

A smoothed two- and three-dimensional interface reconstruction method

Stewart Mosso · Christopher Garasi · Richard Drake

Received: 31 January 2007 / Revised: 26 June 2007 / Published online: 22 April 2008
© Springer-Verlag 2008

Abstract The Patterned Interface Reconstruction algorithm reduces the discontinuity between material interfaces in neighboring computational elements. This smoothing improves the accuracy of the reconstruction for smooth bodies. The method can be used in two- and three-dimensional Cartesian and unstructured meshes. Planar interfaces will be returned for planar volume fraction distributions. The algorithm is second-order accurate for smooth volume fraction distributions.

1 Background

In Eulerian and Arbitrary Lagrange-Eulerian simulation codes, elements flux material to neighboring elements. For volume-of-fluid simulation codes, the volume fraction of

each material in each element is used to reconstruct the interfaces that separate materials. The reconstructed interface is not retained between computational time steps. The volume fraction is stored and evolved to reflect the material motion through the mesh. By accurately reconstructing the material interfaces, the material fluxes will more accurately reflect the true spatial evolution of the materials. Less accurate reconstruction methods produce a distortion of the materials.

The algorithms that are used to move multiple materials through a computational mesh are numerous and diverse. This paper will only recount the history of interface reconstruction that led to the development of this method.

An early interface reconstruction algorithm was developed by Noh [4] called the Single Line Interface Construction (SLIC) method. While it kept a well defined boundary between adjacent materials, it led to large distortions in the overall shape of the materials. For instance, circular bodies that were moved through the mesh in pure translation didn't remain circular. The SLIC method represented the material interfaces as either vertical or horizontal line segments. This lack of varying slope produced the inherent distortion.

The next advance in the current line of interface reconstruction was Youngs' algorithm [9]. The two-dimensional algorithm exactly reproduced a linear material interface; however, its extension to three spatial dimensions was not exact for all linear interface orientations. Youngs' 3D method used a local gradient of the volume fractions to approximate the normal to the piecewise linear representation. Each reconstructed interface was positioned in the mixed element (an element containing more than a single material) such that it enclosed the volume of the material. All the methods discussed in this paper construct piecewise linear interfaces that enclose the material volume. The difference between each algorithm is in the method and accuracy of the interface normal approximation.

Communicated by K. Mikula.

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy's National Security Administration under Contract DE-AC04-94AL85000.

S. Mosso (✉) · R. Drake
Sandia National Laboratories,
Computational Shock and Multiphysics,
1431, PO Box 5800 MS-0378, Albuquerque,
NM 87185-0378, USA
e-mail: sjmosso@sandia.gov

R. Drake
e-mail: rrdrake@sandia.gov

C. Garasi
Sandia National Laboratories, HEDP Theory,
1641, PO Box 5800 MS-1186, Albuquerque,
NM 87185-1186, USA
e-mail: cjgaras@sandia.gov

While the order of accuracy for Youngs' method is higher than the SLIC method, it isn't second-order accurate for all interface orientations. Youngs' original method could only be used for orthogonal, tensor product grids. There have been a number of extensions of the method to non-orthogonal, unstructured grids.

A two-dimensional smoothing method was developed by Mosso et al. [2]. The method employed stability points to refine the results of a Youngs' like reconstruction to exactly reproduce a linear interface in orthogonal and non-orthogonal meshes. The current work reported here has extended this simple smoothing algorithm into a second-order method in two and three spatial dimensions on orthogonal and non-orthogonal meshes.

2 Algorithm overview

2.1 Initial interface orientation

The Patterned Interface Reconstruction (PIR) algorithm starts with the description of a mesh. The mesh is described by the spatial coordinates of nodes, the volume fractions of the material(s) in each element, and the arrangement of the nodes that describe the boundary of each element. This method assumes that the edges of elements are linear and the facets of elements in three dimensions are either planar or have been tessellated into planar triangles. To reduce iteration expense the mesh is reduced into a list of mixed elements and a list of all single material elements that share a node with a mixed element. The neighboring single material elements are used in the gradient computation.

The first step in computing the interface between a material and all other materials is to perform a Youngs' like reconstruction. The interface normal is approximated by the gradient of the volume fraction of the material in each mixed element containing the material. The "home" element is computed using a Dukowicz [1] surface integral formulation, where

$$\hat{\mathbf{n}} \simeq \frac{-\nabla f}{|\nabla f|} = \frac{-\oint f \hat{\mathbf{s}} dS}{\left| \oint f \hat{\mathbf{s}} dS \right|}. \quad (1)$$

Here $\hat{\mathbf{n}}$ is the unit normal of the material interface, f is the volume fraction of the material in each neighboring element, the unit vector $\hat{\mathbf{s}}$ is the outwardly directed control volume unit normal, and dS is the surface differential of the control volume. The first step in evaluating the surface integral is to produce a node averaged volume fraction based upon a volume weighted average of the volume fractions in the elements surrounding the node. An illustration of the nodal volume fraction averaging of the element volume fractions surrounding the nodes is given in Fig. 1. The element of interest is the red sided element. Its nodes are shown as green

