Journal of Computational Physics **148,** 433–466 (1999)
Article ID jcph.1998.6125, available online at http://www.idealibrary.com on IDEAL®

Conservative Remapping and Region Overlays by Intersecting Arbitrary Polyhedra

Jeffrey Grandy

Lawrence Livermore National Laboratory, P.O. Box 808, Livermore, California 94551
E-mail: grandy@llnl.gov

Received April 16, 1998; revised October 8, 1998

An efficient algorithm for first-order grid intersections, by computing geometrically the intersection volume between donor and target zones, is developed for polyhedral meshes. We examine two applications of grid intersections. One application is first-order remapping, in which zone and node centered fields defined on a given mesh are transferred to a different mesh. The second application is region overlays, in which a region with homogeneous material properties is approximated by a grid of polyhedra and mapped onto an arbitrary hexahedral mesh, creating mixed zones on the boundary of the region. We demonstrate the use of this grid intersection algorithm within the framework of hydrodynamics simulations, and using a domain decomposed mesh, we study the feasibility of a parallel implementation. © 1999 Academic Press

Key Words: remapping; rezoning; computational fluid dynamics; computational geometry.

1. BACKGROUND

We examine two applications of polyhedral grid intersection calculations, as they relate to hydrodynamics simulations. One application is direct remapping, in which physical fields such as the mass density, which are defined on a "donor" mesh, are transferred to an unrelated "target" mesh. We assume that both meshes have been provided and focus on the geometric computations and applications here. Another application is to approximate a specific region, such as a sphere, with a set of polyhedral zones, to estimate the fraction of each target zone that is contained within the sphere. This feature is useful, for example, to initialize problems for multimaterial hydrodynamics codes, such that the mesh boundaries are not designed to specifically conform to the surface of the sphere, but instead, the boundary of the sphere is modeled using mixed zones, and interface reconstruction methods [1] are used during the hydrodynamics simulation to maintain the integrity of the material boundary.

We consider two grids, a donor mesh with N_d zones and a target mesh with N_t zones. For a general grid intersection, we compute the integral of some field quantity $q(\mathbf{x})$, defined within a zone z_d of the donor mesh, over the figure of intersection between zone z_d and a zone z_t on the target mesh. The result is

$$I(z_d; z_t) = \int_{\mathcal{P}(z_d;z_t)} d\mathbf{x}\, q(\mathbf{x}), \tag{1}$$

where $\mathcal{P}(z_d; z_t)$ is the figure of intersection. This is a two-phase calculation; one must first obtain boundaries of the figure $\mathcal{P}$, and subsequently evaluate the integral within $\mathcal{P}$. In a first-order grid intersection, we take the field to be constant throughout the donor zone

$$q(\mathbf{x}) = q_{z_d} \tag{2}$$

so that (1) becomes

$$I(z_d; z_t) = V(z_d; z_t)\, q_{z_d}, \tag{3}$$

where $V(z_d; z_t)$ is the volume of intersection between z_d and z_t.

Various methods have been utilized to perform grid intersection calculations, including geometric methods which are capable of giving an exact definition of $\mathcal{P}$, numerical methods, and sampling techniques. A two-dimensional grid intersection calculation by Horak [2] illustrates a sampling method in which a donor zone is represented by a set of points, each assigned a partial area of the donor zone. The target zone containing a sample point receives the area from the donor zone associated with that sample point. Another type of sampling calculation is to approximate the donor and target zones using Cartesian voxels (pixels) in three (two) dimensions and to perform an intersection between octrees (quadtrees) [3]. An advantage of sampling is that it can handle donor and target zones with complicated boundaries; a disadvantage is the slow convergence of the intersection volume (or area in $2d$) as the number of sample points increases.

Three-dimensional grid intersections have been utilized in a second-order remapping code by Dukowicz and Padial [4]. In their computation of the volume of intersection between hexahedral zones, they define the zone boundaries as the quadric surfaces that form the bilinear interpolations between the four corner nodes of each face. A hybrid algorithm is used to approximate the figure of intersection. Gauss' theorem is applied twice to convert the volume integral (1) to line integrals along the edges and curves on the boundary of $\mathcal{P}$. A geometric calculation of points of intersection between donor edges and target faces (in general, quadric surfaces) and vice versa is followed by a Runge–Kutta procedure to construct curves of intersection and numerical integration along these curves to complete the calculation of I. However, Dukowicz and Padial have also pointed out certain cases in which their algorithm runs into difficulty. Since this technique relies on surface–edge intersections to form the endpoints of surface–surface intersection curves, such curves which do not contain an endpoint on the edge, but instead are confined within the boundaries of the surface, are missed. In addition, self-intersecting zones or parity-inverted zones, which can occur in highly distorted meshes, can cause misidentification of intersections between edges and surfaces. A result of missed intersections is local loss of conservation of integrated field quantities, such as mass and momentum. Aware of these potential causes of failure, Dukowicz and Padial have implemented *post facto* checks of the results of the remap, and perturbed the mesh nodes and repeated the remap, where a failure is detected. However, such perturbation procedures are unreliable since there is no *a priori* guarantee that the modified mesh will produce a successful remap result, and mesh perturbation also systematically reduces the accuracy of the remap.

The primary purpose of this article is to develop a geometric algorithm in three dimensions that handles every possible intersection between donor and target polyhedral zones, producing an exact determination of $\mathcal{P}$. Our geometric method can also provide a sensible result when a zone within the mesh, or a group of zones, is self-intersecting. We handle various degenerate cases that would otherwise cause the computation of $V(z_d; z_t)$ to fail, without altering the positions of the donor or target nodes. We use filtering procedures that identify candidates for intersecting donor and target zones by comparing coordinates for these zones, and this procedure enables us to handle pathological meshes that contain self-intersecting zones. The exact details of the filtering depend on the types of grids; the filtering procedures for remapping (Section 3.3) and region overlays (Section 4.2) differ.

Since we focus on the geometric technique, we list applications of first-order grid intersections here. Dukowicz [5] has shown (in two dimensions) that repeated applications of first-order remapping can cause unacceptable levels of diffusion and we expect similar behavior in three dimensions. We will therefore emphasize single-use applications of remapping, as opposed to frequent use within a hydrodynamics simulation. Our algorithm is based on computing the volume of intersection between a triangular polyhedron and a tetrahedron, and by convention we choose the target grid to be composed of tetrahedral zones. In Section 3 we describe the generalization to hexahedral zones, and in Section 4 we utilize donor octahedra and target hexahedra. We can easily apply this method to other target grids with polyhedral zones by decomposing target zones into tets, and triangulating faces of donor polyhedra.

2. INTERSECTION OF A POLYHEDRON WITH A TETRAHEDRON

We describe the intersection volume calculation between a donor triangular polyhedron and a target tetrahedron in this section. We label the donor zone z_d, and, for convenience, define two labels for the target zone. We assume that each target zone z_t has been subdivided into $n_t(z_t)$ tets, labeled by k_t. The volume of intersection is denoted by $V(z_d; z_t, k_t)$. The geometry algorithm, as designed for nonconvex polyhedra, scales as $O(n_{f,d} n_t(z_t))$, where the donor polyhedron has $n_{f,d}$ facets and by Euler's formula

$$n_{e,d} = 3/2 n_{f,d} \tag{4}$$

edges. This geometry algorithm is therefore best suited for volumes of intersection between nonconvex polyhedra with relatively small numbers of facets, and for large, convex or piecewise convex polyhedra, other algorithms with better scaling are available [6]. For the remainder of Section 2 we will refer to donor facets and edges as *surfaces* and *segments*, to distinguish from the target.

The first step is to perform the affine transformation, to cast the target tet to the unit tetrahedron **U**, with corners at

$$\begin{aligned} O &= (0, 0, 0) \\ X &= (1, 0, 0) \\ Y &= (0, 1, 0) \\ Z &= (0, 0, 1). \end{aligned} \tag{5}$$

By defining a fourth coordinate

$$h = 1 - x - y - z, \tag{6}$$

we cast **U** as a regular tetrahedron in the hyperplane (6) in the four-dimensional $(xyzh)$ space, and therefore, most intersection calculations treat these four coordinates in a symmetric fashion. The determinant of the matrix $M(z_t, k_t)$ that performs this transformation from physical coordinates (x_0, y_0, z_0) to the **U**-frame coordinates (x, y, z) is

$$\det M(z_t, k_t) = 6V(z_t, k_t), \tag{7}$$

where $V(z_t, k_t)$ is the volume of constituent tet k_t within the target zone z_t. By transforming the donor zone, we find that

$$V(z_d; z_t, k_t) = \det M(z_t, k_t) v(z_d; z_t, k_t), \tag{8}$$

where $v(z_d; z_t, k_t)$ is the intersection volume between the transformed donor zone and **U**. Since the donor zone is a triangular polyhedron, we compute the volume associated with each donor surface f_d by defining a column by projecting the $z > 0$ portion of the triangle onto the $z = 0$ plane and finding the volume of intersection $v(z_d, f_d; z_t, k_t)$ between the column for the surface f_d and **U**, so that

$$v(z_d; z_t, k_t) = \sum_{f_d=1}^{n_{f,d}} v(z_d, f_d; z_t, k_t). \tag{9}$$

By definition, if the donor zone lies entirely in the $z < 0$ half plane, the volume v is zero. Also, if the donor zone is above **U** in the z-direction, the individual donor surfaces give nonzero v, but the volumes associated with all of the surfaces cancel. For a simple, positively oriented donor zone,

$$0 \le v(z_d; z_t, k_t) \le 1/6. \tag{10}$$

The calculation of v, for a given donor triangular surface in the **U** frame, requires determination of the polygon **A**, which lies in the interior of **U**, and the polygon **B**, which is the projection of the triangle from the $+z$ direction onto the $h = 0$ facet of **U** (Figure 1), and v is the sum

$$v = v(\mathbf{A}) + v(\mathbf{B}) \tag{11}$$

the volume between the polygons and the $z = 0$ plane. In Figure 1, a donor surface is represented by triangle PQR, with vertices (x_p, y_p, z_p), etc. The polygons for the triangle PQR satisfy the following identities:

$$\Pi_{z=0}(\mathbf{A}) \cup \Pi_{z=0}(\mathbf{B}) = \Pi_{z=0}(PQR \cap (z > 0)) \cap \mathbf{U}_2 \tag{12}$$

$$\Pi_{z=0}(\mathbf{A}) \cap \Pi_{z=0}(\mathbf{B}) = 0 \tag{13}$$

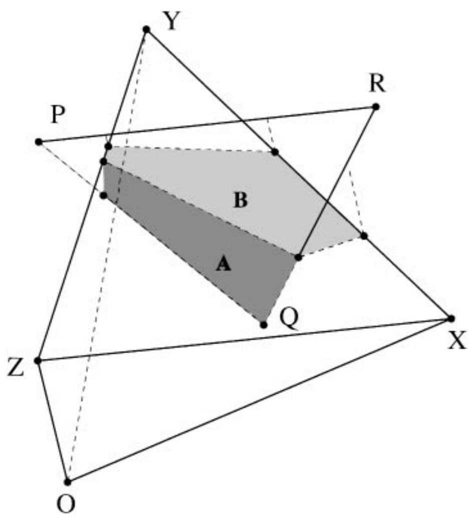

FIG. 1. Intersections between triangle *PQR* and **U** are vertices for polygon **A**, and the projection of *PQR* from the $+z$ direction onto the $h = 0$ facet defines polygon **B**.

where $\mathbf{U}_2$ is the triangle in the $z = 0$ plane, with corners O, X, and Y. The operator $\Pi_{z=0}$ denotes the projection onto the $z = 0$ plane. The first condition (12) states that the two polygons fill the area of the projection of the $z \geq 0$ part of *PQR* onto the $z = 0$ plane inside $\mathbf{U}_2$, and the second condition is that the polygons do not overlap. The intersections that define **A** are computed symmetrically in all four coordinates, and the coordinates z and h only play a special role in determining **B** since we have chosen to project through the $h = 0$ plane onto the $z = 0$ plane. Since v is computed from two-dimensional polygons, it is sufficient to identify the vertices of the polygons, and order them by angle relative to an interior point. If the same vertex is identified multiple times, the accuracy of v is not affected, but all vertices must be identified at least once. By finding volumes associated with individual donor triangular surfaces f_d, we also do not need to assume that the donor zone is convex. We first discuss the method for computing $v(z_d,\ f_d; z_t, k_t)$ with exact arithmetic, and subsequently show modifications needed for computer floating point arithmetic.

2.1. *Computation of Volume*

We describe the calculation of v for triangle *PQR* for the case of exact arithmetic with explicit handling of degenerate cases, where intersecting faces of *PQR* and **U** have total rank <3. If exact arithmetic were indeed available, we would preferably remove degeneracies explicitly by symbolic perturbation [7], but we present this discussion to provide a framework for computer arithmetic calculation of v. Our definitions for the faces of **U** are shown in Table I, and the faces of *PQR* are defined in Table II. Each face is defined to be the set of points generated by interpolating between the specified corners. We refer to the infinite extensions of facets and edges of **U** as planes and axes.

