT+1),MATORD(IMAX,NUMMAT+1),
     |      VOLSUM(IMAX,NUMMAT+1),GNRM(IMAX,3,NUMMAT),
     |      PHIE(NUMMAT+1),PHIW(NUMMAT+1),PHIN(NUMMAT+1),
     |      PHIS(NUMMAT+1),PHIU(NUMMAT+1),PHID(NUMMAT+1),
     |      DPHI(NUMMAT+1),ICAT1(NUMMAT+1),ICAT2(NUMMAT+1),
     |      ICAT3(NUMMAT+1),VOLSUMS(8,NUMMAT+1),
     |      PDISS(8),PDISRS(8),VOLS(8),IPOS(8)
C
      PARAMETER  (PZERO=0.0D0,PONE=1.0D0,PTWO=2.0D0,PSMAL=1.0D-14,
     &            PHALF=0.5D0,PQUART=0.25D0,PEIGHTH=0.125D0)
C
      LOGICAL LTPCE,LMAT,LDE,LVEL,LSDEV,LVISC,LXTVAR,LVOL
C
      DATA  LTPCE/.FALSE./,LMAT/.FALSE./,LDE/.FALSE./,LVEL/.FALSE./,
     &      LSDEV/.FALSE./,LVISC/.FALSE./,LXTVAR/.FALSE./,LVOL/.FALSE./
C
```

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C  Update random numbers used by volume remapping routines
C
      CALL CERNG(RANDX)
      CALL CERNG(RANDY)
      CALL CERNG(RANDZ)
C
C        different sets of data are combined
C        based on the value of iqq
C
CBC      Set logicals for appropriate IQQ
C
         LTPCE=.FALSE.
         LVEL=.FALSE.
         LSDEV=.FALSE.
         LVISC=.FALSE.
         LXTVAR=.FALSE.
         LMAT=.FALSE.
         LDE=.FALSE.

C        LVOL is set when volume fractions are true volumes
C         (starting when IQQ=4 and ending when IQQ=5)
C
         IF (IQQ .EQ. 0) THEN
            LTPCE=.TRUE.
            LVEL=.TRUE.
            LSDEV=.TRUE.
            LXTVAR=.TRUE.
            LMAT=.TRUE.
            LDE=.TRUE.
         ELSEIF (IQQ .EQ. 1) THEN
            LTPCE=.TRUE.
            LVEL=.TRUE.
            LSDEV=.TRUE.
            LVISC=.TRUE.
            LXTVAR=.TRUE.
            LMAT=.TRUE.
            LDE=.TRUE.
         ELSEIF (IQQ .EQ. 2) THEN
            LVEL=.TRUE.
         ELSEIF (IQQ .EQ. 3) THEN
            LXTVAR=.TRUE.
            LDE=.TRUE.
         ELSEIF (IQQ .EQ. 4) THEN
            LMAT=.TRUE.
            LVOL=.TRUE.
         ELSEIF (IQQ .EQ. 5) THEN
            LVEL=.TRUE.
            LMAT=.TRUE.
            LVOL=.FALSE.
         ELSEIF (IQQ .EQ. 6) THEN
            LTPCE=.TRUE.
            LMAT=.TRUE.
         ENDIF
C
C
C-----------------------------------------------------------------------
C
C     We include boundary cells in the loops and provide them with
C     our best estimates. In the loops, I0 and J0 always refer to
C     the lower resolution mesh (physically larger cells) and II
C     and JJ refer to the higher resolution mesh.
```

```
      JSTRT=MAX(J1,0)
      JEND=MIN(J2,JMAX)
      ISTRT=MAX(I1,0)
      IEND=MIN(I2,IMAX)

      IF (IIQ.EQ.1) THEN
       IOFFS = 1
      ELSE
       IOFFS = IMAX/2
      ENDIF

      IF (JJQ.EQ.1) THEN
       JOFFS = 1
      ELSE
       JOFFS = JMAX/2
      ENDIF
C
C     Cell volume arrays (VOLXP, VOLYP, VOLZP) should correspond to
C     the volumes of the higher resolution mesh (the output arrays).

       IF (IGM.LT.20) THEN

        CALL BOMBED('1D block split not implemented')
C
C      two-dimensional problem (only M planes count)

       ELSEIF (IGM.GE.20 .AND. IGM.LT.30) THEN

         DO 1000  JJ=JSTRT,JEND,2
          J0 = JJ/2 + JOFFS
          JM=MAX(JJ,1)
          JP=MIN(JJ+1,JMAX)
          DO 900  II=ISTRT,IEND,2
           I0 = II/2 + IOFFS
           IM=MAX(II,1)
           IP=MIN(II+1,IMAX)

C          split material values
C
           IF (LMAT) THEN
C
C            Materials processing....
C            Perform no interface tracking

             DO 12 N = 1,NUMMAT
              IF (LVOL) THEN
                F1 = VOLXP(IM)*VOLYP(JM)
                F2 = VOLXP(IP)*VOLYP(JM)
                F3 = VOLXP(IM)*VOLYP(JP)
                F4 = VOLXP(IP)*VOLYP(JP)
                FD = F1+F2+F3+F4
                F1 = F1/FD
                F2 = F2/FD
                F3 = F3/FD
                F4 = F4/FD
                XMVM(IM,JM,N) = F1*VMAT(I0,J0,N)
                XMVM(IP,JM,N) = F2*VMAT(I0,J0,N)
                XMVM(IM,JP,N) = F3*VMAT(I0,J0,N)
                XMVM(IP,JP,N) = F4*VMAT(I0,J0,N)
              ELSE
                XMVM(IM,JM,N) = VMAT(I0,J0,N)
                XMVM(IP,JM,N) = VMAT(I0,J0,N)
```

```
                 XMVM(IM,JP,N) = VMAT(I0,J0,N)
                 XMVM(IP,JP,N) = VMAT(I0,J0,N)
                ENDIF
12             CONTINUE

C              Advection volume (during remap)

               IF (LVOL) THEN
                 F1 = VOLXP(IM)*VOLYP(JM)
                 F2 = VOLXP(IP)*VOLYP(JM)
                 F3 = VOLXP(IM)*VOLYP(JP)
                 F4 = VOLXP(IP)*VOLYP(JP)
                 FD = F1+F2+F3+F4
                 F1 = F1/FD
                 F2 = F2/FD
                 F3 = F3/FD
                 F4 = F4/FD
                 DVOLXM(IM,JM) = F1*DVOLX(I0,J0)
                 DVOLXM(IP,JM) = F2*DVOLX(I0,J0)
                 DVOLXM(IM,JP) = F3*DVOLX(I0,J0)
                 DVOLXM(IP,JP) = F4*DVOLX(I0,J0)
                 DVOLYM(IM,JM) = F1*DVOLY(I0,J0)
                 DVOLYM(IP,JM) = F2*DVOLY(I0,J0)
                 DVOLYM(IM,JP) = F3*DVOLY(I0,J0)
                 DVOLYM(IP,JP) = F4*DVOLY(I0,J0)
                 DIVM(IM,JM) = DIV(I0,J0)
                 DIVM(IP,JM) = DIV(I0,J0)
                 DIVM(IM,JP) = DIV(I0,J0)
                 DIVM(IP,JP) = DIV(I0,J0)
               ENDIF

               DO 15 N=1,NUMMAT

                IF(MATMMP.NE.0) THEN
                 PRSM(IM,JM,N) = PRES(I0,J0,N)
                 PRSM(IP,JM,N) = PRES(I0,J0,N)
                 PRSM(IM,JP,N) = PRES(I0,J0,N)
                 PRSM(IP,JP,N) = PRES(I0,J0,N)
                ENDIF
                IF(MATTMP.NE.0) THEN
                 TMPM(IM,JM,N) = TEMP(I0,J0,N)
                 TMPM(IP,JM,N) = TEMP(I0,J0,N)
                 TMPM(IM,JP,N) = TEMP(I0,J0,N)
                 TMPM(IP,JP,N) = TEMP(I0,J0,N)
                ENDIF
C               Split mass conserving volume (same density)
                IF (LVOL) THEN
                 F1 = XMVM(IM,JM,N)
                 F2 = XMVM(IP,JM,N)
                 F3 = XMVM(IM,JP,N)
                 F4 = XMVM(IP,JP,N)
                ELSE
                 F1 = XMVM(IM,JM,N)*VOLXP(IM)*VOLYP(JM)
                 F2 = XMVM(IP,JM,N)*VOLXP(IP)*VOLYP(JM)
                 F3 = XMVM(IM,JP,N)*VOLXP(IM)*VOLYP(JP)
                 F4 = XMVM(IP,JP,N)*VOLXP(IP)*VOLYP(JP)
                ENDIF
                FD = F1+F2+F3+F4
                IF (FD.GT.PZERO) THEN
                 F1 = F1/FD
                 F2 = F2/FD
                 F3 = F3/FD
                 F4 = F4/FD
```

```
               ENDIF
               XMSMM(IM,JM,N) = F1*XMASM(I0,J0,N)
               XMSMM(IP,JM,N) = F2*XMASM(I0,J0,N)
               XMSMM(IM,JP,N) = F3*XMASM(I0,J0,N)
               XMSMM(IP,JP,N) = F4*XMASM(I0,J0,N)
               XMEM(IM,JM,N) = ENRGM(I0,J0,N)
               XMEM(IP,JM,N) = ENRGM(I0,J0,N)
               XMEM(IM,JP,N) = ENRGM(I0,J0,N)
               XMEM(IP,JP,N) = ENRGM(I0,J0,N)
15            CONTINUE

C             Total mass and void fraction

               XMSTM(IM,JM) = XMSMM(IM,JM,1)
               XMSTM(IP,JM) = XMSMM(IP,JM,1)
               XMSTM(IM,JP) = XMSMM(IM,JP,1)
               XMSTM(IP,JP) = XMSMM(IP,JP,1)
               IF (LVOL) THEN
                VOIDM(IM,JM)=VOLXP(IM)*VOLYP(JM)-XMVM(IM,JM,1)
                VOIDM(IP,JM)=VOLXP(IP)*VOLYP(JM)-XMVM(IP,JM,1)
                VOIDM(IM,JP)=VOLXP(IM)*VOLYP(JP)-XMVM(IM,JP,1)
                VOIDM(IP,JP)=VOLXP(IP)*VOLYP(JP)-XMVM(IP,JP,1)
               ELSE
                VOIDM(IM,JM) = PONE-XMVM(IM,JM,1)
                VOIDM(IP,JM) = PONE-XMVM(IP,JM,1)
                VOIDM(IM,JP) = PONE-XMVM(IM,JP,1)
                VOIDM(IP,JP) = PONE-XMVM(IP,JP,1)
               ENDIF
               DO 45 N=2,NUMMAT
                XMSTM(IM,JM)=XMSTM(IM,JM)+XMSMM(IM,JM,N)
                XMSTM(IP,JM)=XMSTM(IP,JM)+XMSMM(IP,JM,N)
                XMSTM(IM,JP)=XMSTM(IM,JP)+XMSMM(IM,JP,N)
                XMSTM(IP,JP)=XMSTM(IP,JP)+XMSMM(IP,JP,N)
                VOIDM(IM,JM)=VOIDM(IM,JM)-XMVM(IM,JM,N)
                VOIDM(IP,JM)=VOIDM(IP,JM)-XMVM(IP,JM,N)
                VOIDM(IM,JP)=VOIDM(IM,JP)-XMVM(IM,JP,N)
                VOIDM(IP,JP)=VOIDM(IP,JP)-XMVM(IP,JP,N)
45             CONTINUE
            ENDIF
C
C           split the pressures, artificial viscosities,
C           temperatures,
C           total energies, and delta energies
C
            IF(LVISC)  THEN

C              For refinement, zero the artificial viscosities

               QXM(IM,JM) = PZERO
               QXM(IP,JM) = PZERO
               QXM(IM,JP) = PZERO
               QXM(IP,JP) = PZERO

               QYM(IM,JM) = PZERO
               QYM(IP,JM) = PZERO
               QYM(IM,JP) = PZERO
               QYM(IP,JP) = PZERO

            ENDIF
C
            IF (LTPCE) THEN
              IF (LVOL) THEN
               F1=PONE/VOLXP(IM)/VOLYP(JM)
```

```
              F2=PONE/VOLXP(IP)/VOLYP(JM)
              F3=PONE/VOLXP(IM)/VOLYP(JP)
              F4=PONE/VOLXP(IP)/VOLYP(JP)
            ELSE
              F1=PONE
              F2=PONE
              F3=PONE
              F4=PONE
            ENDIF
C           The pressure variable is used to hold the
C           advection volume in the remap step.