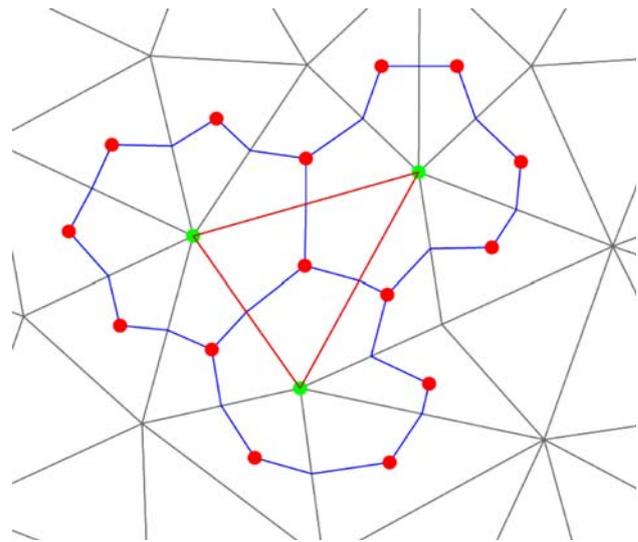


Fig. 1 Illustration of volume fraction gradient computation. The algorithm will calculate the gradient in the element with red edges. The volume fractions will be averaged for each green vertex using the blue control volumes. The red circles are the element centroids. The surface integral gradient will then be computed over the red edges of the element surface

circles. The control volume of each node is bounded by the blue segments. The element centroids are shown as red circles. Each nodal control volume edge extends from an edge midpoint to the element centroid. After the average nodal volume fractions are computed, the surface integral is then evaluated over the surface of each mixed element. The red segments are the edges that form the surface integral boundary. Using a linear average of the nodal values over each side and the outwardly directed edge unit normal, $\hat{\mathbf{s}}$, each edge integral is computed and summed. $\hat{\mathbf{n}}$ will be computed approximately correctly because the averaged volume fraction field approximately generates a differentiable level set representation.

Using the gradient normal as the initial interface normal, an algorithm similar to Yang and James [8] and Scardovelli and Zaleski [5] is used to efficiently iterate for the position of the interface within the element. This ensures that the interface bounds the correct material volume. An efficient positioning algorithm is critical to the efficiency of the PIR method. The PIR method invokes the positioning algorithm about twenty times per material per mixed element.

It was claimed in the first section of this paper that the gradient normal was an approximation to the interface normal. The smoothing algorithm will use the initial gradient normal interfaces to increase the accuracy of the results.

2.2 Linear smoothing

Consider a linear interface passing through the mesh neighborhood in Fig. 2. The exact linear input interface is shown

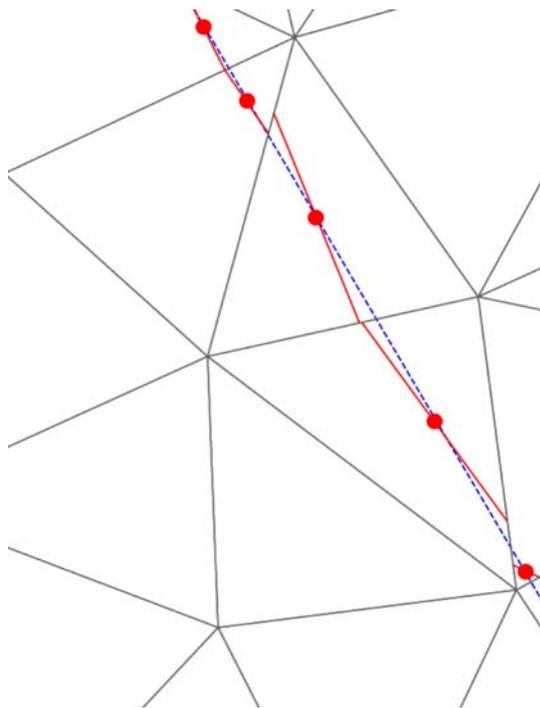


Fig. 2 Illustration of stability points for a linear interface

as a blue, dashed line and the reconstructed interfaces based upon the gradient normal are shown as red segments. The inaccuracy of the gradient normal is large enough to be visible and the interfaces are discontinuous at the element boundaries. In reference [2], the use of Swartz stability points was presented and discussed further in [6]. The stability points are the centroids of the reconstructed interfaces. They are shown as circles in Fig. 2. It can be observed that the end-points of the reconstructed interfaces are farthest from the exact interface while the stability points consistently lie near the exact interface. When the orientation of the reconstructed interfaces is varied slightly, the motion of the stability points is predominantly tangential to the interface. The motion of a stability point as the interface normal is varied is shown in Fig. 3. The interface normal is varied and the stability point is recomputed and plotted. The tangential motion makes the stability point a good approximation for the position of the interface if the normal were more accurately chosen. In reference [2] the linear smoothing algorithm used only two of the neighboring stability points to improve the interface normal. By using just two neighbors the algorithm contained some arbitrariness and the accuracy of the smoothing depended upon the correct selection of which neighbors to use. For a linear interface this choice was not critical; however, for curved interfaces the use of a reduced neighborhood could impact the accuracy of the algorithm if the chosen neighbors were more distant from the home element than other neighbors. In the current work the locations of the stability points

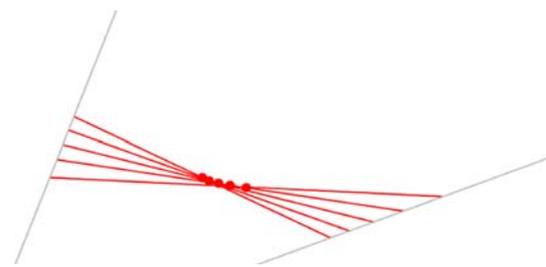


Fig. 3 Illustration of stability point motion as the interface normal is varied at constant volume fraction

in the “home” element and the neighboring elements are used in a linear, least squares fit of a plane (which is a line in two-dimensions) that can be used to improve the interface normal in the home element. The fitted interface is required to pass through the home element’s stability point and the sum of the squares of the orthogonal distance between each neighboring stability point and the fit is minimized. To simplify the algebra involved, we translate the origin of the coordinate system to the home element’s stability point via

$$\mathbf{S}'_i = \mathbf{S}_i - \mathbf{H},$$

where \mathbf{H} is the home element’s stability point position, \mathbf{S}_i is the physical position of a neighboring stability point, and \mathbf{S}'_i is the translated position. The points \mathbf{X} on a plane are described by the equation:

$$\hat{\mathbf{n}} \cdot \mathbf{X} - p = 0,$$

where $\hat{\mathbf{n}}$ is the unit normal to the line and p is the line’s orthogonal distance to the origin. Due to the translation of the coordinate system p will always be zero. The orthogonal distance, d_i , of each neighboring stability point from the smoothing line is:

$$d_i = \hat{\mathbf{n}} \cdot \mathbf{S}'_i.$$

The objective function to be minimized is

$$Obj = \sum_{i=1}^{ngbrs} W_i (\hat{\mathbf{n}} \cdot \mathbf{S}'_i)^2$$

where the summation is performed over the *ngbrs* neighboring, mixed elements. The smoothing algorithm seeks the unit normal, $\hat{\mathbf{n}}$ that produces the smallest objective function. Various weighting schemes were considered for W_i . The most promising weighting involved giving elements with a volume fraction of half a weight of one. Other volume fractions linearly decrease from the weighting of one at a volume fraction of half to a weighting of zero for volume fractions of zero and one. The weighting doesn’t significantly change the results of the smoothing procedure; thus, $W_i = 1$ is now used.