Intersection existence tests are based on the coordinates of P, Q, and R, and double and triple products combining these coordinates. The double products for segment *PQ*

TABLE I
Faces of U

Corners	Description	Polygons	Corners	Description	Polygons
O	O corner	**A**	Y, Z	yz-edge	**A**, **B**
X	X corner	**A**, **B**	Z, X	zx-edge	**A**, **B**
Y	Y corner	**A**, **B**	O, Y, Z	$x = 0$ facet	**A**
Z	Z corner	**A**, **B**	O, Z, X	$y = 0$ facet	**A**
O, X	x-edge	**A**	O, X, Y	$z = 0$ facet	**A**
O, Y	y-edge	**A**	X, Y, Z	$h = 0$ facet	**A**, **B**
O, Z	z-edge	**A**	O, X, Y, Z	Interior	**A**[a]
X, Y	xy-edge	**A**, **B**			

[a] Interior points with $h = 0$ are also in **B**.

are

$$\begin{aligned} c_{xy}^{pq} &= x_p y_q - y_p x_q \\ c_{yz}^{pq} &= y_p z_q - z_p y_q \\ c_{zx}^{pq} &= z_p x_q - x_p z_q \\ c_{xh}^{pq} &= x_p h_q - h_p x_q \\ c_{yh}^{pq} &= y_p h_q - h_p y_q \\ c_{zh}^{pq} &= z_p h_q - h_p z_q \end{aligned} \tag{14}$$

and the triple products for *PQR*, associated with the corners of **U**, are

$$\begin{aligned} t_O &= \begin{vmatrix} x_p & x_q & x_r \\ y_p & y_q & y_r \\ z_p & z_q & z_r \end{vmatrix}, \quad t_X = -\begin{vmatrix} h_p & h_q & h_r \\ y_p & y_q & y_r \\ z_p & z_q & z_r \end{vmatrix}, \\ t_Y &= -\begin{vmatrix} x_p & x_q & x_r \\ h_p & h_q & h_r \\ z_p & z_q & z_r \end{vmatrix}, \quad t_Z = -\begin{vmatrix} x_p & x_q & x_r \\ y_p & y_q & y_r \\ h_p & h_q & h_r \end{vmatrix}. \end{aligned} \tag{15}$$

We immediately eliminate the case where triangle *PQR* has no area and assign $v = 0$ in this case. A surface-edge intersection occurs if *PQR* surrounds the axis containing the edge (the double products with respect to the edge are identical sign) and the two corner endpoints

TABLE II
Faces of *PQR*

Corners	Description	Corners	Description
P	Vertex	*PQ*	Segment
Q	Vertex	*QR*	Segment
R	Vertex	*RP*	Segment
PQR	Surface		

TABLE III
Edges of U

Edge	Corners	Double product	Triple products
x	X, O	c_{yz}	t_X, t_O
y	Y, O	c_{zx}	t_Y, t_O
z	Z, O	c_{xy}	t_Z, t_O
yz	Y, Z	c_{xh}	t_Y, t_Z
zx	Z, X	c_{yh}	t_Z, t_X
xy	X, Y	c_{zh}	t_X, t_Y

are on opposite sides of the surface. For the x-edge, we define the Boolean variables

$$\begin{aligned} E_x(PQR) &= \left(c_{yz}^{pq}c_{yz}^{qr} > 0\right); \quad \left(c_{yz}^{pq}c_{yz}^{rp} > 0\right), \\ D_x(PQR) &= (t_O t_X \le 0); \qquad (t_O - t_X \ne 0), \end{aligned} \tag{16}$$

where $E_x(PQR)$ determines that the triangle surrounds the axis. $D_x(PQR)$ checks that corners O and X are not both in the same half space bounded by the plane of PQR and that both corners are not in the plane of PQR (both triple products zero). If one of the triple products is zero, the surface exactly intersects the corner, and this degenerate intersection is permitted by our definition of $D_x(PQR)$. The intersection is confirmed by

$$E_x(PQR) \quad \text{and} \quad D_x(PQR). \tag{17}$$

Tests for other edges are derived by replacing c_{yz} and the triple products in (16) with the appropriate combinations from Table III.

An analogous test confirms intersections between segments and **U** facets. The three edges on the facet must surround the segment, and the two vertices must be on opposite sides of the facet. To test for intersection with the $z = 0$ facet we define two Boolean variables,

$$\begin{aligned} F_z(PQ) &= \left(-c_{yz}^{pq}c_{zx}^{pq} > 0\right); \quad \left(-c_{yz}^{pq}c_{zh}^{pq} > 0\right), \\ S_z(PQ) &= (z_p z_q \le 0); \qquad (z_p \ne z_q), \end{aligned} \tag{18}$$

and the intersection test is

$$F_z(PQ) \quad \text{and} \quad S_z(PQ). \tag{19}$$

We list the double products which must all have the same sign and coordinates in the definitions for F and S for all four facets in Table IV.

The tests $F(PQ)$ include only nonzero double products, and we handle zero double products as a special case, testing for degenerate intersections. The test for PQ intersecting the x-edge (a degenerate case) is

$$\begin{gathered} \left(c_{yz}^{pq} = 0\right) \quad \text{and} \quad \left(c_{zx}^{pq}c_{zh}^{pq} > 0 \text{ or } c_{xy}^{pq}c_{yh}^{pq} < 0\right), \\ (S_y(PQ) \text{ or } S_z(PQ)) \end{gathered} \tag{20}$$

TABLE IV
Facets of U

Facet	Double products	Coordinates
$x=0$	$c_{xh}, c_{xy}, -c_{zx}$	x_p, x_q
$y=0$	$c_{yh}, c_{yz}, -c_{xy}$	y_p, y_q
$z=0$	$c_{zh}, c_{zx}, -c_{yz}$	z_p, z_q
$h=0$	c_{xh}, c_{yh}, c_{zh}	h_p, h_q

and tests for the other five edges are analogous. Finally, the segment PQ intersects the corner O if the double products for the three edges meeting at O are zero,

$$c_{xy}^{pq} = c_{yz}^{pq} = c_{zx}^{pq} = 0; \quad (S_x(PQ) \text{ or } S_y(PQ) \text{ or } S_z(PQ)), \tag{21}$$

and the same scheme applies for the other three corners. We interpolate the quantities used to test for intersections to compute locations of intersection points, in order to guarantee a nonzero denominator. For example, the location x^* of the surface intersection with the x-edge (17) is

$$x^* = t_O/(t_O - t_X) \tag{22}$$

which ensures that $0 \le x^* \le 1$. For a segment intersection with the $z=0$ facet (19), we obtain

$$\begin{aligned} s &= c_{yz}^{pq} - c_{zh}^{pq} - c_{zx}^{pq} \\ x^* &= -c_{zx}^{pq}/s \\ y^* &= c_{yz}^{pq}/s \end{aligned} \tag{23}$$

with the intersection tests guaranteeing that $s \ne 0$ and also

$$0 < x^* < 1, \quad 0 < y^* < 1, \quad 0 < x^* + y^* < 1. \tag{24}$$

For the degenerate case of the segment intersecting the x-edge, the intersection location is

$$x^* = \frac{c_{zx}^{pq}\left(c_{zx}^{pq} + c_{zh}^{pq}\right) + c_{xy}^{pq}\left(c_{xy}^{pq} - c_{yh}^{pq}\right)}{\left(c_{zx}^{pq} + c_{zh}^{pq}\right)^2 + \left(c_{xy}^{pq} - c_{yh}^{pq}\right)^2} \tag{25}$$

which, given (20), produces $0 < x^* < 1$ for exact arithmetic. Expressions for intersections on other facets and edges of **U** are derived by substituting the appropriate combinations of double and triple products from Tables III and IV into Eqs. (22), (23), (25). To compute vertices of **B**, we define a coordinate

$$H = 1 - x - y \tag{26}$$

and double products

$$\begin{aligned} c_{10}^{pq} &= y_p H_q - H_p y_q \\ c_{01}^{pq} &= H_p x_q - x_p H_q. \end{aligned} \tag{27}$$

The corner X is a vertex of **B** if the surface PQR intersects the ray pointing upward in the z direction from X, so that where

$$\begin{aligned} E_{10}(PQR) &= \left(c_{10}^{pq}c_{10}^{qr} > 0\right); \quad \left(c_{10}^{pq}c_{10}^{rp} > 0\right), \\ n_z &= c_{yz}^{pq} + c_{yz}^{qr} + c_{yz}^{rp} \\ D_{10}(PQR) &= (t_X n_z \geq 0), \end{aligned} \tag{28}$$

these conditions are satisfied:

$$E_{10}(PQR) \quad \text{and} \quad D_{10}(PQR). \tag{29}$$

Similar tests for the Y and Z corners are derived by replacing c_{10} with c_{01} or c_{xy}. The other nondegenerate case that provides a vertex of **B** is a segment intersecting the half-strip above one of the edges of **U** in the $h = 0$ plane, and for the zx-edge the following conditions establish the intersection:

$$\left(c_{xy}^{pq}c_{10}^{pq} < 0\right), \quad S_y(PQ), \quad \left(c_{yh}^{pq}c_{xy}^{pq} > 0\right). \tag{30}$$

The first two conditions are the standard two-dimensional test in the $z = 0$ plane for the intersection between PQ and the $y = 0$ edge of $\mathbf{U}_2$; the third test ensures that the segment passes above the zx-edge of **U**. The coordinates for the intersection vertex in (30) are

$$x^* = c_{xy}^{pq} \Big/ \left(c_{xy}^{pq} - c_{10}^{pq}\right), \quad y^* = 0, z^* = 1 - x^*. \tag{31}$$

For the segment passing above the corner X, a degenerate case, the tests are

$$\begin{gathered} (c_{10} = 0); \quad (S_y(PQ) \text{ or } S_H(PQ)), \\ \left(c_{yh}^{pq} + c_{zh}^{pq}\right)\left(c_{01}^{pq} - c_{xy}^{pq}\right) - c_{yh}^{pq}c_{xy}^{pq} < 0 \end{gathered} \tag{32}$$

with S_H defined analogously to S_z in (18). Finally, the vertex P is identified as a vertex of **A** if it lies in the interior of **U**, and P is also a vertex of **B** if it lies on the $h = 0$ facet of **U** (If $h = 0$ for the entire triangle, we remove **B** from (11) to avoid double counting.)

After the vertices of **A** and **B** have been found, the volume of the columns between these polygons and the $z = 0$ plane is computed. The barycenter $\mathbf{x}_c$ of the m vertices $\mathbf{x}_i$ of the polygon is selected as an interior point,

$$\mathbf{x}_c = (1/m)\sum_{i=0}^{m-1}\mathbf{x}_i \tag{33}$$

and the vertices are sorted in circular order around $\mathbf{x}_c$. The volume of the column under the polygon **A**, using the ordered vertices, is (where $\mathbf{x}_m = \mathbf{x}_0$)

$$v(\mathbf{A}) = \frac{1}{6}\sum_{i=0}^{m-1}(z_i + z_{i+1} + z_c)(x_i(y_{i+1} - y_c) + x_{i+1}(y_c - y_i) + x_c(y_i - y_{i+1})). \tag{34}$$

The types of nondegenerate intersections (excluding the trivial interior points) and their multiplicities are listed in Table V. Combining with (4), a total of $39/2n_{f,d}$ permutations occur in the tests for nondegenerate intersections. For example, if the donor zone is a tetrahedron ($n_{f,d} = 4$), there are 78 potential intersections excluding degeneracies.

TABLE V
Permutations of Intersections

Intersection	Polygons	Donor elements	**U** Elements
(17)	**A**, **B**	$n_{f,d}$ surfaces	6 edges
(19)	**A**, **B**	$n_{e,d}$ segments	4 facets
(29)	**B**	$n_{f,d}$ surfaces	3 rays
(30)	**B**	$n_{e,d}$ segments	3 half-strips

2.2. *Modifications for Computer Arithmetic*

In floating point arithmetic, which is inexact and nonassociative, errors can affect identification and computation of intersection points between *PQR* and **U**. For example, the triple product t_O in Eq. (15) may, for the same points P, Q, and R, give a result that is positive, zero, or negative, depending on how the (3 × 3) determinant is evaluated. Here we describe remedies for various roundoff errors. A computed quantity Q is

$$\begin{aligned} Q &= \hat{Q} + \delta(Q) \\ \delta(Q) &= \Delta(Q)\eta(Q), \end{aligned} \tag{35}$$

where $\hat{Q}$ is the value of Q from exact arithmetic, $\Delta(Q)$ is the upper bound of the error magnitude in the computation of Q, and $\eta(Q)$ is an unknown quantity in $-1 \le \eta(Q) \le 1$. Differentiating Eq. (8) gives

$$dV = v\,d(\det M) + \det M\,dv \tag{36}$$

and we focus on errors from v since det M can generally be computed to high precision unless zones are highly distorted. Schematically, the dependence of v on physical coordinates x_0, transformed coordinates x, double products c, and triple products t has the hierarchy:

$$v(x_0, x, c, t) = v(x_0, x(x_0), c(x_0, x), t(x_0, x, c)). \tag{37}$$

However, this can be simplified since c and d do not explicitly depend on x_0, giving

$$v(x_0, x, c, t) = v(x_0, x(x_0), c(x), t(x, c)) \tag{38}$$

and we use the chain rule to differentiate v:

$$\begin{aligned} dv(x_0, x, c, t) = (\partial v/\partial x)(dx/dx_0)\,dx_0 + (\partial v(x, c(x), t(x, c))/\partial c)\,dc \\ + (\partial v(x, c(x), t(x, c))/\partial t)\,dt. \end{aligned} \tag{39}$$

Thus, the error in computing the transformed coordinates x is separable from the error in obtaining the volume v from exact values of x. The first term in (39) is bounded geometrically since for some vertex P

$$|\nabla_P(v)| \le \sqrt{3}/2 \tag{40}$$

and for the last two terms, which contain computation errors in c and t, the **U**-frame coordinates for vertex i, x_i, y_i, z_i (and also h_i and H_i) are exact. Two kinds of errors that arise from computational noise are logic errors (a sign of a double or triple product changes, changing the intersection test and causing a discontinuity in v), and computation errors (inaccurate interpolated intersection point).