            IF (.NOT.LVOL) THEN
              PRSM(IM,JM,0) = F1*XMVM(IM,JM,1)*PRSM(IM,JM,1)
              PRSM(IP,JM,0) = F2*XMVM(IP,JM,1)*PRSM(IP,JM,1)
              PRSM(IM,JP,0) = F3*XMVM(IM,JP,1)*PRSM(IM,JP,1)
              PRSM(IP,JP,0) = F4*XMVM(IP,JP,1)*PRSM(IP,JP,1)
            ENDIF
            TMPM(IM,JM,0) = F1*XMVM(IM,JM,1)*TMPM(IM,JM,1)
            TMPM(IP,JM,0) = F2*XMVM(IP,JM,1)*TMPM(IP,JM,1)
            TMPM(IM,JP,0) = F3*XMVM(IM,JP,1)*TMPM(IM,JP,1)
            TMPM(IP,JP,0) = F4*XMVM(IP,JP,1)*TMPM(IP,JP,1)
            DO 68 N=2,NUMMAT
              IF (.NOT.LVOL) THEN
               PRSM(IM,JM,0)=PRSM(IM,JM,0)
     2            +F1*XMVM(IM,JM,N)*PRSM(IM,JM,N)
               PRSM(IP,JM,0)=PRSM(IP,JM,0)
     2            +F2*XMVM(IP,JM,N)*PRSM(IP,JM,N)
               PRSM(IM,JP,0)=PRSM(IM,JP,0)
     2            +F3*XMVM(IM,JP,N)*PRSM(IM,JP,N)
               PRSM(IP,JP,0)=PRSM(IP,JP,0)
     2            +F4*XMVM(IP,JP,N)*PRSM(IP,JP,N)
              ENDIF
              TMPM(IM,JM,0) = TMPM(IM,JM,0)
     2            +F1*XMVM(IM,JM,N)*TMPM(IM,JM,N)
              TMPM(IP,JM,0) = TMPM(IP,JM,0)
     2            +F2*XMVM(IP,JM,N)*TMPM(IP,JM,N)
              TMPM(IM,JP,0) = TMPM(IM,JP,0)
     2            +F3*XMVM(IM,JP,N)*TMPM(IM,JP,N)
              TMPM(IP,JP,0) = TMPM(IP,JP,0)
     2            +F4*XMVM(IP,JP,N)*TMPM(IP,JP,N)
68          CONTINUE
C           The total energy variable is used to hold
C           advection volume in the remap step.
            IF (.NOT.LVOL) THEN
              F1 = VOLXP(IM)*VOLYP(JM)
              F2 = VOLXP(IP)*VOLYP(JM)
              F3 = VOLXP(IM)*VOLYP(JP)
              F4 = VOLXP(IP)*VOLYP(JP)
              FD = F1+F2+F3+F4
              F1 = F1/FD
              F2 = F2/FD
              F3 = F3/FD
              F4 = F4/FD
              CEM(IM,JM) = F1*CE(I0,J0)
              CEM(IP,JM) = F2*CE(I0,J0)
              CEM(IM,JP) = F3*CE(I0,J0)
              CEM(IP,JP) = F4*CE(I0,J0)
            ENDIF
          ENDIF

          IF (LDE) THEN
             F1 = VOLXP(IM)*VOLYP(JM)
             F2 = VOLXP(IP)*VOLYP(JM)
```

```
               F3 = VOLXP(IM)*VOLYP(JP)
               F4 = VOLXP(IP)*VOLYP(JP)
               FD = F1+F2+F3+F4
               F1 = F1/FD
               F2 = F2/FD
               F3 = F3/FD
               F4 = F4/FD
               DEM(IM,JM) = F1*DE(I0,J0)
               DEM(IP,JM) = F2*DE(I0,J0)
               DEM(IM,JP) = F3*DE(I0,J0)
               DEM(IP,JP) = F4*DE(I0,J0)
            ENDIF
C
C           split the velocities
C
            IF (LVEL) THEN
               VXM(IM,JM) = VX(I0,J0)
               VYM(IM,JM) = VY(I0,J0)
               VXM(IP,JM) = VX(I0,J0)
               VYM(IP,JM) = VY(I0,J0)
               VXM(IM,JP) = VX(I0,J0)
               VYM(IM,JP) = VY(I0,J0)
               VXM(IP,JP) = VX(I0,J0)
               VYM(IP,JP) = VY(I0,J0)
            ENDIF
C
C           if stress deviators are present,
C           split them
C
            IF ((LSDEV).AND.(NSDD.GT.1))  THEN
             DO 110  NSD=1,NSDD
               SIJM(IM,JM,NSD) = SIJ(I0,J0,NSD)
               SIJM(IP,JM,NSD) = SIJ(I0,J0,NSD)
               SIJM(IM,JP,NSD) = SIJ(I0,J0,NSD)
               SIJM(IP,JP,NSD) = SIJ(I0,J0,NSD)
110          CONTINUE
            ENDIF
C
C           if internal state variables are present,
C           split them
C
            IF ((LXTVAR).AND.(NAEXV.GT.0))  THEN
             DO 125  NXV=1,NAEXV
               EXVM(IM,JM,NXV) = EXVAR(I0,J0,NXV)
               EXVM(IP,JM,NXV) = EXVAR(I0,J0,NXV)
               EXVM(IM,JP,NXV) = EXVAR(I0,J0,NXV)
               EXVM(IP,JP,NXV) = EXVAR(I0,J0,NXV)
125          CONTINUE
            ENDIF

 900       CONTINUE
1000      CONTINUE

C       End 2D

        ELSEIF (IGM.GE.30) THEN

C        three dimensional problem...split into M AND P planes

          KUP=MIN(KPLANE+1,KMAX)
          KDOWN=MAX(KPLANE-1,1)
          IF(LMAT.OR.LVEL.OR.LTPCE)THEN
C
```

```
C  Split volume fractions, preserving interfaces.
C
            CALL VOFSPLIT(KPLANE,  KUP,  KDOWN,
C  input
     |           IOFFS,  ISTRT,  IEND,  JOFFS,  JSTRT,  JEND,
     |           LVOL,
     |           VVOID,  VVOIDL,  VVOIDU,
     |           VMAT,  VMATL, VMATU,
C  scratch (dimensioned IMAX)
     |           NMP,
C  scratch (dimensioned IMAX*(NUMMAT+1))
     |           IMP,  MATORD,  VOLSUM,
C  scratch (dimensioned IMAX*3*NUMMAT)
     |           GNRM,
C  scratch (dimensioned NUMMAT+1)
     |           PHIE,  PHIW,  PHIN,  PHIS,  PHIU,  PHID,
     |           DPHI,  ICAT1,  ICAT2,  ICAT3,
C  scratch (dimensioned 8*(NUMMAT+1))
     |           VOLSUMS,
C  scratch (dimensioned 8)
     |           PDISS,  PDISRS,  VOLS,  IPOS,
C  output
     |           VOIDM,  VOIDP,  XMVM,  XMVP)
        ENDIF
C
        DO 2000  JJ=JSTRT,JEND,2
         J0 = JJ/2 + JOFFS
         JM=MAX(JJ,1)
         JP=MIN(JJ+1,JMAX)
         DO 1900  II=ISTRT,IEND,2
          I0 = II/2 + IOFFS
          IM=MAX(II,1)
          IP=MIN(II+1,IMAX)

C         split material values
C
          IF (LMAT) THEN
C
C           Advection volume (during remap)
C
            IF (LVOL) THEN
               F1 = VOLXP(IM)*VOLYP(JM)*VOLZP(KPLANE)
               F2 = VOLXP(IP)*VOLYP(JM)*VOLZP(KPLANE)
               F3 = VOLXP(IM)*VOLYP(JP)*VOLZP(KPLANE)
               F4 = VOLXP(IP)*VOLYP(JP)*VOLZP(KPLANE)
               F5 = VOLXP(IM)*VOLYP(JM)*VOLZP(KUP)
               F6 = VOLXP(IP)*VOLYP(JM)*VOLZP(KUP)
               F7 = VOLXP(IM)*VOLYP(JP)*VOLZP(KUP)
               F8 = VOLXP(IP)*VOLYP(JP)*VOLZP(KUP)
               FD = F1+F2+F3+F4+F5+F6+F7+F8
               F1 = F1/FD
               F2 = F2/FD
               F3 = F3/FD
               F4 = F4/FD
               F5 = F5/FD
               F6 = F6/FD
               F7 = F7/FD
               F8 = F8/FD
               DVOLXM(IM,JM) = F1*DVOLX(I0,J0)
               DVOLXM(IP,JM) = F2*DVOLX(I0,J0)
               DVOLXM(IM,JP) = F3*DVOLX(I0,J0)
               DVOLXM(IP,JP) = F4*DVOLX(I0,J0)
               DVOLXP(IM,JM) = F5*DVOLX(I0,J0)
```

```
                DVOLXP(IP,JM) = F6*DVOLX(I0,J0)
                DVOLXP(IM,JP) = F7*DVOLX(I0,J0)
                DVOLXP(IP,JP) = F8*DVOLX(I0,J0)
                DVOLYM(IM,JM) = F1*DVOLY(I0,J0)
                DVOLYM(IP,JM) = F2*DVOLY(I0,J0)
                DVOLYM(IM,JP) = F3*DVOLY(I0,J0)
                DVOLYM(IP,JP) = F4*DVOLY(I0,J0)
                DVOLYP(IM,JM) = F5*DVOLY(I0,J0)
                DVOLYP(IP,JM) = F6*DVOLY(I0,J0)
                DVOLYP(IM,JP) = F7*DVOLY(I0,J0)
                DVOLYP(IP,JP) = F8*DVOLY(I0,J0)
                DVOLZM(IM,JM) = F1*DVOLZ(I0,J0)
                DVOLZM(IP,JM) = F2*DVOLZ(I0,J0)
                DVOLZM(IM,JP) = F3*DVOLZ(I0,J0)
                DVOLZM(IP,JP) = F4*DVOLZ(I0,J0)
                DVOLZP(IM,JM) = F5*DVOLZ(I0,J0)
                DVOLZP(IP,JM) = F6*DVOLZ(I0,J0)
                DVOLZP(IM,JP) = F7*DVOLZ(I0,J0)
                DVOLZP(IP,JP) = F8*DVOLZ(I0,J0)
                DIVM(IM,JM) = DIV(I0,J0)
                DIVM(IP,JM) = DIV(I0,J0)
                DIVM(IM,JP) = DIV(I0,J0)
                DIVM(IP,JP) = DIV(I0,J0)
                DIVP(IM,JM) = DIV(I0,J0)
                DIVP(IP,JM) = DIV(I0,J0)
                DIVP(IM,JP) = DIV(I0,J0)
                DIVP(IP,JP) = DIV(I0,J0)
             ENDIF

             DO 1015 N=1,NUMMAT

              IF(MATMMP.NE.0) THEN
               PRSM(IM,JM,N) = PRES(I0,J0,N)
               PRSM(IP,JM,N) = PRES(I0,J0,N)
               PRSM(IM,JP,N) = PRES(I0,J0,N)
               PRSM(IP,JP,N) = PRES(I0,J0,N)
               PRSP(IM,JM,N) = PRES(I0,J0,N)
               PRSP(IP,JM,N) = PRES(I0,J0,N)
               PRSP(IM,JP,N) = PRES(I0,J0,N)
               PRSP(IP,JP,N) = PRES(I0,J0,N)
              ENDIF
              IF(MATTMP.NE.0) THEN
               TMPM(IM,JM,N) = TEMP(I0,J0,N)
               TMPM(IP,JM,N) = TEMP(I0,J0,N)
               TMPM(IM,JP,N) = TEMP(I0,J0,N)
               TMPM(IP,JP,N) = TEMP(I0,J0,N)
               TMPP(IM,JM,N) = TEMP(I0,J0,N)
               TMPP(IP,JM,N) = TEMP(I0,J0,N)
               TMPP(IM,JP,N) = TEMP(I0,J0,N)
               TMPP(IP,JP,N) = TEMP(I0,J0,N)
              ENDIF
C
C  Split mass conserving volume (same density)
C
              IF(LVOL)THEN
                 F1=XMASM(I0,J0,N)/VMAT(I0,J0,N)
                 XMSMM(IM,JM,N)=F1*XMVM(IM,JM,N)
                 XMSMM(IP,JM,N)=F1*XMVM(IP,JM,N)
                 XMSMM(IM,JP,N)=F1*XMVM(IM,JP,N)
                 XMSMM(IP,JP,N)=F1*XMVM(IP,JP,N)
                 XMSMP(IM,JM,N)=F1*XMVP(IM,JM,N)
                 XMSMP(IP,JM,N)=F1*XMVP(IP,JM,N)
                 XMSMP(IM,JP,N)=F1*XMVP(IM,JP,N)
```

```
                 XMSMP(IP,JP,N)=F1*XMVP(IP,JP,N)
              ELSE
                 F1 = VOLXP(IM)*VOLYP(JM)*VOLZP(KPLANE)
                 F2 = VOLXP(IP)*VOLYP(JM)*VOLZP(KPLANE)
                 F3 = VOLXP(IM)*VOLYP(JP)*VOLZP(KPLANE)
                 F4 = VOLXP(IP)*VOLYP(JP)*VOLZP(KPLANE)
                 F5 = VOLXP(IM)*VOLYP(JM)*VOLZP(KUP)
                 F6 = VOLXP(IP)*VOLYP(JM)*VOLZP(KUP)
                 F7 = VOLXP(IM)*VOLYP(JP)*VOLZP(KUP)
                 F8 = VOLXP(IP)*VOLYP(JP)*VOLZP(KUP)
                 FD = F1+F2+F3+F4+F5+F6+F7+F8
                 F1 = F1/FD
                 F2 = F2/FD
                 F3 = F3/FD
                 F4 = F4/FD
                 F5 = F5/FD
                 F6 = F6/FD
                 F7 = F7/FD