2.2.1 Linear smoothing in two-dimensions

The unit normal in two-dimensions can be expressed as the trigonometric functions of the angle ϕ :

$$\hat{\mathbf{n}} = \cos \phi \hat{\mathbf{i}} + \sin \phi \hat{\mathbf{j}}.$$

By using trigonometric functions the normal is guaranteed to be a unit normal. The signed, normal distance, d_i of a neighboring stability point $\{S'_{x_i}, S'_{y_i}\}$ from the interface description is:

$$S'_{x_i} \cos \phi + S'_{y_i} \sin \phi = d_i.$$

The optimization function for the fitting procedure becomes:

$$Obj = \sum_{i=1}^{ngbrs} (S'_{x_i} \cos \phi + S'_{y_i} \sin \phi)^2. \tag{2}$$

At the minimum and maximum objective function values $\frac{\partial Obj}{\partial \phi} = 0$. There will be four values of ϕ that satisfy the relation. Two of the values will be minima and two will be maxima. The two minima will be π radians apart because the two angles define the same orientation of the line in space and are the negation of each other. The two maxima will also lie π radians apart and will lie $\pi/2$ radians from the minima. To determine an expression for the values of ϕ at the extremes, we carry out the differentiation of the objective function with respect to ϕ and set the expression to zero:

$$\begin{aligned} \frac{\partial Obj}{\partial \phi} &= \frac{\partial}{\partial \phi} \left(\sum_{i=1}^{ngbrs} (S'_{x_i} \cos \phi + S'_{y_i} \sin \phi)^2 \right) \\ &= 2 \sum_{i=1}^{ngbrs} (S'_{x_i} \cos \phi + S'_{y_i} \sin \phi)(S'_{y_i} \cos \phi - S'_{x_i} \sin \phi) \\ &= 2(\cos^2 \phi - \sin^2 \phi) \sum_{i=1}^{ngbrs} S'_{x_i} S'_{y_i} \\ &\quad + 2(\cos \phi \sin \phi) \sum_{i=1}^{ngbrs} ((S'_{y_i})^2 - (S'_{x_i})^2). \end{aligned}$$

Solving the following equation for ϕ :

$$\begin{aligned} (\cos^2 \phi - \sin^2 \phi) \sum_{i=1}^{ngbrs} S'_{x_i} S'_{y_i} \\ + (\cos \phi \sin \phi) \sum_{i=1}^{ngbrs} ((S'_{y_i})^2 - (S'_{x_i})^2) = 0 \end{aligned}$$

gives us

$$\phi = \pm \arccos \left(\pm \sqrt{\frac{4S_{xy}^2 + D(D \pm \sqrt{t})}{2t}} \right),$$

where:

$$\begin{aligned} S_{xx} &= \sum_{i=1}^{ngbrs} (S'_{x_i})^2, \\ S_{yy} &= \sum_{i=1}^{ngbrs} (S'_{y_i})^2, \\ S_{xy} &= \sum_{i=1}^{ngbrs} S'_{x_i} S'_{y_i}, \\ D &= S_{xx} - S_{yy}, \\ t &= 4S_{xy}^2 + D^2. \end{aligned} \tag{3}$$

This allows us to compute a value for ϕ ; however, the result isn't as useful as might be thought. The output of the algorithm is a description of the line that best fits the input data. The answer will not be returned as an angle; rather, the answer will be returned as the values of the $\cos \phi$ and $\sin \phi$. Execution expense can be saved by eliminating the call to the arc cosine function in the computation of the angle. It is more efficient to determine the $\cos \phi$ and use the result to evaluate the $\sin \phi$:

$$\begin{aligned} \cos \phi &= \pm \sqrt{\frac{4S_{xy}^2 + (S_{xx} - S_{yy})(S_{xx} - S_{yy} \pm \sqrt{t})}{2t}}, \\ \sin \phi &= \pm \sqrt{1 - \cos^2 \phi}. \end{aligned}$$

There are four sets of values for $\cos \phi$ and $\sin \phi$. The expense in calculating extra values can be reduced. The ± 1 coefficient for the square root used in calculating $\cos \phi$ can be eliminated by recognizing that:

$$\cos \phi = -\cos(\phi + \pi).$$

This means that through the use of the ± 1 coefficient, we will be producing the mirror image of the same line. The second set of two roots, $\pm \sqrt{t}$, will produce the $\cos \phi$ for the minimum and maximum objective function. Both of these possible roots will be evaluated.

In the expression for the $\sin \phi$ in terms of $\cos \phi$, there is also a ± 1 coefficient. These two possible roots are also non-trivial and should both be evaluated. This gives us four sets of possible roots. All four sets should be evaluated and the objective function for each result should also be evaluated. Only by comparing the four values of the objective function can we determine which of the four unit normals is the correct solution.