We discuss various remedies for logic and computation errors. We list the following cases (analogous definitions follow for the other faces of **U**):

- A segment PQ lies in the $h = 0$ plane if $h_p = h_q = 0$;
- A triangle PQR lies in the $h = 0$ plane if $h_p = h_q = h_r = 0$;
- A segment PQ lies on the x-edge if $y_p = y_q = z_p = z_q = 0$,

and initially exclude these cases. We have chosen not to use integer arithmetic since triple products require three multiplications, reducing the precision to less than 1/3 of the number of bits available for integers, and also because in our algorithm, the scale of triangle PQR is not bounded. We assume that there is no underflow, so the product of two nonzero quantities retains the correct sign.

As discussed earlier, the coordinates x, y, z, h, H are considered exact when we determine geometry errors in the **U** frame, the second and third terms of (39),

$$\Delta(x) = \Delta(y) = \Delta(z) = \Delta(h) = \Delta(H) = 0. \tag{41}$$

The computed double product c_{xy}^{pq} can obtain a large fractional error due to subtractive cancellation, because

$$\begin{aligned} c_{xy}^{pq} &= x_p y_q - y_p x_q \\ \Delta\left(c_{xy}^{pq}\right) &= f \cdot (|x_p y_q| + |y_p x_q|) \end{aligned} \tag{42}$$

and this can affect the computed sign of the double product when

$$\left|\hat{c}_{xy}^{pq}\right| < \Delta\left(c_{xy}^{pq}\right), \tag{43}$$

altering the tests that establish the existence of intersections. The tests (19), (20), (21) can fail to detect intersections if the sign patterns of the double products are inconsistent with the linear properties of the segment. We first examine the behavior of segment PQ with respect to an edge of **U** (Fig. 2a). The segment is oriented such that

$$c_{xh}^{pq} > 0, \quad c_{yh}^{pq} > 0, \quad c_{zh}^{pq} > 0, \quad c_{yz}^{pq} > 0, \quad c_{zx}^{pq} < 0, \tag{44}$$

and if $c_{xy}^{pq} = 0$ the segment intersects the z-edge at point E. A slight perturbation to c_{xy}^{pq} may occur due to roundoff error. A value $c_{xy}^{pq} > 0$ generates an intersection with the $x = 0$ facet of **U**, $c_{xy}^{pq} = 0$ intersects the z-edge, and $c_{xy}^{pq} < 0$ gives an intersection with the $y = 0$ facet (Table IV). Regardless of the sign of c_{xy}^{pq}, the segment intersects **U** somewhere in the neighborhood of E, and therefore the segment-edge behavior, even with inexact arithmetic, is consistent.

We now discuss the behavior of the segment with respect to a corner, where three edges of **U** meet. Since there are three double products and each may be positive, zero, or negative, there are 27 possible sign patterns for the double products, all of which are obtainable with

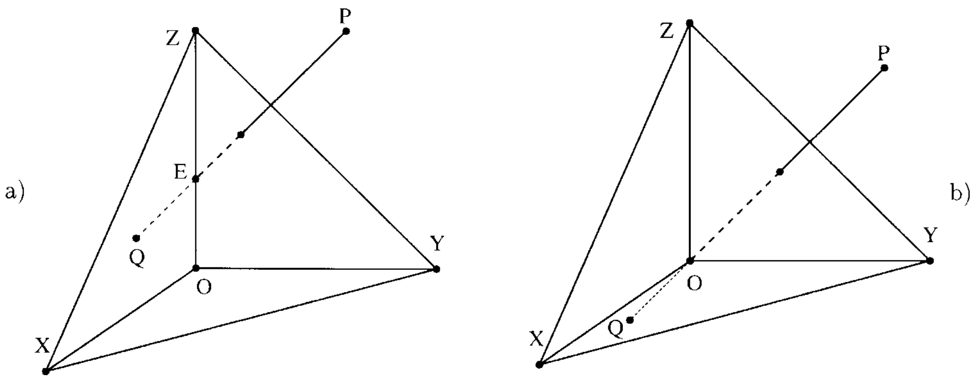

FIG. 2. Segment PQ has large cancellations in (a) the product c_{xy}^{pq} and (b) the products c_{yz}^{pq}, c_{zx}^{pq}, and c_{xy}^{pq} for the three edges that meet at O.

computer arithmetic. However, not all such combinations are geometrically consistent. Consider the segment PQ illustrated in Figure 2b, where P is in the first octant and Q is in the opposite octant, so that

$$c_{xh}^{pq} > 0, \quad c_{yh}^{pq} > 0, \quad c_{zh}^{pq} > 0. \tag{45}$$

In the illustration, c_{xy}^{pq}, c_{yz}^{pq}, and c_{zx}^{pq} are all subject to large cancellation errors since the segment passes near O. If the computed values of these double products are all positive, or all negative, the segment PQ fails to intersect any of the three facets meeting at O (Table IV). Therefore polygon **A** may miss the vertex in the vicinity of O, that would have been computed using exact arithmetic, causing an incorrect result for v. Also, if two of the three double products are zero, and the third is not, the geometry is inconsistent, and there are a total of 14 bad cases out of the 27 possible sign patterns. Individual tests for these cases at each corner are unnecessary, since there is a much simpler way to identify and eliminate them. The double products satisfy the algebraic relation

$$c_{yz}c_{xh} + c_{zx}c_{yh} + c_{xy}c_{zh} = 0. \tag{46}$$

For consistent geometry, either all three terms of (46) are zero, or a pair of nonzero terms has opposite sign. If the computed geometry is inconsistent, we set the three double products for the corner nearest to the infinite extension of PQ to zero, causing the segment to intersect the corner by (21). In addition, in order to facilitate triple product computation, we remove imprecise double products, identified by

$$\left|c_{xy}^{pq}\right| < (F/f)\Delta\left(c_{xy}^{pq}\right) \tag{47}$$

by setting these double products to zero. The selection of the cancellation factor F requires that we know something about the underlying computer arithmetic; we must guarantee that at least $F \geq f$, but larger values of F reduce the precision of the volume calculation. If a double product was not set to zero and $F \geq f$, its fractional error is less than f/F, and its sign is the same as the exact sign of the double product. We have used

$$F = 20f, \tag{48}$$

so that accepted nonzero double products have less than 5% roundoff error. As for computer precision, our ansatz is that

$$f = 4\epsilon_m, \tag{49}$$

where ϵ_m is the machine epsilon for the floating point arithmetic. Double products set to zero are tested for degenerate intersections.

Additional complications arise during the calculation of triple products, because in computer arithmetic the value of the result, and possibly its sign, depend on the computation formula. From Eq. (15) three different expressions for t_X are

$$t_X^{(x)} = -h_p c_{yz}^{qr} - h_q c_{yz}^{rp} - h_r c_{yz}^{pq} \tag{50}$$

$$t_X^{(zx)} = +z_p c_{yh}^{qr} + z_q c_{yh}^{rp} + z_r c_{yh}^{pq} \tag{51}$$

$$t_X^{(xy)} = -y_p c_{zh}^{qr} - y_q c_{zh}^{rp} - y_r c_{zh}^{pq}. \tag{52}$$

These expressions are associated with the x, zx, and yz edges of **U**, respectively. We compute triple products $t_X^{(x)}$ and $t_O^{(x)}$ for the x-edge only if the double products satisfy a modified version $\bar{E}_x(PQR)$ of the surround test $E_x(PQR)$ in (16):

$$\bar{E}_x(PQR) = \left(c_{yz}^{qr} c_{yz}^{rp} \geq 0\right), \quad \left(c_{yz}^{rp} c_{yz}^{pq} \geq 0\right), \quad \left(c_{yz}^{pq} c_{yz}^{qr} \geq 0\right); \tag{53}$$

i.e. no two double products are opposite signs. However, if the x-edge makes a small angle with the plane PQR, all three double products in $t_X^{(x)}$ may contain large cancellations. If we use $t_X^{(x)}$ and

$$t_O^{(x)} = x_p c_{yz}^{qr} + x_q c_{yz}^{rp} + x_r c_{yz}^{pq} \tag{54}$$

to compute x^*, we can write (22) as

$$\begin{aligned} x^* &= \xi_1 x_p + \xi_2 x_q + \xi_3 x_r \\ \xi_1 &= c_{yz}^{qr} \big/ \left(c_{yz}^{qr} + c_{yz}^{rp} + c_{yz}^{pq}\right), \quad \text{etc.} \\ \sum_{i=1}^{3} \xi_i &= 1, \end{aligned} \tag{55}$$

demonstrating that x^* is a barycentric interpolation between the x coordinates of the triangle vertices. We now define

$$\begin{aligned} y^* &= \xi_1 y_p + \xi_2 y_q + \xi_3 y_r \\ z^* &= \xi_1 z_p + \xi_2 z_q + \xi_3 z_r \end{aligned} \tag{56}$$

which are mathematically zero. Imprecise cross products can lead to an intersection point (x^*, y^*, z^*) which is within the triangle, but where the assumed x-edge intersection $(x^*, 0, 0)$ is outside the triangle, causing a violation of (12) and, hence, an erroneous v. Equation (55) only constrains x^* to be within the x range spanned by PQR but not within the part of PQR

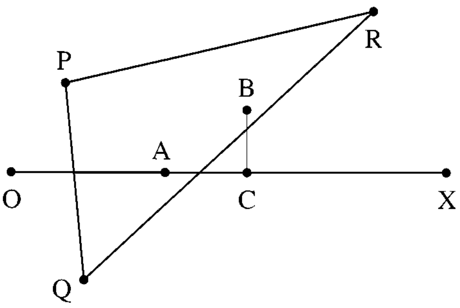

FIG. 3. The surface of triangle *PQR* intersects the x-edge at a very small angle. The normal of *PQR* points out of the paper, and *OX* has a small component out of the paper, passing below *PQ* and above *QR*. Exact arithmetic gives intersection *A*. Imprecise values of c_{yz} give a computed intersection *B*, and an assumed intersection point *C* on the x-edge.

that intersects the projection of the x-edge onto *PQR* (Fig. 3). We fix this interpolation by projecting out the y coordinate (we could have chosen z), creating new weights $\bar{c}_{yz}$ by

$$\begin{aligned}\alpha &= \frac{y_p c_{yz}^{qr} + y_q c_{yz}^{rp} + y_r c_{yz}^{pq}}{\left(y_p c_{yz}^{qr}\right)^2 + \left(y_q c_{yz}^{rp}\right)^2 + \left(y_r c_{yz}^{pq}\right)^2}\\ \bar{c}_{yz}^{qr} &= c_{yz}^{qr}\left(1 - \alpha y_p c_{yz}^{qr}\right)\\ \bar{c}_{yz}^{rp} &= c_{yz}^{rp}\left(1 - \alpha y_q c_{yz}^{rp}\right)\\ \bar{c}_{yz}^{pq} &= c_{yz}^{pq}\left(1 - \alpha y_r c_{yz}^{pq}\right)\end{aligned} \tag{57}$$

and substituting $\bar{c}$ values for c values in (52), (54) to compute $t_X^{(x)}$ and $t_O^{(x)}$. For edges on the $h = 0$ facet, we must project out the h coordinate in order to avoid cancellations that would cause violation of (13) when h_p, h_q, and h_r are all much less than one. If two or more c_{yz} are zero we disallow the x-edge intersection since (55) and (57) do not guarantee a correct x^*. Since we have selected F to allow at most 5% error in c values, in projection (57) the $\bar{c}$ values retain the same signs as c, but for arbitrary c values, the signs are not retained. Without using (47) to eliminate bad double products, (57) may give a $\bar{c}$ with a different sign from c, causing the interpolation that computes x^* to fail. If the surround test passes for more than one of the axes meeting at the corner ($\bar{E}_x(PQR)$, $\bar{E}_{zx}(PQR)$, and $\bar{E}_{xy}(PQR)$ for corner X), we select the edge with the smallest angle to *PQR* for the final value of t_X, preventing a triple product that is sensitive to (57) from being superseded by the value along another edge, which contains cancellations comparable to those in c_{yz}, cancellations whose effect on t_X has not been removed by (57). A single t value at each corner is crucial to defining a consistent set off D tests (16), (28) for all six axes and the rays above the axes in order to ensure that (13) is satisfied.