                 F8 = F8/FD
                 FD=XMASM(I0,J0,N)/VMAT(I0,J0,N)
                 XMSMM(IM,JM,N)=F1*FD*XMVM(IM,JM,N)
                 XMSMM(IP,JM,N)=F2*FD*XMVM(IP,JM,N)
                 XMSMM(IM,JP,N)=F3*FD*XMVM(IM,JP,N)
                 XMSMM(IP,JP,N)=F4*FD*XMVM(IP,JP,N)
                 XMSMP(IM,JM,N)=F5*FD*XMVP(IM,JM,N)
                 XMSMP(IP,JM,N)=F6*FD*XMVP(IP,JM,N)
                 XMSMP(IM,JP,N)=F7*FD*XMVP(IM,JP,N)
                 XMSMP(IP,JP,N)=F8*FD*XMVP(IP,JP,N)
              ENDIF
              XMEM(IM,JM,N) = ENRGM(I0,J0,N)
              XMEM(IP,JM,N) = ENRGM(I0,J0,N)
              XMEM(IM,JP,N) = ENRGM(I0,J0,N)
              XMEM(IP,JP,N) = ENRGM(I0,J0,N)
              XMEP(IM,JM,N) = ENRGM(I0,J0,N)
              XMEP(IP,JM,N) = ENRGM(I0,J0,N)
              XMEP(IM,JP,N) = ENRGM(I0,J0,N)
              XMEP(IP,JP,N) = ENRGM(I0,J0,N)
C
1015        CONTINUE

C           Total mass (void fraction computed during split)

              XMSTM(IM,JM) = XMSMM(IM,JM,1)
              XMSTM(IP,JM) = XMSMM(IP,JM,1)
              XMSTM(IM,JP) = XMSMM(IM,JP,1)
              XMSTM(IP,JP) = XMSMM(IP,JP,1)
              XMSTP(IM,JM) = XMSMP(IM,JM,1)
              XMSTP(IP,JM) = XMSMP(IP,JM,1)
              XMSTP(IM,JP) = XMSMP(IM,JP,1)
              XMSTP(IP,JP) = XMSMP(IP,JP,1)
              DO 1045 N=2,NUMMAT
               XMSTM(IM,JM)=XMSTM(IM,JM)+XMSMM(IM,JM,N)
               XMSTM(IP,JM)=XMSTM(IP,JM)+XMSMM(IP,JM,N)
               XMSTM(IM,JP)=XMSTM(IM,JP)+XMSMM(IM,JP,N)
               XMSTM(IP,JP)=XMSTM(IP,JP)+XMSMM(IP,JP,N)
               XMSTP(IM,JM)=XMSTP(IM,JM)+XMSMP(IM,JM,N)
               XMSTP(IP,JM)=XMSTP(IP,JM)+XMSMP(IP,JM,N)
               XMSTP(IM,JP)=XMSTP(IM,JP)+XMSMP(IM,JP,N)
               XMSTP(IP,JP)=XMSTP(IP,JP)+XMSMP(IP,JP,N)
1045          CONTINUE
           ENDIF

C          split the pressures, artificial viscosities,
```

```
C          temperatures, total energies, and delta energies

           IF(LVISC)  THEN

C             For refinement, zero the artificial viscosities

              QXM(IM,JM) = PZERO
              QXM(IP,JM) = PZERO
              QXM(IM,JP) = PZERO
              QXM(IP,JP) = PZERO
              QXP(IM,JM) = PZERO
              QXP(IP,JM) = PZERO
              QXP(IM,JP) = PZERO
              QXP(IP,JP) = PZERO

              QYM(IM,JM) = PZERO
              QYM(IP,JM) = PZERO
              QYM(IM,JP) = PZERO
              QYM(IP,JP) = PZERO
              QYP(IM,JM) = PZERO
              QYP(IP,JM) = PZERO
              QYP(IM,JP) = PZERO
              QYP(IP,JP) = PZERO

              QZM(IM,JM) = PZERO
              QZM(IP,JM) = PZERO
              QZM(IM,JP) = PZERO
              QZM(IP,JP) = PZERO
              QZP(IM,JM) = PZERO
              QZP(IP,JM) = PZERO
              QZP(IM,JP) = PZERO
              QZP(IP,JP) = PZERO

           ENDIF

           IF (LTPCE) THEN
             IF (LVOL) THEN
              F1=PONE/VOLXP(IM)/VOLYP(JM)/VOLZP(KPLANE)
              F2=PONE/VOLXP(IP)/VOLYP(JM)/VOLZP(KPLANE)
              F3=PONE/VOLXP(IM)/VOLYP(JP)/VOLZP(KPLANE)
              F4=PONE/VOLXP(IP)/VOLYP(JP)/VOLZP(KPLANE)
              F5=PONE/VOLXP(IM)/VOLYP(JM)/VOLZP(KUP)
              F6=PONE/VOLXP(IP)/VOLYP(JM)/VOLZP(KUP)
              F7=PONE/VOLXP(IM)/VOLYP(JP)/VOLZP(KUP)
              F8=PONE/VOLXP(IP)/VOLYP(JP)/VOLZP(KUP)
             ELSE
              F1=PONE
              F2=PONE
              F3=PONE
              F4=PONE
              F5=PONE
              F6=PONE
              F7=PONE
              F8=PONE
             ENDIF
C            The pressure variable is used to hold the
C            advection volume in the remap step.
             IF (.NOT.LVOL) THEN
              PRSM(IM,JM,0) = F1*XMVM(IM,JM,1)*PRSM(IM,JM,1)
              PRSM(IP,JM,0) = F2*XMVM(IP,JM,1)*PRSM(IP,JM,1)
              PRSM(IM,JP,0) = F3*XMVM(IM,JP,1)*PRSM(IM,JP,1)
              PRSM(IP,JP,0) = F4*XMVM(IP,JP,1)*PRSM(IP,JP,1)
              PRSP(IM,JM,0) = F5*XMVP(IM,JM,1)*PRSP(IM,JM,1)
```

```
              PRSP(IP,JM,0) = F6*XMVP(IP,JM,1)*PRSP(IP,JM,1)
              PRSP(IM,JP,0) = F7*XMVP(IM,JP,1)*PRSP(IM,JP,1)
              PRSP(IP,JP,0) = F8*XMVP(IP,JP,1)*PRSP(IP,JP,1)
             ENDIF
             TMPM(IM,JM,0) = F1*XMVM(IM,JM,1)*TMPM(IM,JM,1)
             TMPM(IP,JM,0) = F2*XMVM(IP,JM,1)*TMPM(IP,JM,1)
             TMPM(IM,JP,0) = F3*XMVM(IM,JP,1)*TMPM(IM,JP,1)
             TMPM(IP,JP,0) = F4*XMVM(IP,JP,1)*TMPM(IP,JP,1)
             TMPP(IM,JM,0) = F5*XMVP(IM,JM,1)*TMPP(IM,JM,1)
             TMPP(IP,JM,0) = F6*XMVP(IP,JM,1)*TMPP(IP,JM,1)
             TMPP(IM,JP,0) = F7*XMVP(IM,JP,1)*TMPP(IM,JP,1)
             TMPP(IP,JP,0) = F8*XMVP(IP,JP,1)*TMPP(IP,JP,1)
             DO 1068 N=2,NUMMAT
              IF (.NOT.LVOL) THEN
               PRSM(IM,JM,0)=PRSM(IM,JM,0)
     2            +F1*XMVM(IM,JM,N)*PRSM(IM,JM,N)
               PRSM(IP,JM,0)=PRSM(IP,JM,0)
     2            +F2*XMVM(IP,JM,N)*PRSM(IP,JM,N)
               PRSM(IM,JP,0)=PRSM(IM,JP,0)
     2            +F3*XMVM(IM,JP,N)*PRSM(IM,JP,N)
               PRSM(IP,JP,0)=PRSM(IP,JP,0)
     2            +F4*XMVM(IP,JP,N)*PRSM(IP,JP,N)
               PRSP(IM,JM,0)=PRSP(IM,JM,0)
     2            +F5*XMVP(IM,JM,N)*PRSP(IM,JM,N)
               PRSP(IP,JM,0)=PRSP(IP,JM,0)
     2            +F6*XMVP(IP,JM,N)*PRSP(IP,JM,N)
               PRSP(IM,JP,0)=PRSP(IM,JP,0)
     2            +F7*XMVP(IM,JP,N)*PRSP(IM,JP,N)
               PRSP(IP,JP,0)=PRSP(IP,JP,0)
     2            +F8*XMVP(IP,JP,N)*PRSP(IP,JP,N)
              ENDIF
              TMPM(IM,JM,0) = TMPM(IM,JM,0)
     2            +F1*XMVM(IM,JM,N)*TMPM(IM,JM,N)
              TMPM(IP,JM,0) = TMPM(IP,JM,0)
     2            +F2*XMVM(IP,JM,N)*TMPM(IP,JM,N)
              TMPM(IM,JP,0) = TMPM(IM,JP,0)
     2            +F3*XMVM(IM,JP,N)*TMPM(IM,JP,N)
              TMPM(IP,JP,0) = TMPM(IP,JP,0)
     2            +F4*XMVM(IP,JP,N)*TMPM(IP,JP,N)
              TMPP(IM,JM,0) = TMPP(IM,JM,0)
     2            +F5*XMVP(IM,JM,N)*TMPP(IM,JM,N)
              TMPP(IP,JM,0) = TMPP(IP,JM,0)
     2            +F6*XMVP(IP,JM,N)*TMPP(IP,JM,N)
              TMPP(IM,JP,0) = TMPP(IM,JP,0)
     2            +F7*XMVP(IM,JP,N)*TMPP(IM,JP,N)
              TMPP(IP,JP,0) = TMPP(IP,JP,0)
     2            +F8*XMVP(IP,JP,N)*TMPP(IP,JP,N)
1068         CONTINUE
C            The total energy variable is used to hold
C            advection volume in the remap step.
             IF (.NOT.LVOL) THEN
              F1 = VOLXP(IM)*VOLYP(JM)*VOLZP(KPLANE)
              F2 = VOLXP(IP)*VOLYP(JM)*VOLZP(KPLANE)
              F3 = VOLXP(IM)*VOLYP(JP)*VOLZP(KPLANE)
              F4 = VOLXP(IP)*VOLYP(JP)*VOLZP(KPLANE)
              F5 = VOLXP(IM)*VOLYP(JM)*VOLZP(KUP)
              F6 = VOLXP(IP)*VOLYP(JM)*VOLZP(KUP)
              F7 = VOLXP(IM)*VOLYP(JP)*VOLZP(KUP)
              F8 = VOLXP(IP)*VOLYP(JP)*VOLZP(KUP)
              FD = F1+F2+F3+F4+F5+F6+F7+F8
              F1 = F1/FD
              F2 = F2/FD
              F3 = F3/FD
```

```
            F4 = F4/FD
            F5 = F5/FD
            F6 = F6/FD
            F7 = F7/FD
            F8 = F8/FD
            CEM(IM,JM) = F1*CE(I0,J0)
            CEM(IP,JM) = F2*CE(I0,J0)
            CEM(IM,JP) = F3*CE(I0,J0)
            CEM(IP,JP) = F4*CE(I0,J0)
            CEP(IM,JM) = F5*CE(I0,J0)
            CEP(IP,JM) = F6*CE(I0,J0)
            CEP(IM,JP) = F7*CE(I0,J0)
            CEP(IP,JP) = F8*CE(I0,J0)
           ENDIF
         ENDIF

         IF (LDE) THEN
            F1 = VOLXP(IM)*VOLYP(JM)*VOLZP(KPLANE)
            F2 = VOLXP(IP)*VOLYP(JM)*VOLZP(KPLANE)
            F3 = VOLXP(IM)*VOLYP(JP)*VOLZP(KPLANE)
            F4 = VOLXP(IP)*VOLYP(JP)*VOLZP(KPLANE)
            F5 = VOLXP(IM)*VOLYP(JM)*VOLZP(KUP)
            F6 = VOLXP(IP)*VOLYP(JM)*VOLZP(KUP)
            F7 = VOLXP(IM)*VOLYP(JP)*VOLZP(KUP)
            F8 = VOLXP(IP)*VOLYP(JP)*VOLZP(KUP)
            FD = F1+F2+F3+F4+F5+F6+F7+F8
            F1 = F1/FD
            F2 = F2/FD
            F3 = F3/FD
            F4 = F4/FD
            F5 = F5/FD
            F6 = F6/FD
            F7 = F7/FD
            F8 = F8/FD
            DEM(IM,JM) = F1*DE(I0,J0)
            DEM(IP,JM) = F2*DE(I0,J0)
            DEM(IM,JP) = F3*DE(I0,J0)
            DEM(IP,JP) = F4*DE(I0,J0)
            DEP(IM,JM) = F5*DE(I0,J0)
            DEP(IP,JM) = F6*DE(I0,J0)
            DEP(IM,JP) = F7*DE(I0,J0)
            DEP(IP,JP) = F8*DE(I0,J0)
         ENDIF
C
C        split the velocities
C
         IF (LVEL) THEN
C
C  Determine momentum-averaged mid-plane velocities
C
            I0P1=MIN(I0+1,IMAX)
            F3=VX(I0,J0)
            F1=(XMSTM(IM,JM)+XMSTM(IM,JP)+XMSTP(IM,JM)+XMSTP(IM,JP))
     |            /XMAST(I0,J0)
            F2=(XMSTM(IP,JM)+XMSTM(IP,JP)+XMSTP(IP,JM)+XMSTP(IP,JP))
     |            /XMAST(I0,J0)
            F6=(PONE-F1)*F3+(PONE-F2)*VX(I0P1,J0)
            J0P1=MIN(J0+1,JMAX)
            F4=VY(I0,J0)
            F1=(XMSTM(IM,JM)+XMSTM(IP,JM)+XMSTP(IM,JM)+XMSTP(IP,JM))
     |            /XMAST(I0,J0)
            F2=(XMSTM(IM,JP)+XMSTM(IP,JP)+XMSTP(IM,JP)+XMSTP(IP,JP))
     |            /XMAST(I0,J0)
```

```
              F7=(PONE-F1)*F4+(PONE-F2)*VY(I0,J0P1)
              F5=VZ(I0,J0)
              F1=(XMSTM(IM,JM)+XMSTM(IP,JM)+XMSTM(IM,JP)+XMSTM(IP,JP))
     |             /XMAST(I0,J0)
              F2=(XMSTP(IM,JM)+XMSTP(IP,JM)+XMSTP(IM,JP)+XMSTP(IP,JP))
     |             /XMAST(I0,J0)