In summary, given the set of stability points, we evaluate four possible unit normals:

$$\begin{aligned} \cos \phi_a &= \sqrt{\frac{4S_{xy}^2 + D(D + \sqrt{t})}{2t}}, \\ \cos \phi_b &= \sqrt{\frac{4S_{xy}^2 + D(D - \sqrt{t})}{2t}}, \\ \hat{\mathbf{n}}_1 &= \{\cos \phi_a, \sqrt{1 - \cos^2 \phi_a}\}, \\ \hat{\mathbf{n}}_2 &= \{\cos \phi_a, -\sqrt{1 - \cos^2 \phi_a}\}, \\ \hat{\mathbf{n}}_3 &= \{\cos \phi_b, \sqrt{1 - \cos^2 \phi_b}\}, \\ \hat{\mathbf{n}}_4 &= \{\cos \phi_b, -\sqrt{1 - \cos^2 \phi_b}\}. \end{aligned} \tag{4}$$

The magnitude of the computed objective function for each of the possibilities, is used to determine which normal produces the best fit:

$$\hat{\mathbf{n}}_{LS} = \{\hat{\mathbf{n}} \mid \min (Obj(\hat{\mathbf{n}}_1), Obj(\hat{\mathbf{n}}_2), Obj(\hat{\mathbf{n}}_3), Obj(\hat{\mathbf{n}}_4))\}.$$

In conclusion, given a value of the unit normal that produces the minimum objective function, we need to determine if the negation of the normal should be returned. The original gradient normal that is used to produce the modified Youngs' interface, $\hat{\mathbf{n}}_{grad}$ will be used to orient the least-squares normal, $\hat{\mathbf{n}}_{LS}$:

$$\text{if } (\hat{\mathbf{n}}_{grad} \cdot \hat{\mathbf{n}}_{LS} < 0) \hat{\mathbf{n}}_{LS} = -\hat{\mathbf{n}}_{LS}.$$

2.2.2 Linear smoothing in three-dimensions

In the previous section, a least-squares interface smoothing algorithm was developed. For three spatial dimensions, the development will proceed in a similar fashion. The stability points will be translated in space so that the home element's stability point lies at the origin.

We start by defining a plane in space that runs through the home element's stability point:

$$\hat{\mathbf{n}} \cdot \mathbf{X} = 0.$$

The normal $\hat{\mathbf{n}}$ can be expressed using the trigonometric functions of the angles, ϕ and θ :

$$\hat{\mathbf{n}} = (\sin \theta \cos \phi)\hat{\mathbf{i}} + (\sin \theta \sin \phi)\hat{\mathbf{j}} + (\cos \theta)\hat{\mathbf{k}}.$$

The angle ϕ is the angle that the projection of the plane's normal into the x-y plane forms with the x-axis. The angle θ is the angle between the plane's normal and the z-axis.

To perform the smoothing operation, there will need to be at least two neighboring stability points in addition to the home element's stability point. In practice, there are usually more than two.

The signed, normal distance, d_i of a point $\{x_i, y_i, z_i\}$ from the plane is:

$$x_i \sin \theta \cos \phi + y_i \sin \theta \sin \phi + z_i \cos \theta.$$

The sum of the squares of the stability points' distances from the fitted plane is used as the optimization function for

the least-squares algorithm:

$$Obj = \sum_{i=1}^{nbrs} (x_i \sin \theta \cos \phi + y_i \sin \theta \sin \phi + z_i \cos \theta)^2.$$

A sample plot of the objective function for a set of three points is shown in Fig. 4. Two peaks in the objective function occur when the plane is orthogonal to the angles that produce the minima. The minima occur at the values of the plot corresponding to directions along the plus and minus z-directions. It appears as if the value of the objective function is constant for all values of ϕ when θ is 0 and π . This is incorrect because the direction of the unit normal is independent of ϕ when the value of θ is 0 or π . This simply means that the normal is pointing along the plus and minus z-axis.

The same data may more easily be viewed by looking at the same function as a contour plot. This is shown in Fig. 5.

At the minimum and maximum objective function values $\frac{\partial Obj}{\partial \theta} = 0$ and $\frac{\partial Obj}{\partial \phi} = 0$. These expressions are, using similar symbols for the summation terms:

$$\begin{aligned} &S_{xy} \cos^2(\phi) \sin^2(\theta) - S_{xy} \sin^2(\phi) \sin^2(\theta) \\ &- S_{xx} \cos(\phi) \sin(\phi) \sin^2(\theta) + S_{yy} \cos(\phi) \sin(\phi) \sin^2(\theta) \\ &+ S_{yz} \cos(\phi) \cos(\theta) \sin(\theta) - S_{xz} \cos(\theta) \sin(\phi) \sin(\theta) = 0, \\ &S_{xx} \cos(\theta) \sin(\theta) \cos^2(\phi) + S_{xz} \cos^2(\theta) \cos(\phi) \\ &- S_{xz} \sin^2(\theta) \cos(\phi) + 2S_{xy} \cos(\theta) \sin(\phi) \sin(\theta) \cos(\phi) \\ &- S_{yz} \sin(\phi) \sin^2(\theta) + S_{yz} \cos^2(\theta) \sin(\phi) \\ &+ S_{yy} \cos(\theta) \sin^2(\phi) \sin(\theta) - S_{zz} \cos(\theta) \sin(\theta) = 0. \end{aligned}$$

We could employ a similar solution procedure as in the two-dimensional, linear fit, of solving for $\cos \phi$ and $\cos \theta$; however, the solution is found using a steepest descent followed by a Newton's descent iterative algorithm [3]. Analytic