As described above, for the configuration in Fig. 2b the segment *PQ* intersects a facet, edge, or corner of **U**, because we have disallowed inconsistent cases. The other possibility for segment–corner behavior is that P and Q are in opposite octants external to **U** and R is located such that *PQR* intersects **U**. If *PQ* does not touch **U**, the surface–edge intersection coordinate x^* must be computed with the correct sign in order to ascertain the intersection since the D test discards intersections with $x^* < 0$. We use a method analogous to (47) to remove imprecise triple products, and a zero triple product permits an intersection exactly at the corner.

We now focus on the special cases listed earlier. If the segment PQ is in the $h = 0$ plane, the segment–edge tests (20) for the xy, yz, and zx edges reduce to the set of tests for intersections between a segment and a triangle in two dimensions. A vertex P or Q inside the triangle XYZ is treated as an interior point, and if the segment intersects a corner of XYZ, the corner test (21) is still satisfied since we have enforced geometric consistency. By extension, if the entire PQR is in the $h = 0$ plane, the calculation becomes mathematically equivalent to computing the area of intersection between two triangles, because the surface–edge test covers the cases, where PQR surrounds a corner. Since the intersection between a line segment and a corner in two dimensions is equivalent to the segment–edge behavior described above (by projecting along the edge so the edge collapses to a point) there are no geometric inconsistencies in this two-dimensional computation of the area of intersection. In addition, a projection such as (57) is not needed in two-dimensional zone intersections, because there is no need to constrain the range of coordinates when interpolating to locate an intersection between two segments. Therefore, as mentioned by Dukowicz [8] one may handle special cases in two-dimensional intersections in any of a number of ways, such as perturbing mesh points and/or double products (and not handling degenerate cases at all), or explicit logic in the code. Finally, if the segment lies on an axis (such as the x-axis) the only possible intersections are (21) if the segment passes through an endpoint of the edge, or interior points inside the edge. This is effectively a one-dimensional intersection, a trivial calculation.

To summarize, the geometric calculation of $V(z_d; z_t)$ has been facilitated by decomposing the target zones into simplices and the donor zones into triangular polyhedra. The linear transformation to the **U** frame has allowed us to treat all four facets of the target tet in a symmetric fashion, and the geometric closure condition (46) is explicitly symmetric in the four coordinates. Enforcement of consistency causes the assignment of zero double products, which must be handled as special cases, so that, unlike the two-dimensional case [8], we cannot simplify the algorithm by preemptively removing degeneracies. In addition, due to roundoff error in double products, a projection algorithm (57) must be applied in order to properly constrain the interpolation for intersections that depend on triple products.

3. APPLICATION: FIRST-ORDER DIRECT REMAPPING

Here we describe the use of grid intersections to perform first-order direct remapping. The distinction between direct (general) and incremental (continuous) remapping has been discussed by Dukowicz and Kodis [5, 8]. In incremental remaps, the target mesh is constructed by displacing the nodes of the donor mesh, and the displacement is generally small, compared with the spacing between nodes. During an adaptive Lagrangian–Eulerian (ALE) hydrodynamics simulation an iterative mesh relaxation algorithm such as the Winslow–Crowley [9] method is applied to reduce distortions in the mesh, followed by an incremental remap to reset the physical fields. Incremental remaps have been accomplished in two dimensions using geometric area of intersection calculations [10], but most ALE codes in two and three [11] dimensions use advection [12] to perform incremental remaps. Typically, an ALE code will perform a small number of relaxation and remap iterations per Lagrangian physics cycle. The incremental remaps are therefore performed quite frequently, perhaps one or more times per physics cycle during a hydrodynamics simulation and, therefore, the accuracy of the incremental remap has a major impact on the overall accuracy of the simulation.

A direct remap is the transfer of a physical system from one mesh to an unrelated target mesh. Direct remapping has been studied in two dimensions [8, 13, 14] and three dimensions [4]. For the $2d$ case, Dukowicz and Ramshaw [8, 13] have shown that the density field $q(\mathbf{x})$ defined within each donor zone may be represented as the divergence of a vector field, and by Gauss' theorem the integral of the field over the region of intersection between a donor and target zone may be replaced by line integrals over the boundaries of donor and target zones. In a different approach, Miller [14] defines the field as the curl of a potential and uses Stokes' theorem to define a line integral. Both of these approaches simplify the mathematical form of the remap by reducing the dimensionality of the integral, and when the field within a zone is a complicated analytic form requiring numerical integration to complete the remap, there is a clear advantage to using the divergence theorem to define a surface or line integral. In our case, for a first-order remap, $V(z_d; z_t)$ is computed exactly from the intersection points.

3.1. *Overview of Remap Procedure*

In this section we review various aspects of remapping. Since the donor zone boundaries do not conform with target zone boundaries, the target zone will acquire a mixture of materials if different intersecting donor zones contain different materials. For a mixed zone, the material number, volume fraction, and field quantities are stored separately for each component material within the zone (we find that a linked list is particularly convenient), and the volume fractions f_i for material i must satisfy the closure condition

$$\sum_m f_m = 1. \tag{58}$$

In a pure Lagrangian hydrodynamics simulation, the incoming donor zones are all clean (single material), unless mixed zones were in the initial state of the problem. We consider the case of clean donor zones in this article. However, the geometry algorithm (Section 2) is capable of handling figures other than hexahedra, as would be obtained by breaking a mixed donor zone into parts. Usually, we are interested in remapping zone-centered fields and node-centered fields together, since a hydrodynamics simulation may provide a combination of fields with different centerings, such as zone-centered mass density and node-centered momentum, and we bisect hexahedral zones in order to simultaneously remap these fields.

3.2. *Hexahedron Geometry*

Each hexahedral zone is associated with a list of eight corner nodes for each zone, and these nodes are ordered so that 12 edges may be inferred, connected topologically as a cube. A 3^3 subgrid (Fig. 4a) is constructed by interpolating between the corner nodes with the subgrid points parameterized by

$$\mathbf{x}(\alpha, \beta, \gamma) : (\alpha, \beta, \gamma) \in \{0, 1/2, 1\} \tag{59}$$

with a trilinear interpolation, so for example, in the α direction,

$$\mathbf{x}(1/2, \beta, \gamma) = 1/2(\mathbf{x}(0, \beta, \gamma) + \mathbf{x}(1, \beta, \gamma)) \quad \forall \beta, \gamma. \tag{60}$$

The interpolation parameters for the labeled points in 4a are given in Table VI.

TABLE VI
Labeled Interpolation Points, Fig. 4a

	(α, β, γ)		(α, β, γ)		(α, β, γ)
0	(0, 0, 0)	5	(1, 0, 1)	*c*	(0, 1/2, 1/2)
1	(0, 0, 1)	6	(1, 1, 0)	*d*	(1/2, 0, 0)
2	(0, 1, 0)	7	(1, 1, 1)	*e*	(1/2, 0, 1/2)
3	(0, 1, 1)	*a*	(0, 0, 1/2)	*f*	(1/2, 1/2, 0)
4	(1, 0, 0)	*b*	(0, 1/2, 0)	*g*	(1/2, 1/2, 1/2)

Each hex face comprises nine subgrid points, and the face is divided into eight triangular facets, as shown in Fig. 4b. The pairs of triangles on the same face sharing an edge are coplanar, so therefore our representation of the full zone, although comprising 48 triangles, encloses the same volume as a 24-faceted triangular polyhedron, a tetrakis hexahedron (TH). Dukowicz and Padial [4] have represented a hex face by a hyperboloid constructed from a bilinear interpolation between the nodes. However, the total volume of the zone with hyperboloid surfaces is the same as the TH zone volume, and many physics codes do not depend on the detailed shape of the zone boundary. The TH face may be considered as a nine-point approximation to the hyperboloid boundary, with a systematic difference

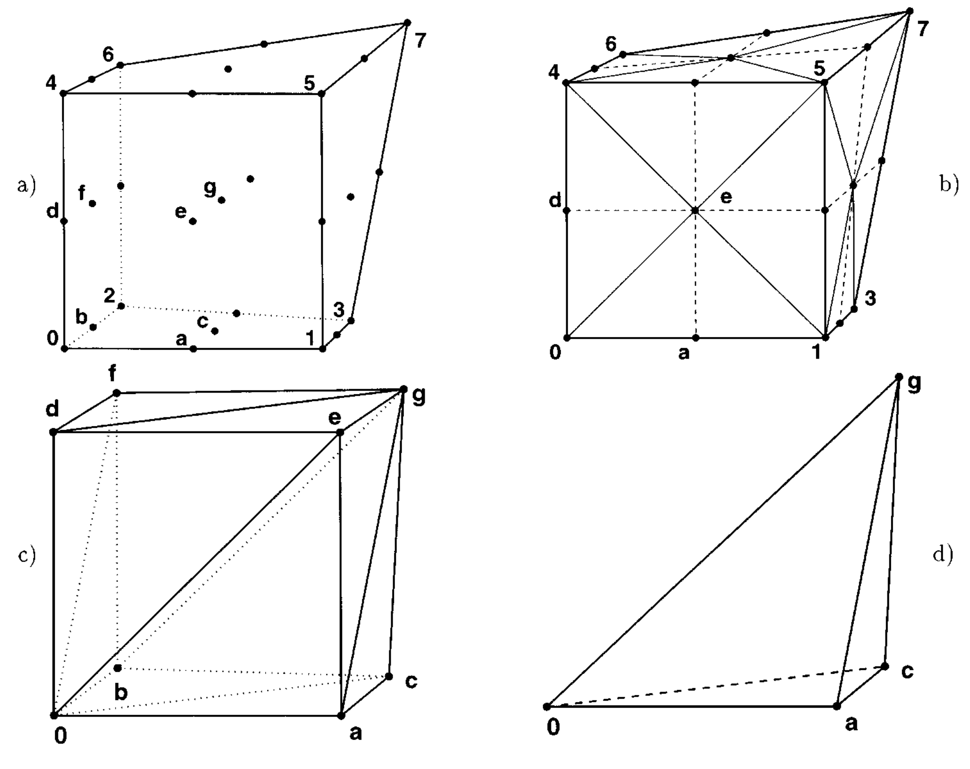

FIG. 4. Bisection of hexahedron: (a) full zone and 3^3 subgrid; (b) triangular facets on the front surfaces of this zone. Pairs of facets adjoining at dashed lines are coplanar: (c) lower left front subzone of zone (a); (d) one of the six tets in the decomposition of the subzone (c). The other five tets are 0*eag*, 0*cbg*, 0*bfg*, 0*deg*, and 0*fdg*.

between the two representations in the spatial distribution of physical field quantities, but not in the globally integrated quantities.

The next step is to complete a subzone. We begin with the six facets (out of 48) that meet at the node $\mathbf{x}_0 = \mathbf{x}(0, 0, 0)$. Eight points, the mesh node $\mathbf{x}_0$ and interpolated points $\mathbf{x}_a$ through $\mathbf{x}_g$ in Fig. 4a, are subzone vertices. The boundary of the subzone is completed by constructing six facets that connect the central point $\mathbf{x}_g$ with $\mathbf{x}_a$ through $\mathbf{x}_f$. The resulting subzone (Fig. 4c) contains 12 facets and 8 vertices, of which two vertices ($\mathbf{x}_0$ and $\mathbf{x}_g$) are of order 6 and the other six vertices are order 4. For a donor mesh with N_d hex zones and a target mesh with N_t hex zones, we actually map $8N_d$ subzones onto $8N_t$ subzones. We label subzones of the donor zone z_d as $z_{d,s}$. The total nodal volume, $v(n_d)$ of node n_d is the sum of volumes of associated subzones $z_{d,s}$, from different donor zones, that contain n_d as a corner,

$$v(n_d) = \sum_{z_{d,s}:n_d=n(z_{d,s})} v(z_{d,s}), \tag{61}$$

where $n(z_{d,s})$ is defined to be the mesh node that is a corner of the subzone $z_{d,s}$. In a structured mesh, eight subzones meeting at each node (except the boundary nodes) define the basis volume for mapping nodal fields. Central points of zones are corners of nodal basis volumes, and thus the nodal basis volume is a 48-faceted polyhedron with the same connectivity as the 48-faceted polyhedron as we have used to represent the zone volume. For the intersection volume calculation, as discussed in Section 2, the target subzone is decomposed into six tets using the long diagonal (LD) method (Fig. 4d). To decompose the subzone into five tets using the corner slicing method would effect a different definition of the facets and edges of the subzones, breaking the symmetry between the polyhedra for nodal and zonal basis volumes.