              F8=(PONE-F1)*F5+(PONE-F2)*VZU(I0,J0)
              VXM(IM,JM)=F3
              VXM(IM,JP)=F3
              VXP(IM,JM)=F3
              VXP(IM,JP)=F3
              VXM(IP,JM)=F6
              VXM(IP,JP)=F6
              VXP(IP,JM)=F6
              VXP(IP,JP)=F6
              VYM(IM,JM)=F4
              VYM(IP,JM)=F4
              VYP(IM,JM)=F4
              VYP(IP,JM)=F4
              VYM(IM,JP)=F7
              VYM(IP,JP)=F7
              VYP(IM,JP)=F7
              VYP(IP,JP)=F7
              VZM(IM,JM)=F5
              VZM(IP,JM)=F5
              VZM(IM,JP)=F5
              VZM(IP,JP)=F5
              VZP(IM,JM)=F8
              VZP(IP,JM)=F8
              VZP(IM,JP)=F8
              VZP(IP,JP)=F8
            ENDIF
C
C           if stress deviators are present,
C           split them
C
            IF ((LSDEV).AND.(NSDD.GT.1))  THEN
             DO 1110  NSD=1,NSDD
              SIJM(IM,JM,NSD) = SIJ(I0,J0,NSD)
              SIJM(IP,JM,NSD) = SIJ(I0,J0,NSD)
              SIJM(IM,JP,NSD) = SIJ(I0,J0,NSD)
              SIJM(IP,JP,NSD) = SIJ(I0,J0,NSD)
              SIJP(IM,JM,NSD) = SIJ(I0,J0,NSD)
              SIJP(IP,JM,NSD) = SIJ(I0,J0,NSD)
              SIJP(IM,JP,NSD) = SIJ(I0,J0,NSD)
              SIJP(IP,JP,NSD) = SIJ(I0,J0,NSD)
1110         CONTINUE
            ENDIF
C
C           if internal state variables are present,
C           split them
C
            IF ((LXTVAR).AND.(NAEXV.GT.0))  THEN
             DO 1125  NXV=1,NAEXV
              EXVM(IM,JM,NXV) = EXVAR(I0,J0,NXV)
              EXVM(IP,JM,NXV) = EXVAR(I0,J0,NXV)
              EXVM(IM,JP,NXV) = EXVAR(I0,J0,NXV)
              EXVM(IP,JP,NXV) = EXVAR(I0,J0,NXV)
              EXVP(IM,JM,NXV) = EXVAR(I0,J0,NXV)
              EXVP(IP,JM,NXV) = EXVAR(I0,J0,NXV)
              EXVP(IM,JP,NXV) = EXVAR(I0,J0,NXV)
              EXVP(IP,JP,NXV) = EXVAR(I0,J0,NXV)
1125         CONTINUE
```

```
          ENDIF

1900      CONTINUE
2000     CONTINUE
       ENDIF
C
C-----------------------------------------------------------------------
C
      RETURN
      END
```

Routine VOFSPLIT.F

```
      SUBROUTINE VOFSPLIT(KPLANE,  KUP,  KDOWN,
C  input
     |     IOFFS,  ISTRT,  IEND,  JOFFS,  JSTRT,  JEND,
     |     LVOL,
     |     PHIV,  PHIVL,  PHIVU,
     |     PHIM,  PHIML, PHIMU,
C  scratch (dimensioned IMAX)
     |     NMP,
C  scratch (dimensioned IMAX*(NUMMAT+1))
     |     IMP,  MATORD,  VOLSUM,
C  scratch (dimensioned IMAX*3*NUMMAT)
     |     GNRM,
C  scratch (dimensioned NUMMAT+1)
     |     PHIE,  PHIW,  PHIN,  PHIS,  PHIU,  PHID,
     |     DPHI,  ICAT1,  ICAT2,  ICAT3,
C  scratch (dimensioned 8*(NUMMAT+1))
     |     VOLSUMS,
C  scratch (dimensioned 8)
     |     PDISS,  PDISRS,  VOLS,  IPOS,
C  output
     |     PHIVM,  PHIVP,  PHIMM,  PHIMP)
C
C*************************************************
C
C  This subroutine partitions the material and void
C  volume fractions from the parent cells in a particular
C  kplane to the child cells in one quadrant of the
C  child kplane.  Successive calls to this routine will
C  split an entire parent kplane into the two child
C  kplanes in four quadrants.
C
C*************************************************
C
#include "impdoubl.h"
#include "dbcdim.h"
#include "dbcvol.h"
#include "dbcxyz.h"
#include "comint.h"
#include "iofils.h"
C
      DIMENSION PHIV(IMAX,JMAX),PHIVL(IMAX,JMAX),PHIVU(IMAX,JMAX),
     |      PHIM(IMAX,JMAX,NUMMAT),PHIML(IMAX,JMAX,NUMMAT),
     |      PHIMU(IMAX,JMAX,NUMMAT)
C
      DIMENSION NMP(IMAX),IMP(IMAX,NUMMAT+1),
     |      MATORD(IMAX,NUMMAT+1),VOLSUM(IMAX,NUMMAT+1),
     |      GNRM(IMAX,3,NUMMAT+1),PHIE(NUMMAT+1),PHIW(NUMMAT+1),
     |      PHIN(NUMMAT+1),PHIS(NUMMAT+1),PHIU(NUMMAT+1),
     |      PHID(NUMMAT+1),DPHI(NUMMAT+1),ICAT1(NUMMAT+1),
     |      ICAT2(NUMMAT+1),ICAT3(NUMMAT+1),VOLSUMS(8,NUMMAT+1),
     |      PDISS(8),PDISRS(8),VOLS(8),IPOS(8)
C
      DIMENSION PHIVM(IMAX,JMAX),PHIVP(IMAX,JMAX),
     |      PHIMM(IMAX,JMAX,NUMMAT),PHIMP(IMAX,JMAX,NUMMAT)
C
      LOGICAL LVOL
C
      PARAMETER(PZERO=0.D0,PONE=1.D0)
C
      IF(LVOL)THEN
         DO 20  JJ=JSTRT,JEND,2
            J0 = JJ/2 + JOFFS
            JM=MAX(JJ,1)
```

```
            JP=MIN(JJ+1,JMAX)
            DO 25  II=ISTRT,IEND,2
               I0 = II/2 + IOFFS
               IM=MAX(II,1)
               IP=MIN(II+1,IMAX)
               F1 = VOLXP(IM)*VOLYP(JM)*VOLZP(KPLANE)
               F2 = VOLXP(IP)*VOLYP(JM)*VOLZP(KPLANE)
               F3 = VOLXP(IM)*VOLYP(JP)*VOLZP(KPLANE)
               F4 = VOLXP(IP)*VOLYP(JP)*VOLZP(KPLANE)
               F5 = VOLXP(IM)*VOLYP(JM)*VOLZP(KUP)
               F6 = VOLXP(IP)*VOLYP(JM)*VOLZP(KUP)
               F7 = VOLXP(IM)*VOLYP(JP)*VOLZP(KUP)
               F8 = VOLXP(IP)*VOLYP(JP)*VOLZP(KUP)
               FD = F1+F2+F3+F4+F5+F6+F7+F8
               PHIV(I0,J0)=PHIV(I0,J0)/FD
               PHIVL(I0,J0)=PHIVL(I0,J0)/FD
               PHIVU(I0,J0)=PHIVU(I0,J0)/FD
               DO 30 MAT=1,NUMMAT
                  PHIM(I0,J0,MAT)=PHIM(I0,J0,MAT)/FD
                  PHIML(I0,J0,MAT)=PHIML(I0,J0,MAT)/FD
                  PHIMU(I0,J0,MAT)=PHIMU(I0,J0,MAT)/FD
 30            CONTINUE
 25         CONTINUE
 20      CONTINUE
      ENDIF
C
C  Initialize arrays
C
      DO 80 J=1,JMAX
         DO 90 I=1,IMAX
            PHIVM(I,J)=PZERO
            PHIVP(I,J)=PZERO
            DO 95 MAT=1,NUMMAT
               PHIMM(I,J,MAT)=PZERO
               PHIMP(I,J,MAT)=PZERO
 95         CONTINUE
 90      CONTINUE
 80   CONTINUE
C
C  loop over all rows
C
      DO 100 JJ=JSTRT,JEND,2
         J = JJ/2 + JOFFS
         DO 110 II=ISTRT,IEND,2
            I = II/2 + IOFFS
            IMAT=0
            IF(PHIV(I,J).NE.PZERO)THEN
               IF(PHIV(I,J).EQ.PONE)THEN
                  J1=MAX(JJ,1)
                  J1P1=MIN(JJ+1,JMAX)
                  I1=MAX(II,1)
                  I1P1=MIN(II+1,IMAX)
                  PHIVM(I1,J1)=PONE
                  PHIVM(I1P1,J1)=PONE
                  PHIVM(I1,J1P1)=PONE
                  PHIVM(I1P1,J1P1)=PONE
                  PHIVP(I1,J1)=PONE
                  PHIVP(I1P1,J1)=PONE
                  PHIVP(I1,J1P1)=PONE
                  PHIVP(I1P1,J1P1)=PONE
                  DO 115 MAT=1,NUMMAT
                     PHIMM(I1,J1,MAT)=PZERO
                     PHIMM(I1P1,J1,MAT)=PZERO
```

```
                  PHIMM(I1,J1P1,MAT)=PZERO
                  PHIMM(I1P1,J1P1,MAT)=PZERO
                  PHIMP(I1,J1,MAT)=PZERO
                  PHIMP(I1P1,J1,MAT)=PZERO
                  PHIMP(I1,J1P1,MAT)=PZERO
                  PHIMP(I1P1,J1P1,MAT)=PZERO
 115           CONTINUE
               NMP(I)=0
               GOTO110
            ELSE
               IMAT=IMAT+1
               IMP(I,IMAT)=0
            ENDIF
         ENDIF
         DO 120 MAT=1,NUMMAT
            IF(PHIM(I,J,MAT).NE.PZERO)THEN
               IF(PHIM(I,J,MAT).EQ.PONE)THEN
                  J1=MAX(JJ,1)
                  J1P1=MIN(JJ+1,JMAX)
                  I1=MAX(II,1)
                  I1P1=MIN(II+1,IMAX)
                  PHIMM(I1,J1,MAT)=PONE
                  PHIMM(I1P1,J1,MAT)=PONE
                  PHIMM(I1,J1P1,MAT)=PONE
                  PHIMM(I1P1,J1P1,MAT)=PONE
                  PHIMP(I1,J1,MAT)=PONE
                  PHIMP(I1P1,J1,MAT)=PONE
                  PHIMP(I1,J1P1,MAT)=PONE
                  PHIMP(I1P1,J1P1,MAT)=PONE
                  PHIVM(I1,J1)=PZERO
                  PHIVM(I1P1,J1)=PZERO
                  PHIVM(I1,J1P1)=PZERO
                  PHIVM(I1P1,J1P1)=PZERO
                  PHIVP(I1,J1)=PZERO
                  PHIVP(I1P1,J1)=PZERO
                  PHIVP(I1,J1P1)=PZERO
                  PHIVP(I1P1,J1P1)=PZERO
                  DO 130 MMAT=1,NUMMAT
                     IF(MMAT.EQ.MAT)GOTO130
                     PHIMM(I1,J1,MMAT)=PZERO
                     PHIMM(I1P1,J1,MMAT)=PZERO
                     PHIMM(I1,J1P1,MMAT)=PZERO
                     PHIMM(I1P1,J1P1,MMAT)=PZERO
                     PHIMP(I1,J1,MMAT)=PZERO
                     PHIMP(I1P1,J1,MMAT)=PZERO
                     PHIMP(I1,J1P1,MMAT)=PZERO
                     PHIMP(I1P1,J1P1,MMAT)=PZERO
 130              CONTINUE
                  NMP(I)=0
                  GOTO110
               ELSE
                  IMAT=IMAT+1
                  IMP(I,IMAT)=MAT
               ENDIF
            ENDIF
 120     CONTINUE
C
         NMP(I)=IMAT

 110  CONTINUE
C
C  For mixed material cells, determine the ordering of the
C  materials in the cell.
```

```
C
         CALL MORDER(KPLANE,  KUP,  KDOWN,  J,
C  input
     |        ISTRT,  IEND,  IOFFS,  IMP,  NMP,  PHIV,
     |        PHIVL,  PHIVU,  PHIM,  PHIML,  PHIMU,
C  output
     |        MATORD,
C  scratch (dimensioned NUMMAT+1)
     |        PHIE,  PHIW,  PHIN,  PHIS,  PHIU,  PHID,
     |        DPHI,  ICAT1,  ICAT2,  ICAT3)
C
C  Determine volume fraction sums and the direction
C  of interface plane normals for material interfaces
C  in this row.
C
         CALL GRADVOF(KPLANE,  KUP,  KDOWN,  J,
C  input
     |       ISTRT,  IEND,  IOFFS,  IMP,  NMP,  MATORD,
     |       PHIV,  PHIVL,  PHIVU,  PHIM,  PHIML,
     |       PHIMU,
C  output
     |       GNRM,  VOLSUM,
C  scratch (dimensioned NUMMAT+1)
     |       PHIE,  PHIW,  PHIN,  PHIS,  PHIU,
     |       PHID)
C
C  Using Youngs algorithm determine the distance from
C  a corner of the unit cube to the interface plane.
C  Then compute volume fractions for the eight subdivided
C  cells.