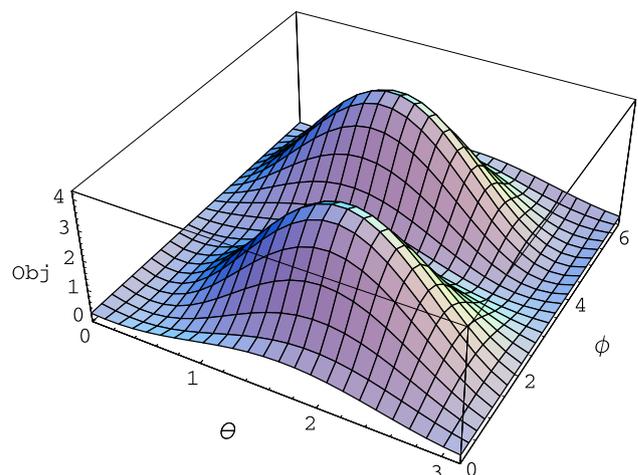


Fig. 4 Sample plot of objective function versus interface normal angles ϕ and θ

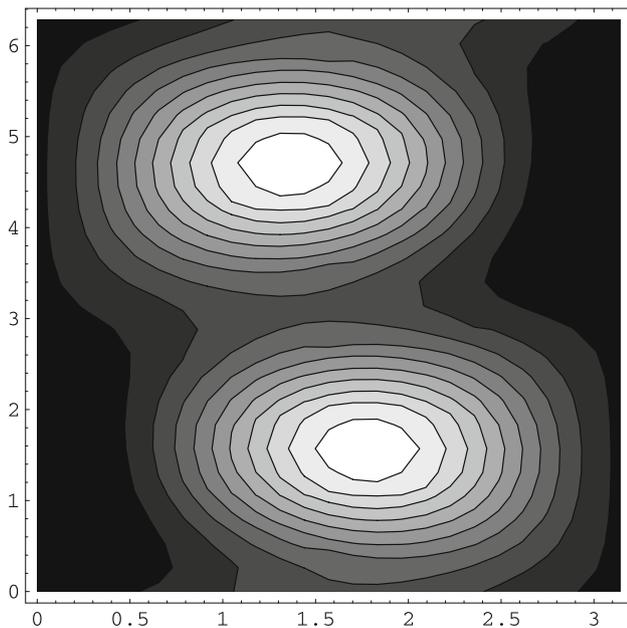


Fig. 5 Contour plot of objective function versus interface normal angles ϕ and θ

expressions for the gradient are used. The starting value of θ and ϕ are derived from the original gradient normal.

2.2.3 Linear smoothing performance

In practice, the linear smoothing produces second-order accurate results in both two- and three-dimensions. When the surface is curved, that is the stability points are significantly displaced from a planar distribution, the results are noticeably inaccurate. This inaccuracy can be caused by an unbalanced distribution of neighboring stability points. If a home element has one or two elements on one side with comparatively long interfaces, and the opposite side has a greater number of neighboring elements with smaller interface lengths, the method inclines the interface to the side with the greater number of neighboring elements. In this case, the use of previously mentioned weighting factors minimizes the inaccuracy but does not eliminate it. The smoothing method in the next section will usually produce a more accurate result.

3 Spherical smoothing

In the development of these algorithms, it was recognized that linear smoothing was not robust enough to produce second-order accurate interfaces for nonplanar interfaces. A spherical smoothing procedure was developed that fits a sphere to a neighborhood of mixed elements' stability points. This method produces a more accurate interface orientation for a wider set of interfaces.

3.1 Circular smoothing in two-dimensions

Several algorithms for smoothing were examined in the development of the current algorithm. It was found that algorithms that used more than the stability points, such as the neighboring interface normals, were slow to converge to an accurate result. This is due to oscillations in the neighboring normals. When the current method was developed, the smoothing rapidly converged to an accurate result.

Consider a home element and a set of neighboring elements as illustrated in Fig. 6. The stability points are placed close to the perimeter of the input circle. The circular smoothing algorithm seeks to compute the center of curvature for the neighborhood around the home element solely using the positions of the stability points. To compute the location of the local center of curvature for the home element, we construct a chord between the home element's stability point and each neighboring stability point. A plane perpendicular to each chord is constructed at the midpoint of each chord. This construction is shown in Fig. 7. A least squares fit is again constructed to these planes, and the point that minimizes the sum of the distances from each midchord plane is found.

For each of the neighboring stability points, a chord is formed that extends from the home stability point to the neighboring stability point. This direction will be used as the midchord plane's normal:

$$\hat{c}_i = \frac{\mathbf{S}_i - \mathbf{H}}{|\mathbf{S}_i - \mathbf{H}|} = \{c_{x_i}, c_{y_i}\}.$$

The position of the plane in space is given by:

$$p_i = \hat{c}_i \cdot \frac{1}{2} (\mathbf{S}_i + \mathbf{H}).$$

Let \mathbf{V} be the unknown position of the center of convergence. The objective function for the least squares fitting will be the sum of the squares of distances of \mathbf{V} from the mid-chord

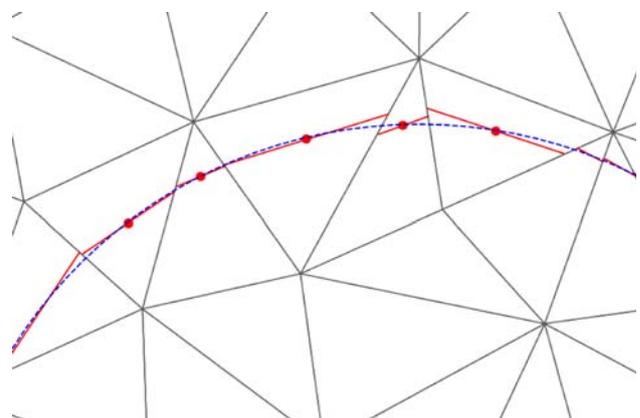


Fig. 6 Illustration of gradient normal based interfaces for a circular body

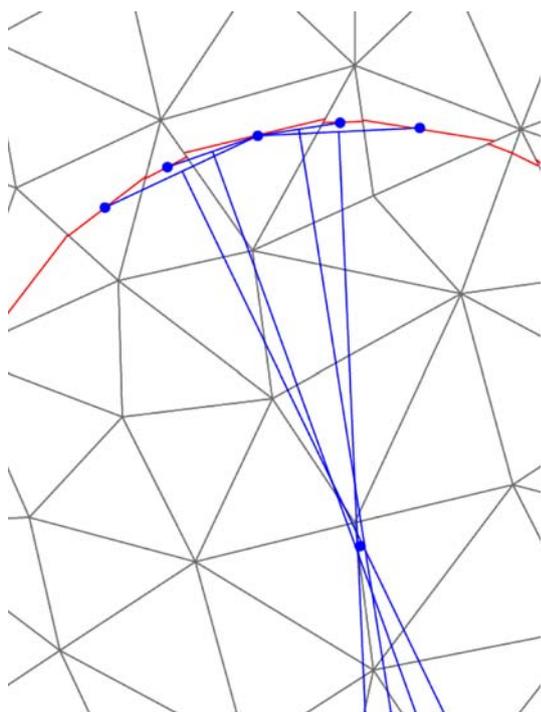


Fig. 7 Illustration of construction of center of curvature for circular smoothing