After decomposing the target subzone into tets, we compute $V(z_{d,s}; z_{t,s})$ using the method of Section 2. In the calculation of $V(z_{d,s}; z_{t,s})$ there are 234 permutations of nondegenerate intersection tests per target tet (12 donor facets, 39/2 permutations per facet), or a total of 1404 permutations per target subzone (six tets per subzone). In the two-dimensional generalization of this algorithm, the donor zone is a quadrilateral, and the target quadrilateral is split into two triangles. There are 20 permutations of tests per target triangle (each of the four donor segments is tested against the three target edges of $\mathbf{U}_2$ and the two rays pointing in the $+y$ direction at $x = 0$ and $x = 1$), or 40 total. In addition, there are eight subzones in $3d$ but only four in $2d$, typically causing a factor of 2 increase in the number of nonzero $V(z_{d,s}; z_{t,s})$ in $3d$ relative to $2d$ (for example, consider a mesh remapped onto itself). The number of possible nondegenerate intersections is therefore about 70 times greater per full donor zone in $3d$ than per full zone in $2d$.

3.3. *Filtering*

In the filtering process, pairs of donor and target zones are preselected before proceeding with the intersection calculation. Our goal is to perform the full geometry calculation only for zone pairs with nontrivial intersections. One may contrive arbitrarily large donor and target meshes, where all donor zones meet all target zones, thus rendering filtering ineffective. Therefore, filtering is not guaranteed to reduce the computer resource requirement below $O(N_d N_t)$, where N_d and N_t are the total numbers of donor and target zones, but if donor and target zones are sufficiently localized, filtering can bring considerable benefits, reducing the time requirement to $O(N_d \log N_d)$ in most practical remap problems.

We find the Cartesian bounding box surrounding the initial donor domain $\mathcal{D}_0$ (a domain is the entire set, or a subset, of donor zones) by identifying the minimum and maximum coordinates in each direction,

$$\begin{aligned} x_{\min}(\mathcal{D}_0) &\le x_0 \le x_{\max}(\mathcal{D}_0) \cap \\ y_{\min}(\mathcal{D}_0) &\le y_0 \le y_{\max}(\mathcal{D}_0) \cap \\ z_{\min}(\mathcal{D}_0) &\le z_0 \le z_{\max}(\mathcal{D}_0). \end{aligned} \tag{62}$$

We compute the bounding box limits of each target zone z_t and test for intersection between the bounding box for z_t and the donor domain bounding box, to obtain the initial list of target zones that match $\mathcal{D}_0$. We rank donor zones z_d in $\mathcal{D}_0$ according to the maximum x coordinate, $x_{\max}(z_d)$, to split $\mathcal{D}_0$ as evenly as possible (ties are broken arbitrarily) into two subdomains, $\mathcal{D}_1$ and $\mathcal{D}_2$. The Cartesian boxes (62), for $\mathcal{D}_1$ and $\mathcal{D}_2$ are enclosed within the $\mathcal{D}_0$ box, and therefore from the initial target zone list for $\mathcal{D}_0$ we select lists for $\mathcal{D}_1$ and $\mathcal{D}_2$. Recursive subdivision of the subdomains, rotating among the x, y, and z coordinates for demarcation, allows us to successively weed out target zones, until we obtain a donor subdomain with one donor zone (or subzone). We only need to simultaneously store one direct line of descendants of donor subdomains. Except for the lists of matching target zones, we only store collective information about the donor subdomains (by passing pointers to $\mathcal{D}_0$ arrays into subdomains), limiting memory usage by the filtering process to roughly several times N_t integers. This filtering method has been applied to both structured and unstructured hex meshes.

We apply a second stage of filtering after transforming the donor zone (or subzone) into the $\mathbf{U}$ frame for each target tet. If the entire donor zone falls into any of the half spaces

$$\begin{aligned} &x \le 0, y \le 0, z \le 0, h \le 0, \\ &x \ge 1, y \ge 1, z \ge 1, h \ge 1, \end{aligned} \tag{63}$$

then $v(z_d; z_t, k_t) = 0$ and the intersection calculation is not performed. We have found that it is advantageous to invest computer time to perform this test before proceeding with the full calculation of $v(z_d; z_t, k_t)$.

3.4. *Tests of Filtering*

We provide results for a simple test of the bounding box filtering algorithm using structured meshes. The target is a regular Cartesian grid with spacing a and 18 zones (36 subzones) in each direction. The donor zones are rectangular, with dimensions $(\ell a, a, a)$, rotated so the direction of elongation is $1/\sqrt{3}(1, 1, 1)$. In this case, the bounding box volume is proportional to ℓ^3 in the large ℓ limit, while the volume of the zone is proportional to ℓ, leading to inefficient filtering. The number of donor zones intersecting the target domain is roughly proportional to $1/\ell$. The filtering efficiency is defined as the ratio of number of nonzero volumes $v(z_{d,s}; z_{t,s}, k_t)$ with $\mathbf{U}$ to the total number of overlap volume calculations that are carried out (in practice we use a threshold $|v| > F$ to avoid counting volumes of "zero plus roundoff error"). The donor domain has 36 zones (72 subzones) in each dimension, centered at

$$(9a, 9a, 9a) + 1/(2\sqrt{3})(\ell a, a, a)$$

so that for $\ell \ge 1$ the donor domain completely encloses the target domain. For $\ell = 32$ the bounding boxes for some donor subzones encompass several thousand target subzones. We

TABLE VII
Performance: Elongated Meshes

Elongation, ℓ	CPU time (s)	Nonzero volumes	Efficiency
1	111.2	1731536	0.608
2	98.1	1382498	0.538
4	110.1	1214546	0.406
8	160.9	1122467	0.245
16	319.3	1077336	0.115
32	770.9	1059208	0.048

compare performance for $1 \leq \ell \leq 32$ in Table VII, where (63) has also been applied. As ℓ increases, the filtering becomes less efficient, permitting more calculations of v between nonintersecting zone pairs and reducing performance. However, a smaller ℓ (smaller donor zones) causes more donor zones to fit within the target volume, thus increasing the number of nontrivial intersections, so that $\ell = 1$ takes more time than $\ell = 2$, despite a higher efficiency. This test suggests that a more sophisticated filtering algorithm may increase performance when zones are highly elongated.

We compare various geometries commonly used in three-dimensional physics problems. The full implementation of the code contains the filtering stages described in Section 3.3 and Eq. (63) and several other filters buried within the geometry algorithm to weed out unnecessary intersection tests. Therefore, different logic and arithmetic operations are performed for different donor triangles *PQR* in the **U** frame, and by testing the algorithm with various meshes we can gain insight into the effect of mesh geometry on performance. For this test, we have defined three meshes, each with 5832 zones. The first mesh is a regular cubic 18^3 Cartesian grid with grid spacing a centered at $(9a, 9a, 9a)$. In the second mesh the nodes are arranged in a cylindrical geometry with a radius of $9a$ and 6 radial zones, 18 zones spanning a length of $18a$, and 54 angular zones, also centered at $(9a, 9a, 9a)$. This is not a true cylindrical mesh, because the zone boundaries are linear interpolations between the nodes rather than arcs. The third mesh is a structured 18^3 grid, with the nodes arranged as a tetrahedron whose base is the equilateral triangle

$$\begin{gathered}(4.8a,\ (9 + 4.2\sqrt{3})a,\ 0.6a),\\ (4.8a,\ (9 - 4.2\sqrt{3})a,\ 0.6a),\\ (17.4a,\ 9a,\ 0.6a)\end{gathered}$$

with apex $(9a, 9a, 17.4a)$. At the apex, all of the nodes in a logical plane meet, resembling the convergence of nodes at the origin of a spherical mesh. The meshes are illustrated in Fig. 5, and the results are shown in Table VIII. In the last three rows, the meshes labeled "+" have been displaced by $10^{-8}(a, a, a)$. In all of these tests we have verified global and local mass conservation, ensuring that the geometry algorithm has successfully computed the grid intersections, despite coincidences between the locations of donor and target nodes, edges, and facets. Although all three meshes have the same number of zones, the total CPU time t_{tot} for the remap varies by more than a factor of 7 for the range of mesh geometries shown here. However, the average CPU time, t_{ave} per geometric calculation of $v(z_{d,s}; z_{t,s}, k_t)$ (ratio of t_{tot} to number of calculations) shows much less variation. Efficiency varies widely,

TABLE VIII
Remap Results, Three-Dimensional Geometries

Donor mesh	Target mesh	Full calculations of $v(z_{d,s}; z_{t,s}, k_t)$[a]	CPU time[b] (μs) per calculation of v	Total CPU time (s)
Cartesian	Cartesian	1.13×10^6	66.1	74.6
Cartesian	Cylinder	1.91×10^6	45.9	87.6
Cartesian	Tetrahedron	0.97×10^6	39.0	37.8
Cylinder	Cartesian	2.06×10^6	45.2	93.1
Cylinder	Cylinder	2.34×10^6	61.7	144.4
Cylinder	Tetrahedron	3.91×10^6	36.0	140.6
Tetrahedron	Cartesian	0.65×10^6	50.3	32.8
Tetrahedron	Cylinder	1.48×10^6	51.3	76.0
Tetrahedron	Tetrahedron	4.41×10^6	55.2	243.4
Cartesian	Cartesian+	1.57×10^6	45.2	70.9
Cylinder	Cylinder+	2.62×10^6	46.0	120.6
Tetrahedron	Tetrahedron+	3.35×10^6	51.0	170.5

[a] After application of (63).
[b] Measured on a 440 MHZ DEC Alpha.

from 0.06 for the tetrahedron self-remap to 0.79 for the Cartesian donor and tetrahedron target. When the donor and target subzones meet at a face, edge, or point, the decision made by (63) of whether to proceed with the geometric calculation of $v(z_{d,s}; z_{t,s}, k_t)$ may be influenced by roundoff effects during the affine transformation to the **U** frame, making

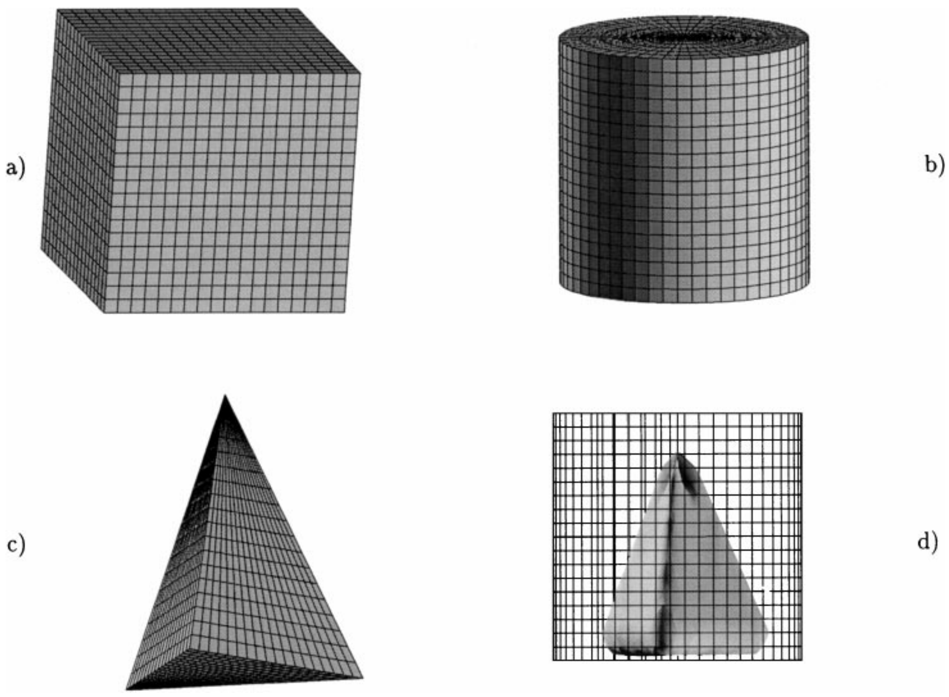

FIG. 5. Meshes for remap tests: (a) Cartesian; (b) cylindrical; (c) tetrahedron; (d) tetrahedron as donor mapped into cylindrical target (contour approximating reconstructed interface).

an accurate prediction of filtering performance difficult in such cases. When the target mesh is displaced slightly, relative to an identical donor mesh, donor zones acquire a small intersection volume with neighboring zones, requiring more computations of nonzero v. However, this is compensated by other donor–target pairs that are separated by displacement, allowing the first stage recursive filtering to discard such pairs and avoiding the expense of the affine transformation. The overall result is a reduction of t_{tot} in our test cases. Except for the self-remaps, the average times t_{ave} vary between 36.0 and 51.3 microseconds per calculation of v on a 440 MHz DEC Alpha, and the average time for the Rayleigh–Taylor remap, nearly 37 μs, falls within this range. These tests with different geometries show that total CPU time is highly dependent on the number of computations of v, emphasizing the importance of filtering.