C
         CALL PARTVOF(JJ,  ISTRT,  IEND,  IOFFS,
C  input
     |       IMP,  NMP,  MATORD,
     |       GNRM,  VOLSUM,
C  output
     |       PHIVM,  PHIVP,  PHIMM,  PHIMP,
C  scratch (dimensioned 8*(NUMMAT+1))
     |       VOLSUMS,
C  scratch (dimensioned 8)
     |       PDISS,  PDISRS,  VOLS,  IPOS)
C
 100  CONTINUE
C
C
      IF(LVOL)THEN
         DO 200  JJ=JSTRT,JEND,2
            J0 = JJ/2 + JOFFS
            JM=MAX(JJ,1)
            JP=MIN(JJ+1,JMAX)
            DO 210  II=ISTRT,IEND,2
               I0 = II/2 + IOFFS
               IM=MAX(II,1)
               IP=MIN(II+1,IMAX)
               F1 = VOLXP(IM)*VOLYP(JM)*VOLZP(KPLANE)
               F2 = VOLXP(IP)*VOLYP(JM)*VOLZP(KPLANE)
               F3 = VOLXP(IM)*VOLYP(JP)*VOLZP(KPLANE)
               F4 = VOLXP(IP)*VOLYP(JP)*VOLZP(KPLANE)
               F5 = VOLXP(IM)*VOLYP(JM)*VOLZP(KUP)
               F6 = VOLXP(IP)*VOLYP(JM)*VOLZP(KUP)
               F7 = VOLXP(IM)*VOLYP(JP)*VOLZP(KUP)
               F8 = VOLXP(IP)*VOLYP(JP)*VOLZP(KUP)
               FD = F1+F2+F3+F4+F5+F6+F7+F8
```

```
               PHIV(I0,J0)=PHIV(I0,J0)*FD
               PHIVL(I0,J0)=PHIVL(I0,J0)*FD
               PHIVU(I0,J0)=PHIVU(I0,J0)*FD
               PHIVM(IM,JM)=PHIVM(IM,JM)*F1
               PHIVM(IP,JM)=PHIVM(IP,JM)*F2
               PHIVM(IM,JP)=PHIVM(IM,JP)*F3
               PHIVM(IP,JP)=PHIVM(IP,JP)*F4
               PHIVP(IM,JM)=PHIVM(IM,JM)*F5
               PHIVP(IP,JM)=PHIVM(IP,JM)*F6
               PHIVP(IM,JP)=PHIVM(IM,JP)*F7
               PHIVP(IP,JP)=PHIVM(IP,JP)*F8
               DO 220 MAT=1,NUMMAT
                  PHIM(I0,J0,MAT)=PHIM(I0,J0,MAT)*FD
                  PHIML(I0,J0,MAT)=PHIML(I0,J0,MAT)*FD
                  PHIMU(I0,J0,MAT)=PHIMU(I0,J0,MAT)*FD
                  PHIMM(IM,JM,MAT)=PHIMM(IM,JM,MAT)*F1
                  PHIMM(IP,JM,MAT)=PHIMM(IP,JM,MAT)*F2
                  PHIMM(IM,JP,MAT)=PHIMM(IM,JP,MAT)*F3
                  PHIMM(IP,JP,MAT)=PHIMM(IP,JP,MAT)*F4
                  PHIMP(IM,JM,MAT)=PHIMP(IM,JM,MAT)*F5
                  PHIMP(IP,JM,MAT)=PHIMP(IP,JM,MAT)*F6
                  PHIMP(IM,JP,MAT)=PHIMP(IM,JP,MAT)*F7
                  PHIMP(IP,JP,MAT)=PHIMP(IP,JP,MAT)*F8
 220           CONTINUE
 210        CONTINUE
 200     CONTINUE
      ENDIF
C
C
      RETURN
      END
```

Routine MORDER.F

```
      SUBROUTINE MORDER(KPLANE,  KUP,  KDOWN,  J,
C  input
     |        ISTRT,  IEND,  IOFFS,  IMP,  NMP,  PHIV,
     |        PHIVL,  PHIVU,  PHIM,  PHIML,  PHIMU,
C  output
     |        MATORD,
C  scratch (dimensioned NUMMAT+1)
     |        PHIE,  PHIW,  PHIN,  PHIS,  PHIU,  PHID,
     |        DPHI,  ICAT1,  ICAT2,  ICAT3)
C
C*************************************************
C
C  This subroutine determines the order of the
C  materials in mixed cells.
C
C*************************************************
C
#include "impdoubl.h"
#include "dbcdim.h"
#include "dbcvol.h"
#include "dbcxyz.h"
#include "comint.h"
#include "iofils.h"
C
C
      DIMENSION IMP(IMAX,NUMMAT),NMP(IMAX)
      DIMENSION PHIV(IMAX,JMAX),PHIVL(IMAX,JMAX),PHIVU(IMAX,JMAX)
      DIMENSION PHIM(IMAX,JMAX,NUMMAT),PHIML(IMAX,JMAX,NUMMAT)
      DIMENSION PHIMU(IMAX,JMAX,NUMMAT)
C
      DIMENSION MATORD(IMAX,NUMMAT+1)
C
      DIMENSION PHIE(NUMMAT+1),PHIW(NUMMAT+1),PHIN(NUMMAT+1)
      DIMENSION PHIS(NUMMAT+1),PHIU(NUMMAT+1),PHID(NUMMAT+1)
      DIMENSION ICAT1(NUMMAT+1),ICAT2(NUMMAT+1),ICAT3(NUMMAT+1)
      DIMENSION DPHI(NUMMAT+1)
C
      PARAMETER(PZERO=0.D0,PTWO=2.D0,PFOUR=4.D0)
C
C  main loop
C
      DO 100 II=ISTRT,IEND,2
         I=II/2+IOFFS
         NMAT=NMP(I)
         IF(NMAT.EQ.0)GOTO100
C
         IP3=MIN(I+3,IMAX)
         IP2=MIN(I+2,IMAX)
         IP1=MIN(I+1,IMAX)
         IM1=MAX(I-1,1)
         IM2=MAX(I-2,1)
         JP3=MIN(J+3,JMAX)
         JP2=MIN(J+2,JMAX)
         JP1=MIN(J+1,JMAX)
         JM1=MAX(J-1,1)
         JM2=MAX(J-2,1)
         KP3=MIN(KPLANE+3,KMAX)
         KP2=MIN(KPLANE+2,KMAX)
         KM2=MAX(KPLANE-2,1)
C
         DD=DDX(I)+DDX(IP1)
         DDE=(DDX(IP2)+DDX(IP3))/DD
         DDW=(DDX(IM1)+DDX(IM2))/DD
```

```
      DD=DDY(J)+DDY(JP1)
      DDN=(DDY(JP2)+DDY(JP3))/DD
      DDS=(DDY(JM1)+DDY(JM2))/DD
      DD=DDZ(KPLANE)+DDZ(KUP)
      DDU=(DDZ(KP2)+DDZ(KP3))/DD
      DDD=(DDZ(KDOWN)+DDZ(KM2))/DD
C
      DO 110 MMAT=1,NMAT
         MAT=IMP(I,MMAT)
         IF(MAT.EQ.0)THEN
C
            DENOM=(PTWO+DDS+DDN)*(PTWO+DDU+DDD)
C
            PHIW(MMAT)=(DDD*(DDS*PHIVL(IM1,JM1)+
     |                       DDN*PHIVL(IM1,JP1))+
     |                  DDU*(DDS*PHIVU(IM1,JM1)+
     |                       DDN*PHIVU(IM1,JP1))+
     |                  PTWO*(DDD*PHIVL(IM1,J)+
     |                        DDU*PHIVU(IM1,J)+
     |                        DDS*PHIV(IM1,JM1)+
     |                        DDN*PHIV(IM1,JP1))+
     |                  PFOUR*PHIV(IM1,J))/DENOM
C
            PHIE(MMAT)=(DDD*(DDS*PHIVL(IP1,JM1)+
     |                       DDN*PHIVL(IP1,JP1))+
     |                  DDU*(DDS*PHIVU(IP1,JM1)+
     |                       DDN*PHIVU(IP1,JP1))+
     |                  PTWO*(DDD*PHIVL(IP1,J)+
     |                        DDU*PHIVU(IP1,J)+
     |                        DDS*PHIV(IP1,JM1)+
     |                        DDN*PHIV(IP1,JP1))+
     |                  PFOUR*PHIV(IM1,J))/DENOM
C
            DENOM=(PTWO+DDW+DDE)*(PTWO+DDU+DDD)
C
            PHIS(MMAT)=(DDD*(DDW*PHIVL(IM1,JM1)+
     |                       DDE*PHIVL(IP1,JM1))+
     |                  DDU*(DDW*PHIVU(IM1,JM1)+
     |                       DDE*PHIVU(IP1,JM1))+
     |                  PTWO*(DDD*PHIVL(I,JM1)+
     |                        DDU*PHIVU(I,JM1)+
     |                        DDW*PHIV(IM1,JM1)+
     |                        DDE*PHIV(IP1,JM1))+
     |                  PFOUR*PHIV(I,JM1))/DENOM
C
            PHIN(MMAT)=(DDD*(DDW*PHIVL(IM1,JP1)+
     |                       DDE*PHIVL(IP1,JP1))+
     |                  DDU*(DDW*PHIVU(IM1,JP1)+
     |                       DDE*PHIVU(IP1,JP1))+
     |                  PTWO*(DDD*PHIVL(I,JP1)+
     |                        DDU*PHIVU(I,JP1)+
     |                        DDW*PHIV(IM1,JP1)+
     |                        DDE*PHIV(IP1,JP1))+
     |                  PFOUR*PHIV(I,JP1))/DENOM
C
            DENOM=(PTWO+DDW+DDE)*(PTWO+DDS+DDN)
C
            PHID(MMAT)=(DDS*(DDW*PHIVL(IM1,JM1)+
     |                       DDE*PHIVL(IP1,JM1))+
     |                  DDN*(DDW*PHIVL(IM1,JP1)+
     |                       DDE*PHIVL(IP1,JP1))+
     |                  PTWO*(DDS*PHIVL(I,JM1)+
     |                        DDN*PHIVL(I,JP1)+
```

```
     |                          DDW*PHIVL(IM1,J)+
     |                          DDE*PHIVL(IP1,J))+
     |                     PFOUR*PHIVL(I,J))/DENOM
C
               PHIU(MMAT)=(DDS*(DDW*PHIVU(IM1,JM1)+
     |                         DDE*PHIVU(IP1,JM1))+
     |                     DDN*(DDW*PHIVU(IM1,JP1)+
     |                         DDE*PHIVU(IP1,JP1))+
     |                     PTWO*(DDS*PHIVU(I,JM1)+
     |                          DDN*PHIVU(I,JP1)+
     |                          DDW*PHIVU(IM1,J)+
     |                          DDE*PHIVU(IP1,J))+
     |                     PFOUR*PHIVL(I,J))/DENOM
C
            ELSE
C
               DENOM=(PTWO+DDS+DDN)*(PTWO+DDU+DDD)
C
               PHIW(MMAT)=(DDD*(DDS*PHIML(IM1,JM1,MAT)+
     |                         DDN*PHIML(IM1,JP1,MAT))+
     |                     DDU*(DDS*PHIMU(IM1,JM1,MAT)+
     |                         DDN*PHIMU(IM1,JP1,MAT))+
     |                     PTWO*(DDD*PHIML(IM1,J,MAT)+
     |                          DDU*PHIMU(IM1,J,MAT)+
     |                          DDS*PHIM(IM1,JM1,MAT)+
     |                          DDN*PHIM(IM1,JP1,MAT))+
     |                     PFOUR*PHIM(IM1,J,MAT))/DENOM
C
               PHIE(MMAT)=(DDD*(DDS*PHIML(IP1,JM1,MAT)+
     |                         DDN*PHIML(IP1,JP1,MAT))+
     |                     DDU*(DDS*PHIMU(IP1,JM1,MAT)+
     |                         DDN*PHIMU(IP1,JP1,MAT))+
     |                     PTWO*(DDD*PHIML(IP1,J,MAT)+
     |                          DDU*PHIMU(IP1,J,MAT)+
     |                          DDS*PHIM(IP1,JM1,MAT)+
     |                          DDN*PHIM(IP1,JP1,MAT))+
     |                     PFOUR*PHIM(IP1,J,MAT))/DENOM
C
               DENOM=(PTWO+DDW+DDE)*(PTWO+DDU+DDD)
C
               PHIS(MMAT)=(DDD*(DDW*PHIML(IM1,JM1,MAT)+
     |                         DDE*PHIML(IP1,JM1,MAT))+
     |                     DDU*(DDW*PHIMU(IM1,JM1,MAT)+
     |                         DDE*PHIMU(IP1,JM1,MAT))+
     |                     PTWO*(DDD*PHIML(I,JM1,MAT)+
     |                          DDU*PHIMU(I,JM1,MAT)+
     |                          DDW*PHIM(IM1,JM1,MAT)+
     |                          DDE*PHIM(IP1,JM1,MAT))+
     |                     PFOUR*PHIM(I,JM1,MAT))/DENOM
C
               PHIN(MMAT)=(DDD*(DDW*PHIML(IM1,JP1,MAT)+
     |                         DDE*PHIML(IP1,JP1,MAT))+
     |                     DDU*(DDW*PHIMU(IM1,JP1,MAT)+
     |                         DDE*PHIMU(IP1,JP1,MAT))+
     |                     PTWO*(DDD*PHIML(I,JP1,MAT)+
     |                          DDU*PHIMU(I,JP1,MAT)+
     |                          DDW*PHIM(IM1,JP1,MAT)+
     |                          DDE*PHIM(IP1,JP1,MAT))+
     |                     PFOUR*PHIM(I,JP1,MAT))/DENOM
C
               DENOM=(PTWO+DDW+DDE)*(PTWO+DDS+DDN)
C
               PHID(MMAT)=(DDS*(DDW*PHIML(IM1,JM1,MAT)+
```

```
     |                         DDE*PHIML(IP1,JM1,MAT))+
     |                    DDN*(DDW*PHIML(IM1,JP1,MAT)+
     |                         DDE*PHIML(IP1,JP1,MAT))+
     |                    PTWO*(DDS*PHIML(I,JM1,MAT)+
     |                          DDN*PHIML(I,JP1,MAT)+
     |                          DDW*PHIML(IM1,J,MAT)+
     |                          DDE*PHIML(IP1,J,MAT))+
     |                    PFOUR*PHIML(I,J,MAT))/DENOM
C
               PHIU(MMAT)=(DDS*(DDW*PHIMU(IM1,JM1,MAT)+
     |                         DDE*PHIMU(IP1,JM1,MAT))+
     |                    DDN*(DDW*PHIMU(IM1,JP1,MAT)+
     |                         DDE*PHIMU(IP1,JP1,MAT))+
     |                    PTWO*(DDS*PHIMU(I,JM1,MAT)+
     |                          DDN*PHIMU(I,JP1,MAT)+
     |                          DDW*PHIMU(IM1,J,MAT)+
     |                          DDE*PHIMU(IP1,J,MAT))+
     |                    PFOUR*PHIMU(I,J,MAT))/DENOM
            ENDIF
C
 110     CONTINUE
C
C  Test to see in which direction the material
C  ordering will be based.