planes. The objective function can be written as:

$$\begin{aligned}
 Obj &= \sum_{i=1}^{ngbrs} (\hat{\mathbf{c}}_i \cdot \mathbf{V} - p_i)^2 \\
 &= V_x^2 \sum_{i=1}^{ngbrs} (c_{x_i})^2 + 2V_x V_y \sum_{i=1}^{ngbrs} (c_{x_i} c_{y_i}) \\
 &\quad - 2V_x \sum_{i=1}^{ngbrs} (p_i c_{x_i}) + V_y^2 \sum_{i=1}^{ngbrs} (c_{y_i})^2 \\
 &\quad - 2V_y \sum_{i=1}^{ngbrs} (p_i c_{y_i}) + \sum_{i=1}^{ngbrs} (p_i)^2.
 \end{aligned}$$

To find the values of V_x and V_y that minimize the objective function, we differentiate the objective function with respect to V_x and V_y :

$$\begin{aligned}
 \frac{\partial Obj}{\partial V_x} = 0 &= V_x \sum_{i=1}^{ngbrs} (c_{x_i})^2 + V_y \sum_{i=1}^{ngbrs} (c_{x_i} c_{y_i}) \\
 &\quad - \sum_{i=1}^{ngbrs} (p_i c_{x_i}),
 \end{aligned}$$

$$\begin{aligned}
 \frac{\partial Obj}{\partial V_y} = 0 &= V_x \sum_{i=1}^{ngbrs} (c_{x_i} c_{y_i}) - V_y \sum_{i=1}^{ngbrs} (c_{y_i})^2 \\
 &\quad - \sum_{i=1}^{ngbrs} (p_i c_{y_i}).
 \end{aligned}$$

Solving for V_x and V_y gives us:

$$\begin{aligned}
 V_x &= \frac{1}{D} (C_{xp} C_{yy} - C_{yp} C_{xy}), \\
 V_y &= \frac{1}{D} (C_{yp} C_{xx} - C_{xp} C_{xy}),
 \end{aligned} \tag{5}$$

where:

$$\begin{aligned}
 C_{xx} &= \sum_{i=1}^{ngbrs} (c_{x_i})^2, & C_{yy} &= \sum_{i=1}^{ngbrs} (c_{y_i})^2, \\
 C_{xy} &= \sum_{i=1}^{ngbrs} c_{x_i} c_{y_i}, & C_{xp} &= \sum_{i=1}^{ngbrs} (c_{x_i} p_i), \\
 C_{yp} &= \sum_{i=1}^{ngbrs} (c_{y_i} p_i), & D &= C_{xx} C_{yy} - C_{xy}^2.
 \end{aligned} \tag{6}$$

In Fig. 7 the results of this fit are shown at the convergence of the radial chord planes by a blue circle.

The new interface normal is constructed by passing a unit vector originating at the center of convergence through the home interface’s stability point:

$$\hat{\mathbf{n}} = \frac{\mathbf{H} - \mathbf{V}}{|\mathbf{H} - \mathbf{V}|}. \tag{7}$$

As in the previous linear fit, the orientation of the new normal is determined by examining its vector dot product with the volume fraction gradient normal. If the dot product results in a positive value, the orientation of the fit normal is retained. If the dot product results in a negative value, the orientation of the fit normal is reversed.

3.2 Spherical smoothing in three-dimensions

Derivation of the spherical smoothing expressions for three spatial dimensions proceeds in a similar manner as the two-dimensional derivation. The chords are constructed in the same manner and the objective function is:

$$\begin{aligned}
 Obj &= \sum_{i=1}^{ngbrs} (\hat{\mathbf{c}}_i \cdot \mathbf{V} - p_i)^2 \\
 &= \sum_{i=1}^{ngbrs} (p_i^2 - 2c_{x_i} V_x p_i + c_{x_i}^2 V_x^2 - 2c_{y_i} V_y p_i \\
 &\quad + 2c_{x_i} c_{y_i} V_x V_y + c_{y_i}^2 V_y^2 - 2c_{z_i} V_z p_i \\
 &\quad + 2c_{x_i} c_{z_i} V_x V_z + 2c_{y_i} c_{z_i} V_y V_z + c_{z_i}^2 V_z^2).
 \end{aligned}$$

The objective function is differentiated with respect to the three components of the position of the center of convergence. Each of these equations is set equal to zero and the coordinates of the center are solved for:

$$\begin{aligned} C_{xx}V_x + C_{xy}V_y + C_{xz}V_z - C_{xp} &= 0 \\ C_{xy}V_x + C_{yy}V_y + C_{yz}V_z - C_{yp} &= 0 \\ C_{xz}V_x + C_{yz}V_y + C_{zz}V_z - C_{zp} &= 0 \end{aligned}$$

where:

$$\begin{aligned} C_{xx} &= \sum_{i=1}^{ngbrs} c_{x_i}^2, & C_{yy} &= \sum_{i=1}^{ngbrs} c_{y_i}^2, \\ C_{zz} &= \sum_{i=1}^{ngbrs} c_{z_i}^2, & C_{xy} &= \sum_{i=1}^{ngbrs} c_{x_i}c_{y_i}, \\ C_{xz} &= \sum_{i=1}^{ngbrs} c_{x_i}c_{z_i}, & C_{yz} &= \sum_{i=1}^{ngbrs} c_{y_i}c_{z_i}, \\ C_{xp} &= \sum_{i=1}^{ngbrs} c_{x_i}p_i, & C_{yp} &= \sum_{i=1}^{ngbrs} c_{y_i}p_i, \\ C_{zp} &= \sum_{i=1}^{ngbrs} c_{z_i}p_i. \end{aligned}$$