3.5. *Rayleigh–Taylor Instability*

To demonstrate successful remapping of highly distorted meshes and to examine the effect of domain decomposition on performance, we have constructed a Lagrangian hydrodynamics simulation of a configuration with a Rayleigh–Taylor instability. The mesh is a $(25 \times 25 \times 50)$ hexahedral grid. The outer boundary of the problem is a rectangular box, elongated in the z direction, with horizontal lengths $L_h = 25a$ and vertical length $50a$. The grid is initially regular in the x and y directions, and the nodes are displaced from a regular grid in the z direction in a sinusoidal pattern. The initial coordinates for node (i, j, k) are

$$x = ai \tag{64}$$

$$y = aj \tag{65}$$

$$z = ak + (a/2)(1 - |k - 25|/25)\sin(2\pi x/L_h + \pi/4)\sin(2\pi y/L_h + \pi/4). \tag{66}$$

The top 25 zones in each column are filled with a heavy fluid with density ρ_h, and the bottom 25 zones contain a light fluid with density

$$\rho_l = 0.5\rho_h. \tag{67}$$

To simplify this test simulation, both fluids are treated as $\gamma = 5/3$ ideal gases. A uniform gravitational field in the $-z$ direction is balanced by an upward pressure gradient to remove the free-fall acceleration. The sinusoidal displacement profile serves as the initial seed for the growth of the Rayleigh–Taylor instability, and the zones themselves are clean (single material). Upon simulation using Lagrangian hydrodynamics, the displacement has grown as a function of time, and the mesh has become highly distorted (Figs. 6a and 8). In Fig. 8 we have illustrated a boomerang zone; the mesh also contains some self-intersecting zones. For our purposes, the distorted mesh has already been provided, and we utilize it to test the remapping capability. We remap the physical fields (both zone- and node-centered) from this distorted mesh onto another mesh which is closer to an orthogonal grid. We use a Cartesian target mesh of $(26 \times 26 \times 50)$ zones, each zone possessing horizontal dimensions $(25/26)a$ and vertical dimension a, so that the node coordinates are

$$(x, y, z) = ((25/26)ai, (25/26)aj, ak) \tag{68}$$

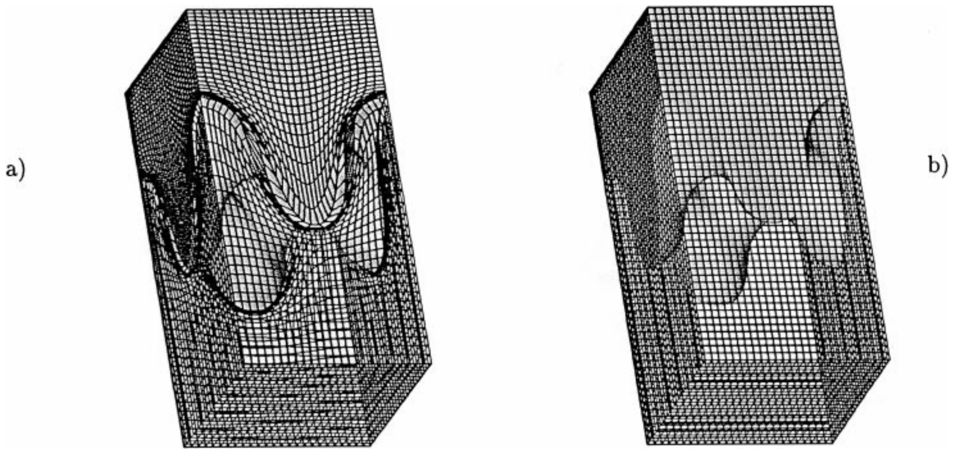

FIG. 6. Remapping example, a Rayleigh–Taylor unstable physical configuration. Gravity along the long axis of mesh causes acceleration of heavy fluid (shaded) into light fluid (transparent). Heavy fluid region is shaded (a) after Lagrangian simulation leads to highly distorted mesh, (b) after remap onto Cartesian mesh.

and a distorted mesh, where nodes with only odd indices are displaced,

$$\begin{aligned}(x, y, z) &= ((25/26)a(i + 0.1\eta), (25/26)a(j + 0.1\eta), a(k + 0.1\eta))\\ \eta &= (i \bmod 2) \cdot (j \bmod 2) \cdot (k \bmod 2).\end{aligned} \tag{69}$$

Except at the problem boundary, 50% of the hex faces become nonplanar under distortion.

The resulting interface between the light and heavy fluids on the mesh after the remap onto the Cartesian mesh is shown in Fig. 6. In Table IX we show the change $|\delta M|$ in the integrated mass density M, demonstrating that global mass conservation is accurate to within several bits of machine precision. We have found similar accuracy in the conservation of global momentum, based on the remap of the node-centered momentum density. As an additional test for each donor and target subzone we have tested the sum rules

$$\begin{aligned}V(z_{d,s}) &= \sum_{z_{t,s}} V(z_{d,s}; z_{t,s})\\ V(z_{t,s}) &= \sum_{z_{d,s}} V(z_{d,s}; z_{t,s})\end{aligned} \tag{70}$$

TABLE IX
Rayleigh–Taylor Problem Remaps

Target mesh	Target domains	CPU time[a] (s)	$\lvert\delta M\rvert/M$
Cartesian	1	596.8	3.6×10^{-14}
Cartesian	8	598.8	3.3×10^{-14}
Distorted	1	608.6	2.0×10^{-14}
Distorted	8	611.9	3.2×10^{-14}

[a] Measured on a 440 MHZ DEC Alpha.

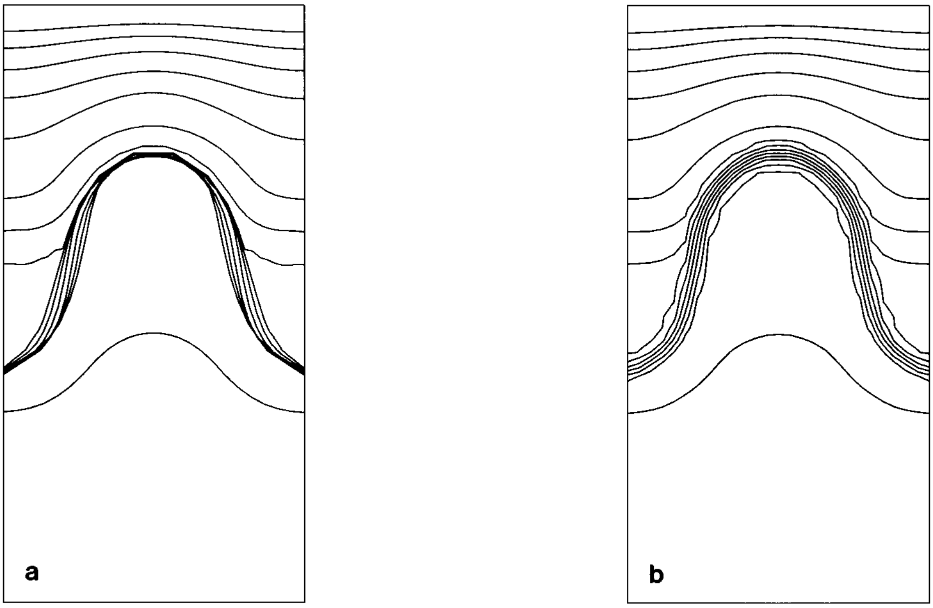

FIG. 7. Corresponding density contours of the central x-slice for the configuration shown in Fig. 6, (a) before and (b) after remapping.

to ensure that the summed intersection volumes for each donor and target subzone do not fractionally exceed (in magnitude) the total subzone volume by more than a threshold of 10^{-12}. The conservation of mass to nearly machine precision (Table IX) illustrates the ability of our grid intersection algorithm to handle meshes with distorted and self-intersecting zones. Using the second filtering stage (63) we obtain 64% efficiency in remapping this Rayleigh–Taylor problem for both target meshes. The density profile of a cross section is shown in Fig. 7. The spreading of the contour lines, especially near the fluid interface, illustrates diffusion that occurs as a consequence of the remapping process. Dukowicz and Kodis [5] have shown in a two-dimensional calculation that the diffusion can be mitigated with a second-order remap.

In Table IX we have also shown the effect of domain decomposition of the target mesh on performance. In a conventional remap on a serial machine, the entire target mesh is supplied as one domain and filtering selects target zones that match a donor zone, using the entire pool of target zones as a starting point. However, we may choose to treat the whole target as the union of multiple target meshes (domains), each with a portion of the original target. The donor domain is mapped onto each target domain separately, with the filtering repeated for each target domain. We have decomposed the full target into eight domains of dimension $(13 \times 13 \times 25)$ zones and found that the extra filtering leads to only a marginal increase in total CPU time (Table IX). This demonstrates the feasibility of a parallel implementation, in which each of the target domains would be kept on a separate processor, suggesting that with well-designed domain decomposition and communication methods, the remapping algorithm can potentially scale efficiently on parallel platforms. The reason for this is that the filtering algorithm can select and remove nonintersecting donor–target zone pairs in a small fraction of the time needed to accomplish an intersection calculation of $V(z_{d,s}; z_{t,s})$ for zones that do match.

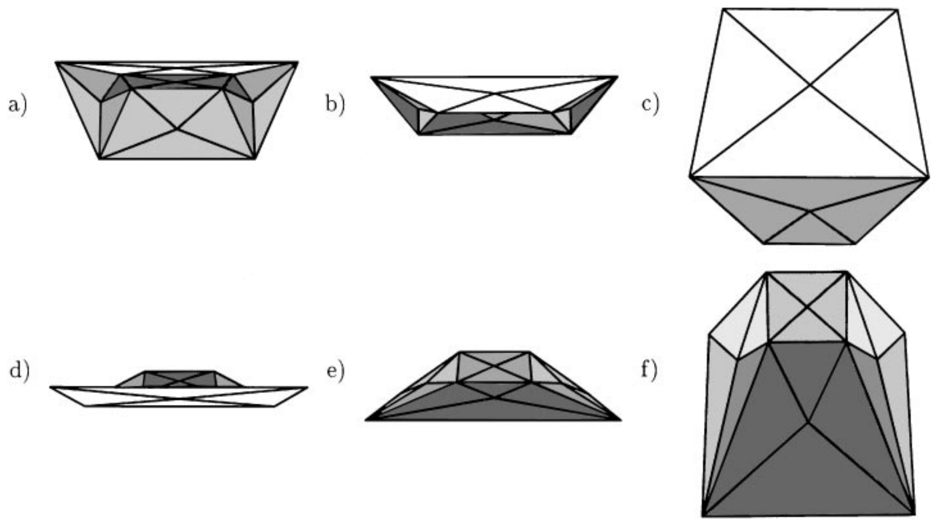

FIG. 8. A nonconvex zone resulting from running Lagrangian hydrodynamics to simulate the growth of a Rayleigh–Taylor instability. Six different views of the zone are shown, representing successive rotations about the horizontal axis.

4. APPLICATION: REGION OVERLAYS

Here we discuss the application of the geometric grid intersection algorithm to region overlays. In region overlays, a donor region within an arbitrary target mesh is filled with homogeneous material, and the boundary of this region does not conform to mesh zone boundaries, allowing some zones to be partially filled with the overlay material. Hence, the region boundary after the overlay is represented by volume fractions of the region's material in mixed zones that partially intersect the region. Typically the region is a shape, such as a sphere, specified using collective parameters, such as the radius and the location of the center. Region overlays may be applied when it is not convenient to design a mesh that conforms simultaneously to different regions containing different materials. The essential requirement of region overlays is to compute the volume of intersection $V(\mathcal{R}; z_t)$ between the region $\mathcal{R}$ and each zone z_t of the mesh. General shapes with curved boundaries would require nonlinear intersection calculations to obtain the exact value of $V(\mathcal{R}; z_t)$, but since the complexity and expense of such calculations increases rapidly with the order of the shape boundary and a fast calculation of intersection volumes is necessary to make the overlay procedure practical for meshes with hundreds of thousands of zones per processor, we construct a set of triangular polyhedra $\mathcal{P}_i$ to approximate the region and implement the polyhedron intersection algorithm described in Section 2 to estimate $V(\mathcal{R}; z_t)$ by

$$V(\mathcal{R}; z_t) \approx \sum_i V(\mathcal{P}_i; z_t), \tag{71}$$

where $V(\mathcal{P}_i; z_t)$ is the intersection volume of the ith polyhedron with the zone z_t.