C
         DPHIXMAX=-PTWO
         DPHIXMIN=PTWO
         DPHIYMAX=-PTWO
         DPHIYMIN=PTWO
         DPHIZMAX=-PTWO
         DPHIZMIN=PTWO
         DO 120 MMAT=1,NMAT
            XTEST=PHIE(MMAT)-PHIW(MMAT)
            DPHIXMAX=MAX(DPHIXMAX,XTEST)
            DPHIXMIN=MIN(DPHIXMIN,XTEST)
            YTEST=PHIN(MMAT)-PHIS(MMAT)
            DPHIYMAX=MAX(DPHIYMAX,YTEST)
            DPHIYMIN=MIN(DPHIYMIN,YTEST)
            ZTEST=PHIU(MMAT)-PHID(MMAT)
            DPHIZMAX=MAX(DPHIZMAX,ZTEST)
            DPHIZMIN=MIN(DPHIZMIN,ZTEST)
 120     CONTINUE
C
         XTEST=DPHIXMAX-DPHIXMIN
         YTEST=DPHIYMAX-DPHIYMIN
         ZTEST=DPHIZMAX-DPHIZMIN
         XYZMAX=MAX(XTEST,YTEST,ZTEST)
C
         NCAT1=0
         NCAT2=0
         NCAT3=0
         IF(XYZMAX.EQ.XTEST)THEN
C
C  Use x direction ordering.  First group materials
C  according to categories.
C     Category 1:  some ahead, none behind.
C     Category 2:  some ahead and behind (or none ahead or behind).
C     Category 3:  none ahead, some behind.
C
            DO 130 MMAT=1,NMAT
               DPHI(MMAT)=PHIE(MMAT)-PHIW(MMAT)
               IF(PHIE(MMAT).GT.PZERO.AND.PHIW(MMAT).EQ.PZERO)THEN
                  NCAT1=NCAT1+1
```

```
                  ICAT1(NCAT1)=MMAT
               ELSEIF(PHIE(MMAT).EQ.PZERO.AND.PHIW(MMAT).GT.PZERO)THEN
                  NCAT3=NCAT3+1
                  ICAT3(NCAT3)=MMAT
               ELSE
                  NCAT2=NCAT2+1
                  ICAT2(NCAT2)=MMAT
               ENDIF
 130        CONTINUE
C
         ELSEIF(XYZMAX.EQ.YTEST)THEN
C
C  Use y direction ordering.
C
            DO 140 MMAT=1,NMAT
               DPHI(MMAT)=PHIN(MMAT)-PHIS(MMAT)
               IF(PHIN(MMAT).GT.PZERO.AND.PHIS(MMAT).EQ.PZERO)THEN
                  NCAT1=NCAT1+1
                  ICAT1(NCAT1)=MMAT
               ELSEIF(PHIN(MMAT).EQ.PZERO.AND.PHIS(MMAT).GT.PZERO)THEN
                  NCAT3=NCAT3+1
                  ICAT3(NCAT3)=MMAT
               ELSE
                  NCAT2=NCAT2+1
                  ICAT2(NCAT2)=MMAT
               ENDIF
 140        CONTINUE
C
         ELSE
C
C  Use z direction ordering.
C
            DO 150 MMAT=1,NMAT
               DPHI(MMAT)=PHIU(MMAT)-PHID(MMAT)
               IF(PHIU(MMAT).GT.PZERO.AND.PHID(MMAT).EQ.PZERO)THEN
                  NCAT1=NCAT1+1
                  ICAT1(NCAT1)=MMAT
               ELSEIF(PHIU(MMAT).EQ.PZERO.AND.PHID(MMAT).GT.PZERO)THEN
                  NCAT3=NCAT3+1
                  ICAT3(NCAT3)=MMAT
               ELSE
                  NCAT2=NCAT2+1
                  ICAT2(NCAT2)=MMAT
               ENDIF
 150        CONTINUE
C
         ENDIF
C
C  Order category 1 materials first, from largest
C  to smallest.
C
         NORD=0
 155     DPHIMAX=-PTWO
         IF(NORD.EQ.NCAT1)GOTO165
         DO 160 ICAT=1,NCAT1
            MMAT=ICAT1(ICAT)
            IF(DPHI(MMAT).GT.DPHIMAX)THEN
               DPHIMAX=DPHI(MMAT)
               NEXT=MMAT
            ENDIF
 160     CONTINUE
C
         NORD=NORD+1
```

```
         MATORD(I,NORD)=NEXT
         DPHI(NEXT)=-PTWO
         GOTO155
C
C  Now category 2, from largest to smallest.  Note
C  that fragments will be ordered according to their
C  material number (probably should be placed in
C  random order later).
C
 165     DPHIMAX=-PTWO
         IF(NORD.EQ.NCAT1+NCAT2)GOTO175
         DO 170 ICAT=1,NCAT2
            MMAT=ICAT2(ICAT)
            IF(DPHI(MMAT).GT.DPHIMAX)THEN
               DPHIMAX=DPHI(MMAT)
               NEXT=MMAT
            ENDIF
 170     CONTINUE
C
         NORD=NORD+1
         MATORD(I,NORD)=NEXT
         DPHI(NEXT)=-PTWO
         GOTO165
C
C  Now category 3 materials.
C
 175     DPHIMAX=-PTWO
         IF(NORD.EQ.NCAT1+NCAT2+NCAT3)GOTO200
         DO 180 ICAT=1,NCAT3
            MMAT=ICAT3(ICAT)
            IF(DPHI(MMAT).GT.DPHIMAX)THEN
               DPHIMAX=DPHI(MMAT)
               NEXT=MMAT
            ENDIF
 180     CONTINUE
C
         NORD=NORD+1
         MATORD(I,NORD)=NEXT
         DPHI(NEXT)=-PTWO
         GOTO175
C
 200     CONTINUE
C
 100  CONTINUE
C
C
      RETURN
      END
```

Routine GRADVOF.F

```
      SUBROUTINE GRADVOF(KPLANE,  KUP,  KDOWN,  J,
C  input
     |       ISTRT,  IEND,  IOFFS,  IMP,  NMP,  MATORD,
     |       PHIV,  PHIVL,  PHIVU,  PHIM,  PHIML,
     |       PHIMU,
C  output
     |       GNRM,  VOLSUM,
C  scratch (dimensioned NUMMAT+1)
     |       VOFE,  VOFW,  VOFN,  VOFS,  VOFU,
     |       VOFD)
C
C
C*************************************************
C
C  This subroutine sums the volume fractions using
C  the ordering determined in subroutine MORDER and
C  computes normals for each of the interface
C  planes.
C
C*************************************************
C
C
#include "impdoubl.h"
#include "dbcdim.h"
#include "dbcvol.h"
#include "dbcxyz.h"
#include "comint.h"
#include "iofils.h"
#include "vofrnd.h"
C
      DIMENSION IMP(IMAX,NUMMAT),NMP(IMAX),MATORD(IMAX,NUMMAT+1)
      DIMENSION PHIV(IMAX,JMAX),PHIVL(IMAX,JMAX),PHIVU(IMAX,JMAX)
      DIMENSION PHIM(IMAX,JMAX,NUMMAT),PHIML(IMAX,JMAX,NUMMAT)
      DIMENSION PHIMU(IMAX,JMAX,NUMMAT)
C
      DIMENSION GNRM(IMAX,3,NUMMAT),VOLSUM(IMAX,NUMMAT+1)
C
      DIMENSION VOFE(NUMMAT+1),VOFW(NUMMAT+1),VOFN(NUMMAT+1)
      DIMENSION VOFS(NUMMAT+1),VOFU(NUMMAT+1),VOFD(NUMMAT+1)
C
      PARAMETER(PZERO=0.D0,PONE=1.D0,PTWO=2.D0,PFOUR=4.D0,
     |      PHALF=PONE/PTWO)
C
C  main loop
C
      DO 100 II=ISTRT,IEND,2
         I=II/2+IOFFS
         NMAT=NMP(I)
         IF(NMAT.EQ.0)GOTO100
C
         IP3=MIN(I+3,IMAX)
         IP2=MIN(I+2,IMAX)
         IP1=MIN(I+1,IMAX)
         IM1=MAX(I-1,1)
         IM2=MAX(I-2,1)
         JP3=MIN(J+3,JMAX)
         JP2=MIN(J+2,JMAX)
         JP1=MIN(J+1,JMAX)
         JM1=MAX(J-1,1)
         JM2=MAX(J-2,1)
         KP3=MIN(KPLANE+3,KMAX)
         KP2=MIN(KPLANE+2,KMAX)
         KM2=MAX(KPLANE-2,1)
```

```
C
         DD=DDX(I)+DDX(IP1)
         DDE=(DDX(IP2)+DDX(IP3))/DD
         DDW=(DDX(IM1)+DDX(IM2))/DD
         DD=DDY(J)+DDY(JP1)
         DDN=(DDY(JP2)+DDY(JP3))/DD
         DDS=(DDY(JM1)+DDY(JM2))/DD
         DD=DDZ(KPLANE)+DDZ(KUP)
         DDU=(DDZ(KP2)+DDZ(KP3))/DD
         DDD=(DDZ(KDOWN)+DDZ(KM2))/DD
C
         SUMVOF=PZERO
         SUMVOFW=PZERO
         SUMVOFE=PZERO
         SUMVOFS=PZERO
         SUMVOFN=PZERO
         SUMVOFD=PZERO
         SUMVOFU=PZERO
C
C  Determine volume fraction sums based on the material
C  ordering.  The contents of volsum are:
C          volsum(i,1) = all volume fractions (should be one)
C          volsum(i,2) = all volume fractions except the first
C                        material in the matord array
C          volsum(i,3) = all volume fractions except the first
C                        two materials in the matord array
C           ... etc.
C  The arrays vofe, vofw, vofn, vofs, vofu and vofd are also
C  summed this way.