Solving this set of simultaneous equations gives:

$$\begin{aligned} V_x &= \frac{1}{D}(C_{xp}C_{xz}C_{yy} + C_{xz}C_{yp}C_{yz} - C_{xp}C_{yz}^2 \\ &\quad - C_{xz}C_{yy}C_{zp} + C_{xp}C_{yy}C_{zz} \\ &\quad - C_{xy}(C_{xz}C_{yp} - C_{yz}C_{zp} + C_{yp}C_{zz})), \\ V_y &= \frac{1}{D}(C_{xy}C_{xz}C_{zp} - C_{xp}C_{xy}(C_{xz} + C_{zz}) \\ &\quad + C_{xx}(C_{xz}C_{yp} - C_{yz}C_{zp} + C_{yp}C_{zz})), \\ V_z &= \frac{1}{D}(C_{xp}C_{xy}C_{yz} - C_{xx}C_{yp}C_{yz} - C_{xy}^2C_{zp} \\ &\quad + C_{xx}C_{yy}C_{zp}) \end{aligned}$$

where

$$\begin{aligned} D &= C_{xy}C_{xz}C_{yz} - C_{xy}^2(C_{xz} + C_{zz}) \\ &\quad + C_{xx}(C_{xz}C_{yy} - C_{yz}^2 + C_{yy}C_{zz}). \end{aligned}$$

4 Neighboring element selection for smoothing

For most interface shapes, the entire set of mixed elements that share a node with the home element can be included in the smoothing operations. A number of configurations can degrade the accuracy of the smoothing operation. Filters are used during the neighbor list formation to eliminate or reduce

the effect of using problematic neighbors. Each of these filters will be discussed in two-dimensions only due to the relative ease of presentation.

4.1 Spherical smoothing caution

When the interface is oriented nearly parallel to a home element’s edge and is nearly coincident with that edge, a potential problem arises. If the neighbor that has the edge or face in common with the home element is mixed, slight inaccuracies in the gradient normal can produce large errors in the position of the center of convergence. This problem is illustrated in Fig. 8. The home element is element 163. The interfaces in the illustration were computed using the gradient normal. The interface is nearly tangential to the common edge between elements 163 and 133. Similarly the stability points are oriented almost radially. The midchord line is seen extending to the right of the figure. Least squares fitting algorithms work best when the magnitude of the fitting errors are similar in magnitude. In this case, the orthogonal distance of element 133’s midchord plane will dominate the fitting procedure and render the result grossly erroneous if it is allowed to participate in the fit. The algorithm can recognize this potential problem by comparing each neighbors’ stability point displacement from the home element’s stability point with the tangential distance to the home element’s interface dimension. Only elements whose stability points are not radially located with respect to the home element are eliminated from the smoothing. The neighbor rejection

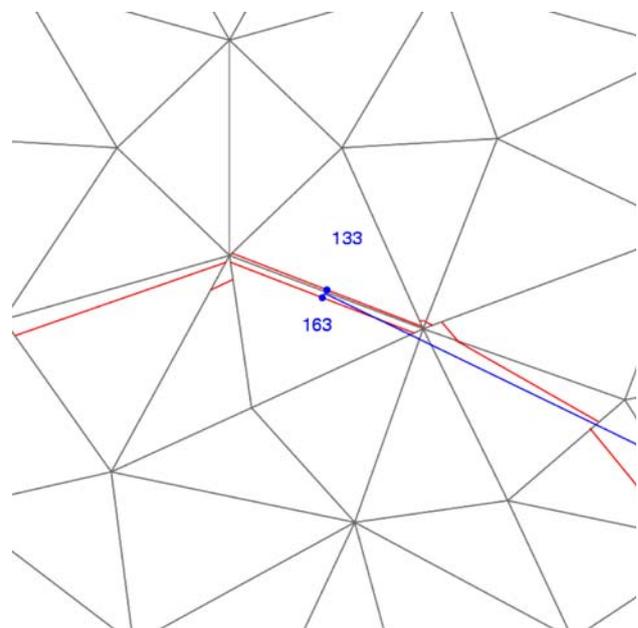


Fig. 8 Illustration of circular smoothing caution

algorithm is

$$\begin{aligned}
 \mathbf{C}_i &= \mathbf{S}_i - \mathbf{H}, \\
 C_{\text{disp}} &= |\mathbf{C}_i - (\mathbf{C}_i \cdot \hat{\mathbf{n}}_{\text{grad}})\hat{\mathbf{n}}_{\text{grad}}| \\
 \text{if}(C_{\text{disp}} < \frac{1}{2}L_{\text{home}}) &\rightarrow \text{reject}
 \end{aligned}
 \tag{8}$$

where L_{home} is the home interface length in two-dimensions and a characteristic diameter derived from the area of the home element's interface is used in three-dimensions, $L_{\text{home}} = \sqrt{\frac{A}{\pi}}$. The term $\hat{\mathbf{n}}_{\text{grad}}$ is the gradient normal for the home element.