4.1. *Triangulation*

A finite approximation of $\mathcal{R}$ using polyhedral elements is associated with an approximation of its surface boundary with triangular patches. The elements form a decomposition of

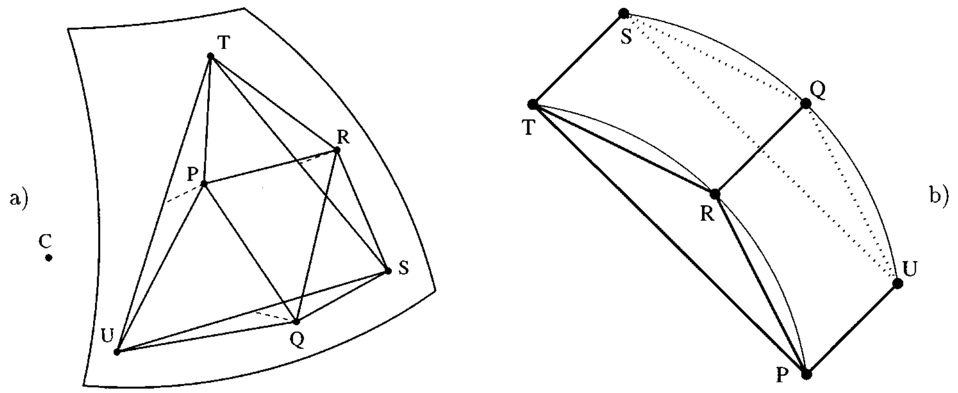

FIG. 9. Polyhedra derived from bisecting patches that approximate the surface of (a) sphere centered at C; (b) cylinder. The triangular prism (b) is converted to an octahedron by drawing segments *PQ*, *RS*, and *TU*.

the figure bounded by the patches. Two cases of special interest, which we describe here, are spheres and cylinders. Polyhedral approximation methods for spheres are well known and can be generalized for any star-shaped object for which the radius r is a continuous and first-differentiable function of the angular coordinates $\boldsymbol{\Omega}$. The polyhedra are constructed by inscribing a regular octahedron within a sphere and by using the midpoint projection method [16] to recursively bisect the triangular patches, producing a hierarchy of approximations to the sphere. For this bisection (Fig. 9a), from a triangular patch *STU* with vertices on the surface of the sphere, one constructs points P, Q, and R by projecting the line segments *TU*, *US*, and *ST* from the center of the sphere onto its surface. These six points define an octahedron, four of whose facets are new surface patches, and each of these new patches approaches $1/4$ the area and $1/2$ the linear dimensions of the original patch *STU* as the solid angle of *STU* approaches zero. Recursion level m is a polyhedron that contains N_p patches and N_d donor octahedra, with

$$N_p = 2 \times 4^m$$
$$N_d = (2 \times 4^m - 5)/3. \tag{72}$$

The total volume of the polyhedra v_p from this bisection method, compared with the sphere volume v_{sph}, in the limit as m approaches infinity, is

$$(v_p - v_{\text{sph}})/v_{\text{sph}} \sim -12.07/N_p. \tag{73}$$

The volume of intersection between each octahedron and each zone is computed to obtain the approximate shape–zone intersection volume through (71). This bisection method suffers from the well-known problem that at any given level of bisection the patches are smallest near the vertices of the initial octahedron, with less resolution elsewhere, but by increasing m to improve the overall accuracy of the calculation we can reduce the effects of this problem.

The recursive bisection is handled differently for a cylinder. The cylinder of length l_{cyl} is cut into n_b slices of thickness

$$l_s = l_{\text{cyl}}/n_b. \tag{74}$$

For a given slice an equilateral triangular prism is inscribed by drawing triangles on the disks at each end of the slice and connecting corresponding vertices along the length of the slice. The resulting figure is a triangular prism which is cast as an octahedron by triangulating the rectangular facets. In the bisection (Fig. 9b), points Q and R are the midpoints of segments PT and US, respectively, projected from the axis onto the surface of the cylinder. We hold l_s constant and use an inherently one-dimensional method to approximate the circular disk with the polygon. The total number of rectangular patches and octahedra for m levels of recursion is

$$\begin{aligned} N_p &= n_b(3 \times 2^m)/2 \\ N_d &= n_b(3 \times 2^m - 4)/2. \end{aligned} \tag{75}$$

A one-dimensional bisection method is effective because the cylinder is a ruled surface (its curvature is zero in one direction), and the error decreases more rapidly as a function of N_p for a cylinder than for a sphere,

$$(v_p - v_{\text{cyl}})/v_{\text{cyl}} \sim -6.58 n_b^2 \big/ N_p^2, \tag{76}$$

and, therefore, the number of elements (polyhedra) in $\mathcal{R}$ as a function of volume accuracy increases more slowly for a cylinder than for a sphere. For both the sphere and the cylinder, the systematic volume error asymptotically scales as 4^{-m} for large m. In our examples of overlays of spheres and cylinders, we use the same geometry algorithm described in Section 2 for computing polyhedron intersections to compute $V(\mathcal{P}_i; z_t)$. The target meshes are still composed of tetrakis hexahedra, but the donor zones, instead of 12 or 24 faceted polyhedra derived from hexahedra, now form an unstructured hierarchical list of octahedra.

It is also possible to map a sphere or cylinder onto an arbitrary mesh by creating a donor hexahedral mesh with nodes arranged in the geometry of a sphere or cylinder and applying a general remap. However, this conventional donor mesh gives a less accurate approximation of the sphere or cylinder than the specialized decompositions described above. If we consider a structured mesh, designed with an overall spherical geometry, with n_r radial zones, πn_r zones in the polar direction, and $2\pi n_r$ azimuthal zones, for a total number of donor zones

$$N_d = 2\pi^2 n_r^3, \tag{77}$$

the zones on the surface at the equator approximate cubes with side length r/n_r in the limit of large n_r and the solid angle of one such equatorial zone is

$$\begin{aligned} \Omega &= 1 \big/ n_r^2 \text{sr} \\ &= \left(N_d/2\pi^2\right)^{-2/3}. \end{aligned} \tag{78}$$

For the sphere approximation, the average solid angle per patch is

$$\begin{aligned} \Omega &= 4\pi/N_p \\ &= 4\pi(3N_d + 5)^{-1}. \end{aligned} \tag{79}$$

This surface patch bisection method therefore improves the accuracy of the region overlay by concentrating the smallest donor elements (octahedra) near the surface and also performs well because the octahedra become smaller with each successive bisection, so that their bounding boxes match fewer target zones. If we were to design a donor hex mesh with one radial zone (so that all zones are pyramid-shaped with four nodes coincident at the center of the sphere) the solid angle would scale as N_p^{-1} as in (79), but the long, pencil-shaped zones would create large bounding boxes and, therefore, require a more elaborate filtering scheme than we have implemented here (see Section 3.3). Development of specialized polyhedral approximations of a region $\mathcal{R}$ requires knowledge of the analytic form of $\mathcal{R}$, and for our first-order remap, homogeneous fields throughout its volume.

4.2. *Filtering for Region Overlays*

As in Section 3.3, we seek to select only zones and octahedra for which the result is nontrivial before proceeding with the geometry calculation. For example, a recursion level of $m = 9$ on a sphere of 20 zones radius produces accuracy of about 10^{-4} in overlay volume fraction (ratio of intersection volume to target zone volume), and contains 2^{19} surface triangles and 174,761 octahedra, which for a target mesh of 40^3 zones produces over 10^{11} potential combinations of source octahedron and target tetrahedron. We describe the various preselection filters in detail for a sphere shape, and filter methods for many other shapes are analogous. Unlike for zonal remapping, we cannot simply draw a bounding box around a constituent octahedron, analogous to (62) for the remap, because descendants of the octahedron may protrude outside that bounding box, thus causing target zones to be prematurely dropped from the list. Therefore an enlarged bounding box, which accounts for the curvature of the sphere, is constructed.

Because the material inside the sphere is homogeneous, we do not explicitly compute the volume of intersection for zones known to be completely enclosed within the sphere. The sphere of radius r_s is centered at (x_c, y_c, z_c) in physical coordinates and the distance

$$r_i^2 = (x_i - x_c)^2 + (y_i - y_c)^2 + (z_i - z_c)^2 \tag{80}$$

for each node is calculated. For a zone bounded by eight nodes, if the maximum nodal distance r_i for each node i satisfies

$$\max\left(r_i^2\right) \le r_s^2, \tag{81}$$

the zone is inside the sphere and assigned an intersection volume $V(\mathcal{R}; z_t)$ equal to the full target zone volume $V(z_t)$. Likewise, we identify exterior zones, with a zero overlap volume, by

$$\min\left(r_i^2\right) \ge r_s^2 + \ell^2/4, \tag{82}$$

where ℓ^2 is the diagonal length of the Cartesian rectangular solid surrounding the zone, added to account for the possibility of the sphere invading the zone between the nodes.

The initial list of zones assigned to the initial octahedron $\mathcal{P}_0$ is the set of zones satisfying neither (81) nor (82) and the overlay calculation is enacted between $\mathcal{P}_0$ and these zones. Using the bisection procedure from Section 4.1, we define $\mathcal{P}_1$ to be one of the $m = 2$

octahedra derived from the bisection of a $\mathcal{P}_0$ facet. If the x-coordinate of $\mathcal{P}_1$ is bounded by

$$x_0 \leq x \leq x_1, \tag{83}$$

the limits for the enlarged bounding box are

$$\begin{aligned} \rho^2 &= r_s^2 - \ell_{PQR}^2/4 \\ \tilde{x}_0 &= \begin{cases} x_0 - (x_c - x_0) r_s/\rho & (x_0 < x_c), \\ x_0 & (x_0 \geq x_c); \end{cases} \\ \tilde{x}_1 &= \begin{cases} x_1 + (x_c - x_1) r_s/\rho & (x_1 > x_c), \\ x_1 & (x_1 \leq x_c), \end{cases} \end{aligned} \tag{84}$$

where ℓ_{PQR} is the diagonal length of the Cartesian box surrounding PQR in Fig. 9a. Zones from the $\mathcal{P}_0$ list with bounding boxes that intersect the enlarged $\mathcal{P}_1$ box are selected for intersection calculations with $\mathcal{P}_1$. By using these enlarged boxes, we ensure that the box for each descendant octahedron is contained within the parent's box, and therefore, the zone selection by recursive elimination is consistent. For a cylinder, we also enlarge the donor bounding boxes to account for curvature.

4.3. *Examples of Region Overlays*

We show examples of overlays of spheres and cylinders, onto regular Cartesian and distorted target meshes. The Cartesian mesh contains 50 cubic zones of length a in each of the three directions for a total of 125,000 zones, and the nodes are logically numbered from 0 to 50 in each direction. The center of the sphere is located at the physical center of the cube enclosing the mesh, and we overlay spheres with radii of $1a$, $10a$, and $25a$. For the distorted mesh, nodes whose logical coordinates are all odd are displaced by

$$0.02(a, a, a)$$

(no nodes on the boundary are displaced). In the region overlays, only zone-centered fields are applied to the target. A tetrahedral decomposition for the 24-faceted TH target zone is used, with an average of about 12 tets per zone if all zone boundaries are nonplanar, or six tets per zone if all boundaries are planar. In our distorted mesh, 50% of the zone boundaries for interior zones (away from the domain boundary) are nonplanar. For the cylinder, the length is $50a$ along one Cartesian direction, and we use radii of $1a$, $10a$, and $25a$. We have used for the cylinder

$$\begin{aligned} l_s &= \text{floor}(l_{\text{cyl}}/a_t) \\ a_t &= (V_t/N_t)^{1/3}, \end{aligned} \tag{85}$$

where V_t is the total volume of the target mesh, N_t is the number of zones, and a_t is an estimate for the average length of a zone, which yields $n_b = 50$ slices for our test cylinders. In Fig. 10, we show the CPU times for spheres and cylinders on a single 440 MHz DEC Alpha for various radii and levels of bisection, and we also compare the Cartesian mesh with the distorted mesh which contains surface tets on nonplanar faces. The filtering saturates at

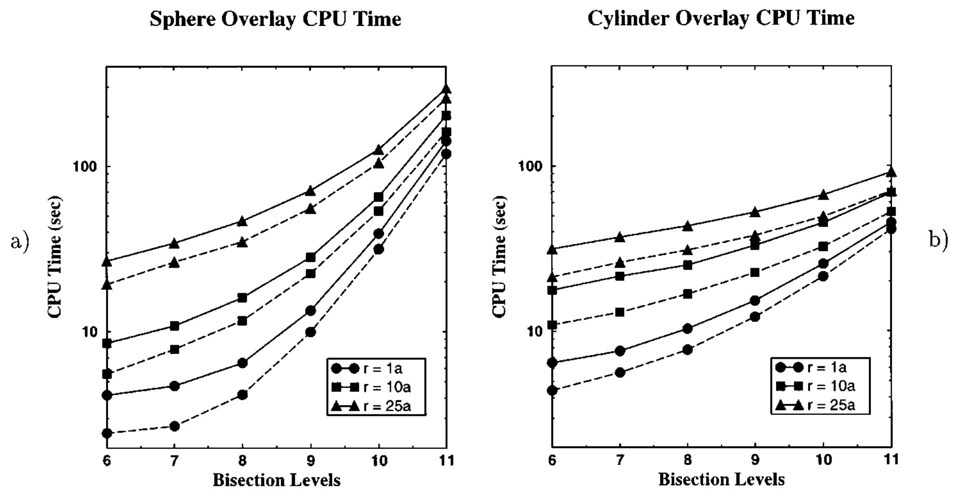

FIG. 10. CPU time required to overlay (a) spheres and (b) cylinders of various radii onto Cartesian (dashed) and distorted (solid) meshes, as a function of number of bisection levels, m.

large m when the donor elements (octahedra) become point-like for a sphere, or filament-like for a cylinder. For example, a donor octahedron and all of its progeny may reside within a single target zone, and if no other target zones' bounding boxes touch the octahedron all progeny will contain only that one target zone as their lists of matching zones. The total number of elements N_d at layer m increases by a factor of 4 per layer for a sphere, but only 2 per layer for a cylinder, so we expect that for large m when the element volume becomes much smaller than a zone volume (and filtering saturates), the computer time spent on geometry will asymptotically scale as $N_d \sim 4^m$ for the sphere and as $N_d \sim 2^m$ for the cylinder (linearly with N_d). At the larger values of m the sphere time increases more rapidly as a function of m than does the cylinder time. Even in going from $m = 10$ to $m = 11$, however, the increases of both the sphere and the cylinder CPU times are still considerably less than linear in N_d, suggesting that the filtering has not quite reached saturation.