C
         DO 110 IORD=NMAT,1,-1
            MMAT=MATORD(I,IORD)
            MAT=IMP(I,MMAT)
            IF(MAT.EQ.0)THEN
C
               SUMVOF=SUMVOF+PHIV(I,J)
C
               DENOM=(PTWO+DDS+DDN)*(PTWO+DDU+DDD)
C
               SUMVOFW=SUMVOFW+
     |            (DDD*(DDS*PHIVL(IM1,JM1)+
     |                  DDN*PHIVL(IM1,JP1))+
     |             DDU*(DDS*PHIVU(IM1,JM1)+
     |                  DDN*PHIVU(IM1,JP1))+
     |             PTWO*(DDD*PHIVL(IM1,J)+
     |                   DDU*PHIVU(IM1,J)+
     |                   DDS*PHIV(IM1,JM1)+
     |                   DDN*PHIV(IM1,JP1))+
     |             PFOUR*PHIV(IM1,J))/DENOM
C
               SUMVOFE=SUMVOFE+
     |            (DDD*(DDS*PHIVL(IP1,JM1)+
     |                  DDN*PHIVL(IP1,JP1))+
     |             DDU*(DDS*PHIVU(IP1,JM1)+
     |                  DDN*PHIVU(IP1,JP1))+
     |             PTWO*(DDD*PHIVL(IP1,J)+
     |                   DDU*PHIVU(IP1,J)+
     |                   DDS*PHIV(IP1,JM1)+
     |                   DDN*PHIV(IP1,JP1))+
     |             PFOUR*PHIV(IP1,J))/DENOM
C
               DENOM=(PTWO+DDW+DDE)*(PTWO+DDU+DDD)
C
```

```
                SUMVOFS=SUMVOFS+
     |              (DDD*(DDW*PHIVL(IM1,JM1)+
     |                   DDE*PHIVL(IP1,JM1))+
     |              DDU*(DDW*PHIVU(IM1,JM1)+
     |                   DDE*PHIVU(IP1,JM1))+
     |              PTWO*(DDD*PHIVL(I,JM1)+
     |                    DDU*PHIVU(I,JM1)+
     |                    DDW*PHIV(IM1,JM1)+
     |                    DDE*PHIV(IP1,JM1))+
     |              PFOUR*PHIV(I,JM1))/DENOM
C
                SUMVOFN=SUMVOFN+
     |              (DDD*(DDW*PHIVL(IM1,JP1)+
     |                   DDE*PHIVL(IP1,JP1))+
     |              DDU*(DDW*PHIVU(IM1,JP1)+
     |                   DDE*PHIVU(IP1,JP1))+
     |              PTWO*(DDD*PHIVL(I,JP1)+
     |                    DDU*PHIVU(I,JP1)+
     |                    DDW*PHIV(IM1,JP1)+
     |                    DDE*PHIV(IP1,JP1))+
     |              PFOUR*PHIV(I,JP1))/DENOM
C
                DENOM=(PTWO+DDW+DDE)*(PTWO+DDS+DDN)
C
                SUMVOFD=SUMVOFD+
     |              (DDS*(DDW*PHIVL(IM1,JM1)+
     |                   DDE*PHIVL(IP1,JM1))+
     |              DDN*(DDW*PHIVL(IM1,JP1)+
     |                   DDE*PHIVL(IP1,JP1))+
     |              PTWO*(DDS*PHIVL(I,JM1)+
     |                    DDN*PHIVL(I,JP1)+
     |                    DDW*PHIVL(IM1,J)+
     |                    DDE*PHIVL(IP1,J))+
     |               PFOUR*PHIVL(I,J))/DENOM
C
                SUMVOFU=SUMVOFU+
     |              (DDS*(DDW*PHIVU(IM1,JM1)+
     |                   DDE*PHIVU(IP1,JM1))+
     |              DDN*(DDW*PHIVU(IM1,JP1)+
     |                   DDE*PHIVU(IP1,JP1))+
     |              PTWO*(DDS*PHIVU(I,JM1)+
     |                    DDN*PHIVU(I,JP1)+
     |                    DDW*PHIVU(IM1,J)+
     |                    DDE*PHIVU(IP1,J))+
     |               PFOUR*PHIVU(I,J))/DENOM
C
             ELSE
C
                SUMVOF=SUMVOF+PHIM(I,J,MAT)
C
                DENOM=(PTWO+DDS+DDN)*(PTWO+DDU+DDD)
C
                SUMVOFW=SUMVOFW+
     |             (DDD*(DDS*PHIML(IM1,JM1,MAT)+
     |                   DDN*PHIML(IM1,JP1,MAT))+
     |              DDU*(DDS*PHIMU(IM1,JM1,MAT)+
     |                   DDN*PHIMU(IM1,JP1,MAT))+
     |              PTWO*(DDD*PHIML(IM1,J,MAT)+
     |                    DDU*PHIMU(IM1,J,MAT)+
     |                    DDS*PHIM(IM1,JM1,MAT)+
     |                    DDN*PHIM(IM1,JP1,MAT))+
     |              PFOUR*PHIM(IM1,J,MAT))/DENOM
C
```

```
                SUMVOFE=SUMVOFE+
     |             (DDD*(DDS*PHIML(IP1,JM1,MAT)+
     |                   DDN*PHIML(IP1,JP1,MAT))+
     |              DDU*(DDS*PHIMU(IP1,JM1,MAT)+
     |                   DDN*PHIMU(IP1,JP1,MAT))+
     |              PTWO*(DDD*PHIML(IP1,J,MAT)+
     |                    DDU*PHIMU(IP1,J,MAT)+
     |                    DDS*PHIM(IP1,JM1,MAT)+
     |                    DDN*PHIM(IP1,JP1,MAT))+
     |              PFOUR*PHIM(IP1,J,MAT))/DENOM
C
                DENOM=(PTWO+DDW+DDE)*(PTWO+DDU+DDD)
C
                SUMVOFS=SUMVOFS+
     |              (DDD*(DDW*PHIML(IM1,JM1,MAT)+
     |                   DDE*PHIML(IP1,JM1,MAT))+
     |              DDU*(DDW*PHIMU(IM1,JM1,MAT)+
     |                   DDE*PHIMU(IP1,JM1,MAT))+
     |              PTWO*(DDD*PHIML(I,JM1,MAT)+
     |                    DDU*PHIMU(I,JM1,MAT)+
     |                    DDW*PHIM(IM1,JM1,MAT)+
     |                    DDE*PHIM(IP1,JM1,MAT))+
     |              PFOUR*PHIM(I,JM1,MAT))/DENOM
C
                SUMVOFN=SUMVOFN+
     |              (DDD*(DDW*PHIML(IM1,JP1,MAT)+
     |                   DDE*PHIML(IP1,JP1,MAT))+
     |              DDU*(DDW*PHIMU(IM1,JP1,MAT)+
     |                   DDE*PHIMU(IP1,JP1,MAT))+
     |              PTWO*(DDD*PHIML(I,JP1,MAT)+
     |                    DDU*PHIMU(I,JP1,MAT)+
     |                    DDW*PHIM(IM1,JP1,MAT)+
     |                    DDE*PHIM(IP1,JP1,MAT))+
     |              PFOUR*PHIM(I,JP1,MAT))/DENOM
C
                DENOM=(PTWO+DDW+DDE)*(PTWO+DDS+DDN)
C
                SUMVOFD=SUMVOFD+
     |              (DDS*(DDW*PHIML(IM1,JM1,MAT)+
     |                   DDE*PHIML(IP1,JM1,MAT))+
     |              DDN*(DDW*PHIML(IM1,JP1,MAT)+
     |                   DDE*PHIML(IP1,JP1,MAT))+
     |              PTWO*(DDS*PHIML(I,JM1,MAT)+
     |                    DDN*PHIML(I,JP1,MAT)+
     |                    DDW*PHIML(IM1,J,MAT)+
     |                    DDE*PHIML(IP1,J,MAT))+
     |               PFOUR*PHIML(I,J,MAT))/DENOM
C
                SUMVOFU=SUMVOFU+
     |              (DDS*(DDW*PHIMU(IM1,JM1,MAT)+
     |                   DDE*PHIMU(IP1,JM1,MAT))+
     |              DDN*(DDW*PHIMU(IM1,JP1,MAT)+
     |                   DDE*PHIMU(IP1,JP1,MAT))+
     |              PTWO*(DDS*PHIMU(I,JM1,MAT)+
     |                    DDN*PHIMU(I,JP1,MAT)+
     |                    DDW*PHIMU(IM1,J,MAT)+
     |                    DDE*PHIMU(IP1,J,MAT))+
     |               PFOUR*PHIMU(I,J,MAT))/DENOM
C
            ENDIF
C
            VOLSUM(I,IORD)=SUMVOF
            VOFW(IORD)=SUMVOFW
```

```
            VOFE(IORD)=SUMVOFE
            VOFS(IORD)=SUMVOFS
            VOFN(IORD)=SUMVOFN
            VOFD(IORD)=SUMVOFD
            VOFU(IORD)=SUMVOFU
C
 110     CONTINUE
C
         DENOM=VOLSUM(I,1)
         VOLSUM(I,1)=PONE
         DO 120 IORD=2,NMAT
            VOLSUM(I,IORD)=VOLSUM(I,IORD)/DENOM
 120     CONTINUE
C
C  Compute the components of the normal for the unit cube.
C
         DO 130 IORD=2,NMAT
            GRADX=VOFW(IORD)-VOFE(IORD)
            GRADY=VOFS(IORD)-VOFN(IORD)
            GRADZ=VOFD(IORD)-VOFU(IORD)
            GRADM=SQRT(GRADX**2+GRADY**2+GRADZ**2)
            I2=IORD-1
C
            IF(GRADM.EQ.PZERO)THEN
C
C  The remaining materials in this cell are either
C  isolated, or possibly form an interface parallel
C  to a coordinate direction.
C
               IG=0
               IF(VOFE(IORD).EQ.PZERO)IG=100
               IF(VOFN(IORD).EQ.PZERO)IG=IG+10
               IF(VOFU(IORD).EQ.PZERO)IG=IG+1
               IF(IG.GE.100)THEN
                  IF(IG.EQ.111)THEN
C
C  Remaining materials are isolated; assign normal to
C  a random direction.
C
                     GRADM=SQRT(RANDX**2+RANDY**2+RANDZ**2)
                     GNRM(I,1,I2)=RANDX/GRADM
                     GNRM(I,2,I2)=RANDY/GRADM
                     GNRM(I,3,I2)=RANDZ/GRADM
C
                  ELSEIF(IG.EQ.110)THEN
C
C  Assign normal to x-y direction
C
                     GNRM(I,1,I2)=SQRT(PHALF)
                     GNRM(I,2,I2)=SQRT(PHALF)
                     GNRM(I,3,I2)=PZERO
                  ELSEIF(IG.EQ.101)THEN
C
C  Assign normal to x-z direction
C
                     GNRM(I,1,I2)=SQRT(PHALF)
                     GNRM(I,2,I2)=PZERO
                     GNRM(I,3,I2)=SQRT(PHALF)
C
                  ELSE
C
C  Assign normal to x direction
C
```

```
                     GNRM(I,1,I2)=PONE
                     GNRM(I,2,I2)=PZERO
                     GNRM(I,3,I2)=PZERO
                  ENDIF
               ELSEIF(IG.GE.10)THEN
                  IF(IG.EQ.11)THEN
C
C  Assign normal to Y-z direction
C
                     GNRM(I,1,I2)=PZERO
                     GNRM(I,2,I2)=SQRT(PHALF)
                     GNRM(I,3,I2)=SQRT(PHALF)
C
                  ELSE
C
C  Assign normal to the y direction
C
                     GNRM(I,1,I2)=PZERO
                     GNRM(I,2,I2)=PONE
                     GNRM(I,3,I2)=PZERO
                  ENDIF
               ELSE
C
C  Assign normal to the Z direction
C
                  GNRM(I,1,I2)=PZERO
                  GNRM(I,2,I2)=PZERO
                  GNRM(I,3,I2)=PONE
               ENDIF
            ELSE
               GNRM(I,1,I2)=GRADX/GRADM
               GNRM(I,2,I2)=GRADY/GRADM
               GNRM(I,3,I2)=GRADZ/GRADM
            ENDIF
 130     CONTINUE
C
 100  CONTINUE
C
      RETURN
      END
```

Routine PARTVOF.F

```
      SUBROUTINE PARTVOF(JJ,  ISTRT,  IEND,  IOFFS,
C  input
     |       IMP,  NMP,  MATORD,
     |       GNRM,  VOLSUM,
C  output
     |       PHIVM,  PHIVP,  PHIMM,  PHIMP,
C  scratch (dimensioned 8*(NUMMAT+1))
     |       VOLSUMS,
C  scratch (dimensioned 8)
     |       PDISS,  PDISRS,  VOLS,  IPOS)
C
C
C*************************************************
C
C  This subroutine uses Youngs interface tracking
C  algorithm to determine the distance from a corner
C  of the unit cube to the interface plane.  Using this
C  distance, the volume fractions are properly
C  partitioned in the refined cells.
C
C*************************************************
C
#include "impdoubl.h"
#include "dbcxyz.h"
#include "comint.h"
#include "iofils.h"
C
      DIMENSION NMP(IMAX),IMP(IMAX,NUMMAT+1),MATORD(IMAX,NUMMAT+1),
     |      GNRM(IMAX,3,NUMMAT),VOLSUM(IMAX,NUMMAT+1)
C
      DIMENSION PHIVM(IMAX,JMAX),PHIVP(IMAX,JMAX),
     |      PHIMM(IMAX,JMAX,NUMMAT+1),PHIMP(IMAX,JMAX,NUMMAT+1)
C
      DIMENSION VOLSUMS(8,NUMMAT+1),PDISS(8),PDISRS(8),VOLS(8),
     |      IPOS(8)
C
      PARAMETER(PZERO=0.D0,PONE=1.D0,PTWO=2.D0,PTHREE=3.D0,
     |      PFOUR=4.D0,PSIX=6.D0,PTWELVE=12.D0,PHALF=PONE/PTWO,
     |      PTHIRD=PONE/PTHREE,PSMALL=1.D-14,
     |      PPI=3.141592653589793D0,PTWOPI=PTWO*PPI,
     |      PT75=PTHREE/PFOUR,P1P5=1.5D0,PFOURPI=PFOUR*PPI)

C
C  main loop
C
      DO 100 II=ISTRT,IEND,2
         I=II/2+IOFFS
         NMAT=NMP(I)
         IF(NMAT.EQ.0)GOTO100
         DO 110 IORD=2,NMAT
            I2=IORD-1
            VOL=VOLSUM(I,IORD)
            IVOL=1
            IF(VOL.GT.PHALF)THEN
               VOL=PONE-VOL
               IVOL=-1
            ENDIF
            GSUM=ABS(GNRM(I,1,I2))+ABS(GNRM(I,2,I2))+ABS(GNRM(I,3,I2))
            GMAG=SQRT(GNRM(I,1,I2)**2+GNRM(I,2,I2)**2+GNRM(I,3,I2)**2)
            GMAX=PZERO
            GMIN=PONE
            DO 120 INRM=1,3
               IF(ABS(GNRM(I,INRM,I2)).GT.ABS(GMAX))THEN
```

```
                 GMAX=GNRM(I,INRM,I2)
                 INRM3=INRM
              ENDIF
              IF(ABS(GNRM(I,INRM,I2)).LT.ABS(GMIN))THEN
                 GMIN=GNRM(I,INRM,I2)
                 INRM1=INRM
              ENDIF
 120        CONTINUE
            HNRM3=ABS(GMAX)/GMAG
            HNRM1=ABS(GMIN)/GMAG
            INRM2=6-INRM1-INRM3
            HNRM2=GSUM/GMAG-HNRM1-HNRM3
            GMAG=GSUM-HNRM3
C
C  Check for 2D OR 3D
C
            IF(HNRM1.EQ.PZERO)THEN
C
C  2D
C
              TEST=PTWO*VOL*HNRM3
              IF(HNRM2.GE.TEST)THEN
C
C  Intersection is a quadrilateral A
C
                 PDIS=SQRT(TEST*HNRM2)
C
              ELSE
C
C  Intersection is a quadrilateral B
C
                 PDIS=VOL*HNRM3+PHALF*HNRM2
C
              ENDIF
C
            ELSE
C
C  3D
C
              TEST=PSIX*HNRM1*HNRM2*HNRM3*VOL
              TESTTRI=HNRM1**3
              TESTQUAD=HNRM2**3-(HNRM2-HNRM1)**3
              IF(TEST.LE.TESTTRI)THEN
C
C  Intersection is a triangular section.