5. GRID INTERSECTIONS IN HYDRODYNAMICS

We perform tests of the remap and region overlay calculations as they relate to hydrodynamics simulations of two test problems, the spherically symmetric expansion of an ideal gas [15] with specific heat ratio $\gamma = 5/3$ and the Sedov spherical shock wave [15]. We consider a situation where it is necessary to represent the physical system on a Cartesian mesh after the simulation and measure the total energy lost to momentum diffusion. For example, a postprocessing code may require a Cartesian grid. The hydrodynamics code uses a second-order explicit Lagrangian predictor–corrector acceleration algorithm, and after every Lagrange cycle the mesh nodes are shifted in order to reduce distortion and an incremental remap, based on a stripwise implementation of a second-order advection procedure [12], is used to transfer material to neighboring zones on the adjusted mesh. The amount of transfer between neighboring zones is computed directly from the displacement of nodes on the boundary between the two zones during mesh relaxation. During the advection a gradient limiter is applied [12] to prevent the formation of artificial critical points, and thus, the advection is not strictly second-order accurate everywhere. These second-order

Lagrange and advection procedures have recently been applied in $3D$ laser physics simulations [11]. We consider two schemes for arriving at a Cartesian mesh at simulation time $t = T$, beginning with a Cartesian mesh at $t = 0$:

(1) Relax the nodes to the original Cartesian positions after every cycle, and remap the fields.
(2) Relax the nodes using a local equipotential method [9] and remap after every cycle an ALE method, allowing the mesh to deform over time. Remap directly to a Cartesian mesh at $t = T$.

For both tests, we begin with a 40^3 Cartesian mesh with total side length 6 cm and the coordinate origin at a corner. The background gas in the expansion (rarefaction) wave has density ρ and specific internal energy ϵ of

$$\begin{aligned} \rho &= 10^{-9}\ \text{g/cm}^3 \\ \epsilon &= 0. \end{aligned} \tag{86}$$

One octant of a sphere of radius $R = 1.5$ cm, centered at the corner (0, 0, 0), is placed onto the Cartesian mesh using the method of Section 4. This sphere contains the same gas (no material boundary is tracked) with

$$\begin{aligned} \rho &= 1\ \text{g/cm}^3 \\ \epsilon &= 5 \times 10^{10}\ \text{erg/g} \end{aligned} \tag{87}$$

(pressure is $10/3 \times 10^{10}$ dyn/cm^2 inside the sphere). In the Sedov shock, we initialize with cold background material with

$$\begin{aligned} \rho &= 10^{-3}\ \text{g/cm}^3 \\ \epsilon &= 0, \end{aligned} \tag{88}$$

except in the single zone at the origin, where we set the specific internal energy to 5×10^{10} erg/g (pressure is $10/3 \times 10^7$dyn/cm^2). Reflective boundary conditions are enforced on the three planes of the mesh that meet at the origin. On the opposite three planes, in order to maintain the integrity of the mesh, we fix the node locations, allowing material to flow outward through the boundary. We determine an appropriate final time T by demanding that less than 10^{-7} of the total mass has been lost through the outer boundary, ensuring that the loss of total energy due to outflowing material is insignificant, and choose

$$\begin{aligned} T &= 6.0\ \mu\text{s} \quad &&\text{(rarefaction)} \\ T &= 1250\ \mu\text{s} \quad &&\text{(Sedov shock).} \end{aligned} \tag{89}$$

Since the gas before the wave (or shock front) is not traveling significantly faster than the wave itself, and the gas at the outer boundary begins at zero temperature, the fraction of lost energy due to mass loss is comparable to or less than the fraction of lost mass. We measure the total problem energy before and after the incremental remap at every cycle and, thus, isolate the total energy deficit due to momentum diffusion during the incremental remap from numerical drift of the energy during the Lagrange cycle. At $t = T$, we remap the fields

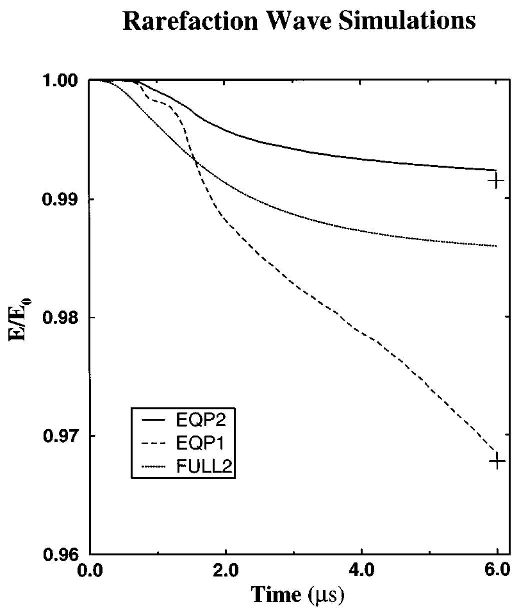

FIG. 11. Energy normalized to initial energy for rarefaction wave versus simulation time. The vertical crosses are the total energy of the EQP2 and EQP1 systems after direct remap to Cartesian mesh.

from the distorted ALE mesh to the original Cartesian mesh and calculate the kinetic energy before and after remapping. In Fig. 11 we show the total energy as a function of time for the simulation of the rarefaction wave, and the energy for the Sedov shock is shown in Fig. 12. In all of these runs, over 98% of the energy lost during the simulation is attributable

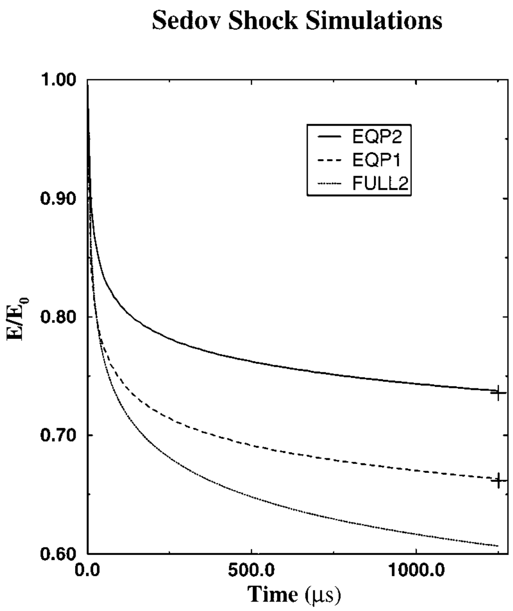

FIG. 12. Same as Fig. 11, for Sedov shock.

to the momentum diffusion during the incremental remaps, with the remainder of energy loss during the Lagrangian evolution.

In these figures, we have used three grid motion schemes, local equipotential relaxation of the nodes after every Lagrange step with second-order advection (EQP2), equipotential relaxation with first-order advection (EQP1), and full relaxation to the original positions after each Lagrange step (FULL2). The FULL2 method returns the problem to a Cartesian mesh after every step and uses second-order stripwise advection. At the final time T, the EQP2 and EQP1 results are directly remapped to the Cartesian mesh, as described in Sections 2 and 3. The energies of these configurations after the direct remap are shown as plus signs in Figs. 11 and 12. The plots show that the energy lost in an application of the first-order direct remap, performed after the simulation, is less than the energy lost to momentum diffusion during the advection. This interpolation technique (3), if needed for frequent use within the hydro, would need to be improved to second-order in order to prevent excessive diffusion. The direct remap CPU items for the EQP2 runs are 1210 s on a 440 MHz DEC Alpha for the Sedov wave, and 1010 s for the rarefaction wave.

6. SUMMARY

We have developed a fully geometric algorithm for grid intersections between polyhedral meshes and demonstrated applications to remapping of physical configurations from one mesh to a different mesh and to region overlays. This geometry algorithm represents an improvement over the method of Dukowicz and Padial [4] because we are able to detect all possible intersections between zones and because we handle coincidences between donor and target nodes, edges, and facets without altering the positions of nodes, thus allowing us to conserve mass to nearly machine precision. We have demonstrated the use of region overlays to initialize and remapping to postprocess meshes for hydrodynamics simulations. For our remap tests, the speed ranges from 53 full target zones per second for the Sedov shock and 57 zones per second for the Rayleigh–Taylor test (Cartesian target) to 64 target zones per second for the rarefaction wave. A preliminary test using domain decomposition suggests that a parallel implementation will be relatively straightforward and potentially scale efficiently with the number of processors.

There are several ways in which we plan to provide enhancements to the remapping procedure. We will handle mixed zones in the donor mesh by decomposing these zones into polyhedral parts, each of which contains a single material, before computing the volumes of intersection between each part and the target zones. We also plan to implement more optimal filtering algorithms, tailoring these algorithms to particular mesh geometries. A second-order remap procedure, with the density field $q(\mathbf{x})$ within each zone a linear function of $\mathbf{x}$, is planned in order to reduce the effects, such as loss of kinetic energy, arising from diffusion, thus increasing the accuracy of the remap.

ACKNOWLEDGMENTS

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract W-7405-Eng-48. Figures 5, 6, and 7 were drawn using MeshTV, a scientific visualization program developed at Lawrence Livermore National Laboratory. Figure 8 was drawn using Geomview, a program developed at the Geometry Center at the University of Minnesota.

REFERENCES

1. E. G. Puckett, A high-order projection for tracking fluid interfaces in variable density incompressible flows, *J. Comput. Phys.* **130**, 269 (1997).
2. H. G. Horak *et al.*, An algorithm for the discrete rezoning of Lagrangian meshes, *J. Comput. Phys.* **26**, 277 (1978).
3. M. Schneier, Calculation of geometric properties using quadtrees, *Comput Graphics Image Process.* **16**, 296 (1981).
4. J. K. Dukowicz and N. T. Padial, *REMAP3D: A Conservative Three-Dimensional Remapping Code*, LA-12136-MS, Los Alamos National Laboratory, 1991.
5. J. K. Dukowicz, and J. W. Kodis, Accurate conservative remapping (rezoning) for arbitrary Lagrange–Eulerian computations, *SIAM J. Sci. Stat. Comput.* **8** (1987), pp. 305–321.
6. B. Chazelle, An optimal algorithm for intersecting three-dimensional convex polyhedra, *SIAM J. Comput.* **21** (1992), pp. 671–696.
7. N. Arnenta *et al.* (The Computational Geometry Impact Task Force), *Application Challenges to Computational Geometry*, Technical Report TR-521-96, Princeton University, April 1996.
8. J. K. Dukowicz, Conservative rezoning (remapping) for general quadrilateral meshes, *J. Comput. Phys.* **54**, 411 (1984).
9. A. M. Winslow, Numerical solution of the quasipotential poisson equation in a nonuniform triangle mesh, *J. Comput. Phys.* **1**, 149 (1966).
10. K. B. Wallick, *REZONE: A Method for Automatic Rezoning in Two-Dimensional Lagrangian Hydrodynamics Problems*, Los Alamos National Laboratory, LA-10829-MS (1987).
11. M. M. Marinak *et al.*, Three-dimensional simulations of nova high growth factor capsule implosion experiments, *Comput. Phys. Plasmas* **3**, 2070 (1996).
12. B. Van Leer, Toward the ultimate differencing scheme. V. A second-order sequel to godunov's method, *J. Comput. Phys.* **32**, 101 (1979).
13. J. D. Ramshaw, Conservative rezoning algorithm for generalized two-dimensional meshes, *J. Comput. Phys.* **59**, 193 (1985).
14. D. S. Miller, D. E. Burton, and J. S. Oliviera, *Efficient Second Order Remapping on Arbitrary Two Dimensional Meshes*, UCRL-ID-123530, Lawrence Livermore National Laboratory (LLNL), Livermore, CA, 1996.
15. Y. B. Zeldovich and Y. P. Raizer, *Physics of Shock Waves and High-Temperature Hydrodynamic Phenomena* (Academic Press, New York, 1966).
16. J. Leech, Sphere triangulation code, ftp://ftp-graphics.stanford.edu/pub/-Graphics/sphere.c (1989).