C
                 PDIS=TEST**3
C
              ELSEIF(TEST.LE.TESTQUAD)THEN
C
C  Intersection is a quadrilateral section A
C
                 PDIS=PHALF*HNRM1+SQRT(PTHIRD*TEST/HNRM1-
     |                HNRM1**2/PTWELVE)
C
              ELSE
C
C  Test for pentagon, hexagon or quadrilateral section B
C
                 TESTPNT1=HNRM3**3-(HNRM3-HNRM1)**3-(HNRM3-HNRM2)**3
                 TESTPNT2=(HNRM1+HNRM2)**3-HNRM2**3-HNRM1**3
                 IF(HNRM3.GE.GMAG)THEN
C
```

```
                    IF(TEST.LT.TESTPNT1)THEN
C
C  Intersection is a pentagon.  This solution
C  was copied from RLB routine vofidx.f
C
                       R=PTHREE*HNRM1*HNRM2*(PHALF*GMAG-HNRM3*VOL)
                       Q=PTWO*HNRM1*HNRM2
                       AR=R/(MAX(Q*SQRT(Q),PSMALL))
                       IF(ABS(AR).GT.PONE)THEN
                          ARG=SIGN(PONE,AR)
                       ELSE
                          ARG=AR
                       ENDIF
                       S=PTHIRD*(ACOS(ARG)-PTWOPI)
                       PDIS=GMAG+PTWO*SQRT(Q)*COS(S)
C
                    ELSE
C
C  Intersection is a hexagon.  This solution
C  was copied from RLB routine vofidx.f
C
                       A=PT75*(HNRM1**2-HNRM2**2+HNRM3**2)
     |                      -P1P5*(HNRM1*HNRM2+HNRM2*HNRM3+
     |                      +HNRM3*HNRM1)
                       B=PTHREE*HNRM1*HNRM2*HNRM3*(VOL-PHALF)
                       R=PTWO*SQRT(MAX(-PTHIRD*A,PZERO))
                       AR=PTHREE*B/SIGN(MAX(ABS(A*R),PSMALL),A*R)
                       IF(ABS(AR).GT.PONE)THEN
                          ARG=SIGN(PONE,AR)
                       ELSE
                          ARG=AR
                       ENDIF
                       S=PTHIRD*(ACOS(ARG)+PFOURPI)
                       PDIS=PHALF*(HNRM1+HNRM2+HNRM3)+R*COS(S)
C
                    ENDIF
C
                 ELSE
C
                    IF(TEST.LT.TESTPNT2)THEN
C
C  Intersection is a pentagon.  This solution
C  was copied from RLB routine vofidx.f
C
                       R=PTHREE*HNRM1*HNRM2*(PHALF*GMAG-HNRM3*VOL)
                       Q=PTWO*HNRM1*HNRM2
                       AR=R/(MAX(Q*SQRT(Q),PSMALL))
                       IF(ABS(AR).GT.PONE)THEN
                          ARG=SIGN(PONE,AR)
                       ELSE
                          ARG=AR
                       ENDIF
                       S=PTHIRD*(ACOS(ARG)-PTWOPI)
                       PDIS=GMAG+PTWO*SQRT(Q)*COS(S)
C
                    ELSE
C
C  Intersection is a quadrilateral section B
C
                       PDIS=VOL*HNRM3+PHALF*GMAG
C
                    ENDIF
C
```

```
                 ENDIF
              ENDIF
           ENDIF
C
C  Reverse direction if necessary
C
           IF(IVOL.LT.0)PDIS=GSUM-PDIS
C
C  Compute corner distances for the subdivided unit
C  cubes.
C
           TWOPDIS=PTWO*PDIS
           PDISS(1)=TWOPDIS
           PDISS(2)=TWOPDIS-HNRM1
           PDISS(3)=TWOPDIS-HNRM2
           PDISS(4)=PDISS(3)-HNRM1
           PDISS(5)=TWOPDIS-HNRM3
           PDISS(6)=PDISS(5)-HNRM1
           PDISS(7)=PDISS(5)-HNRM2
           PDISS(8)=PDISS(7)-HNRM1
C
           TWOPDIS=PTWO*(GSUM-PDIS)
           PDISRS(8)=TWOPDIS
           PDISRS(7)=TWOPDIS-HNRM1
           PDISRS(6)=TWOPDIS-HNRM2
           PDISRS(5)=PDISRS(6)-HNRM1
           PDISRS(4)=TWOPDIS-HNRM3
           PDISRS(3)=PDISRS(4)-HNRM1
           PDISRS(2)=PDISRS(4)-HNRM2
           PDISRS(1)=PDISRS(2)-HNRM1
C
           DO 140 ISUB=1,8
              IVOL=1
              PD1=PDISS(ISUB)
              PDR1=PDISRS(ISUB)
C
C  Check for completely full or empty
C
              IF(PD1.LE.PZERO)THEN
                 VOLS(ISUB)=FLOAT((-IVOL+1)/2)
              ELSEIF(PDR1.LE.PZERO)THEN
                 VOLS(ISUB)=FLOAT((IVOL+1)/2)
              ELSE
C
C  Check if corner distance exceeds upper limit;
C  reverse direction if necessary
C
                 IF(PD1.GT.GMAG)THEN
                    IF(PD1.GT.HNRM3)THEN
                       IVOL=-IVOL
                       PD1=PDR1
                    ENDIF
                 ENDIF
C
C  Check for 2D or 3D
C
                 IF(HNRM1.EQ.PZERO)THEN
C
C  2D
C
                    IF(PD1.LT.HNRM2)THEN
C
C  Intersection is a quadrilateral A
```

```
C
                        VS=PHALF*PD1**2/HNRM2/HNRM3
C
                     ELSE
C
C  Intersection is a quadrilateral B
C
                        VS=(PD1-PHALF*HNRM2)/HNRM3
C
                     ENDIF
C
                  ELSE
C
C  3D
C
                     TEST=PSIX*HNRM1*HNRM2*HNRM3
                     IF(PD1.LE.HNRM1)THEN
C
C  Intersection is a triangle.
C
                        VS=PD1**3/TEST
                     ELSEIF(PD1.LE.HNRM2)THEN
C
C  Intersection is a quadrilateral section A
C
                        VS=(PD1**3-(PD1-HNRM1)**3)/TEST
                     ELSE
                        IF(PD1.LE.GMAG)THEN
                           IF(PD1.LE.HNRM3)THEN
C
C  Intersection is a pentagon.
C
                              VS=(PD1**3-(PD1-HNRM1)**3-
     |                             (PD1-HNRM2)**3)/TEST

                           ELSE
C
C  Intersection is a hexagon.
C
                              VS=(PD1**3-(PD1-HNRM1)**3-(PD1-HNRM2)**3
     |                             -(PD1-HNRM3)**3)/TEST
                           ENDIF
                        ELSE
C
C  Intersection must be a quadrilateral section B
C
                           VS=(PD1-PHALF*GMAG)/HNRM3
                        ENDIF
                     ENDIF
                  ENDIF
                  VOLS(ISUB)=FLOAT((1-IVOL)/2)+VS*IVOL
               ENDIF
               IPOS(ISUB)=ISUB
 140        CONTINUE
C
C  Assign volume fractions to the proper cells based
C  on the orientation of the unit cube to the physical
C  coordinates.
C
C  Column 1 correction
C
            IF(INRM1.NE.1)THEN
               IF(INRM2.EQ.1)THEN
```

```
                  ITMP=IPOS(3)
                  IPOS(3)=IPOS(2)
                  IPOS(2)=ITMP
                  ITMP=IPOS(7)
                  IPOS(7)=IPOS(6)
                  IPOS(6)=ITMP
                  INRM2=INRM1
                  INRM1=1
               ELSE
                  ITMP=IPOS(2)
                  IPOS(2)=IPOS(5)
                  IPOS(5)=ITMP
                  ITMP=IPOS(4)
                  IPOS(4)=IPOS(7)
                  IPOS(7)=ITMP
                  INRM3=INRM1
                  INRM1=1
               ENDIF
            ENDIF
C
C  Column 2 correction
C
            IF(INRM2.NE.2)THEN
               ITMP=IPOS(3)
               IPOS(3)=IPOS(5)
               IPOS(5)=ITMP
               ITMP=IPOS(4)
               IPOS(4)=IPOS(6)
               IPOS(6)=ITMP
               INRM3=INRM2
               INRM2=2
            ENDIF
C
 150        CONTINUE
C
C  Now correct for sign changes
C
            IF(GNRM(I,1,I2).LT.PZERO)THEN
C
C  Do x reversal
C
               ITMP=IPOS(1)
               IPOS(1)=IPOS(2)
               IPOS(2)=ITMP
               ITMP=IPOS(3)
               IPOS(3)=IPOS(4)
               IPOS(4)=ITMP
               ITMP=IPOS(5)
               IPOS(5)=IPOS(6)
               IPOS(6)=ITMP
               ITMP=IPOS(7)
               IPOS(7)=IPOS(8)
               IPOS(8)=ITMP
            ENDIF
C
            IF(GNRM(I,2,I2).LT.PZERO)THEN
C
C  Do y reversal
C
               ITMP=IPOS(1)
               IPOS(1)=IPOS(3)
               IPOS(3)=ITMP
               ITMP=IPOS(2)
```

```
               IPOS(2)=IPOS(4)
               IPOS(4)=ITMP
               ITMP=IPOS(5)
               IPOS(5)=IPOS(7)
               IPOS(7)=ITMP
               ITMP=IPOS(6)
               IPOS(6)=IPOS(8)
               IPOS(8)=ITMP
            ENDIF
C
            IF(GNRM(I,3,I2).LT.PZERO)THEN
C
C  Do z reversal
C
               ITMP=IPOS(1)
               IPOS(1)=IPOS(5)
               IPOS(5)=ITMP
               ITMP=IPOS(2)
               IPOS(2)=IPOS(6)
               IPOS(6)=ITMP
               ITMP=IPOS(3)
               IPOS(3)=IPOS(7)
               IPOS(7)=ITMP
               ITMP=IPOS(4)
               IPOS(4)=IPOS(8)
               IPOS(8)=ITMP
            ENDIF
C
C  Now partition this summed volumes among
C  the subdivided cells.
C
            DO 160 ISUB=1,8
               VOLSUMS(ISUB,IORD)=VOLS(IPOS(ISUB))
 160        CONTINUE
C
 110     CONTINUE
C
         IM=MAX(II,1)
         JM=MAX(JJ,1)
         DO 170 ISUB=1,8
            MMAT=MATORD(I,1)
            MAT=IMP(I,MMAT)
            VS=PONE-VOLSUMS(ISUB,2)
            IF(VS.LT.PSMALL)VS=PZERO
            IF(ABS(VS-PONE).LT.PSMALL)VS=PONE
            IF(ISUB.LE.4)THEN
               IS=(ISUB/2*2-ISUB+1)+IM
               JS=(ISUB-1)/2+JM
               IF(MAT.EQ.0)THEN
                  PHIVM(IS,JS)=VS
               ELSE
                  PHIMM(IS,JS,MAT)=VS
               ENDIF
            ELSE
               IS=(ISUB/2*2-ISUB+1)+IM
               JS=(ISUB-5)/2+JM
               IF(MAT.EQ.0)THEN
                  PHIVP(IS,JS)=VS
               ELSE
                  PHIMP(IS,JS,MAT)=VS
               ENDIF
            ENDIF
            DO 180 IORD=2,NMAT-1
```

```
                MMAT=MATORD(I,IORD)
                MAT=IMP(I,MMAT)
                VS=VOLSUMS(ISUB,IORD)-VOLSUMS(ISUB,IORD+1)
                IF(VS.LT.PSMALL)VS=PZERO
                IF(ABS(VS-PONE).LT.PSMALL)VS=PONE
                IF(ISUB.LE.4)THEN
                   IS=(ISUB/2*2-ISUB+1)+IM
                   JS=(ISUB-1)/2+JM
                   IF(MAT.EQ.0)THEN
                      PHIVM(IS,JS)=VS
                   ELSE
                      PHIMM(IS,JS,MAT)=VS
                   ENDIF
                ELSE
                   IS=(ISUB/2*2-ISUB+1)+IM
                   JS=(ISUB-5)/2+JM
                   IF(MAT.EQ.0)THEN
                      PHIVP(IS,JS)=VS
                   ELSE
                      PHIMP(IS,JS,MAT)=VS
                   ENDIF
                ENDIF
 180         CONTINUE
             MMAT=MATORD(I,NMAT)
             MAT=IMP(I,MMAT)
             VS=VOLSUMS(ISUB,NMAT)
             IF(VS.LT.PSMALL)VS=PZERO
             IF(ABS(VS-PONE).LT.PSMALL)VS=PONE
             IF(ISUB.LE.4)THEN
                IS=(ISUB/2*2-ISUB+1)+IM
                JS=(ISUB-1)/2+JM
                IF(MAT.EQ.0)THEN
                   PHIVM(IS,JS)=VS
                ELSE
                   PHIMM(IS,JS,MAT)=VS
                ENDIF
             ELSE
                IS=(ISUB/2*2-ISUB+1)+IM
                JS=(ISUB-5)/2+JM
                IF(MAT.EQ.0)THEN
                   PHIVP(IS,JS)=VS
                ELSE
                   PHIMP(IS,JS,MAT)=VS
                ENDIF
             ENDIF
 170      CONTINUE
C
 100  CONTINUE
C
C
      RETURN
      END
```