4.2 Opposite sides of thin layers

When a material is thin compared to the mesh size, most reconstruction methods will have problems. (An alternate method of avoiding this trouble, the ‘‘onion skin’’ model, will be mentioned in a later section). Thin in this context means that the opposite side of the material boundary is contained within the home element's computational neighborhood. This is illustrated in Fig. 9. The actual material configuration is shown as the blue shaded region and the interfaces that result from the initial volume fraction gradient normal are shown as red line segments. Notice that the elements near the right side of the mesh poorly represent the body; while, elements farther to left side of the mesh represent a typical gradient normal error. Elements 102 and 53 are significantly affected by the gradient stencil enclosing both sides of

the body. The most extreme gradient normal errors occur in element 179. This element contains both sides of the body. There is not a single segment interface that can accurately represent the interface in this element. It is apparent that element 102 should not utilize elements 14, 53, or 124 in its smoothing stencil because the interfaces in these elements lie on the opposite side of the body. The interfaces in these elements don't represent useful smoothing information for the top side of the body. To determine which elements to exclude from the smoothing stencil, the algorithm compares the vector dot product of the home element's gradient normal and each neighboring mixed element's gradient normal with a threshold. The threshold is a user selectable quantity; however, the default value is the cosine of 45 degrees. To illustrate, element 102 would utilize neighboring elements 76, 130, 131, and 226 for the linear smoothing. Element 53 would exclude elements 130, 226, 102, 76, and 179 from its linear smoothing. The results of the linear smoothing are shown in Fig. 10.

4.3 Interface fragmentation

A problem for all interface reconstruction algorithms is under-resolved corners of high curvature interfaces. The reconstructed interfaces for these surfaces tends to emit isolated fragments. While the gradient normal is relatively insensitive to these fragments the smoothing operations can be very sensitive to fragments because the fragments are

Fig. 9 Unsmoothed gradient normal illustration of thin layer caution

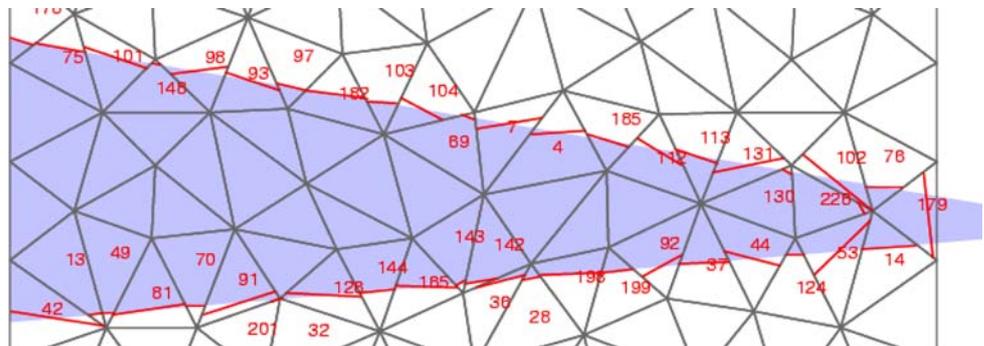
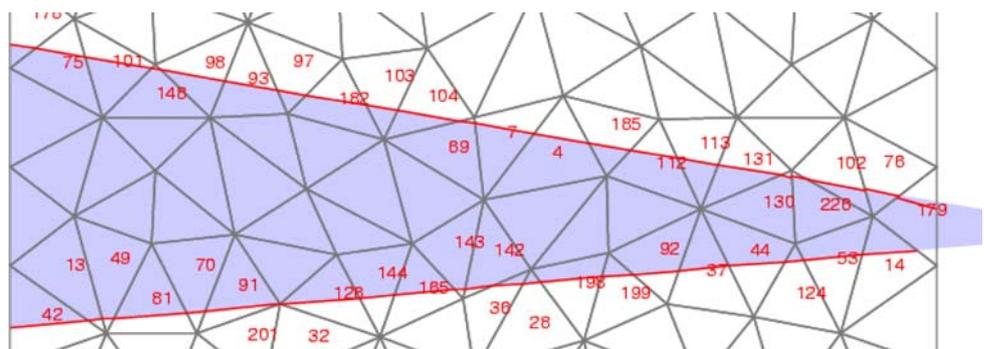


Fig. 10 Smoothed normal illustration of thin layer caution



possibly included in the smoothing operation. The amount of fragmentation is increased by alternating direction advection algorithms. The key to minimizing the influence of fragments is to include a fragment recognition algorithm. After being tagged as a fragment neighboring elements would not use the fragment in their smoothing operations. Fragments are flagged when: none of its neighbors has a volume fraction of one for this material, none of the mixed neighbors contains a volume fraction of the material greater than a threshold (usually a quarter), and there are few neighboring elements that contain this material (usually four; however, this depends upon the element connectivity in each problem).

4.4 Degenerate three-dimensional planar smoothing

To avoid degenerate solutions in the planar smoothing, the derivatives of the objective function with respect to the two angles are examined at the minima. If the values of the two angles can be varied and the objective function is still at its minima the solution is flagged as degenerate. Degeneracy occurs when there are enough suitable neighboring elements to attempt a planar smoothing; however, the points are colinear. This usually occurs when an interface is at a problem boundary. A plot of the objective function for a degenerate solution is shown in Figs. 11 and 12.

5 Choice of smoothed normal method

There are three interface normals that have been discussed, a volume fraction gradient normal, a planar smoothing normal, and a spherical smoothing normal. A quality metric is used to determine which normal will be employed. The quality measure compares interface position and orientation in neighboring mixed elements with the extrapolated values.

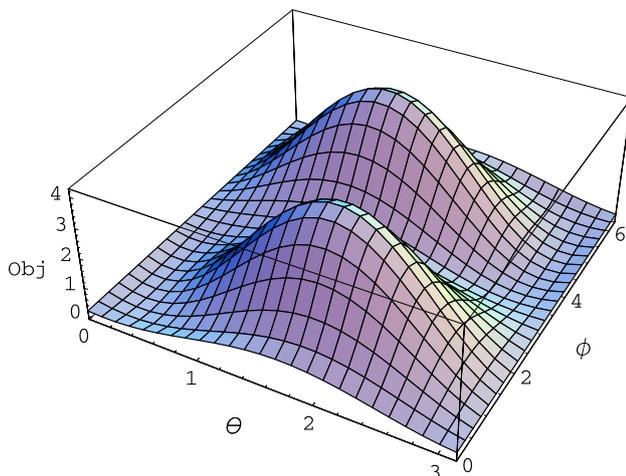


Fig. 11 Plot of objective function for a colinearly degenerate planar fit

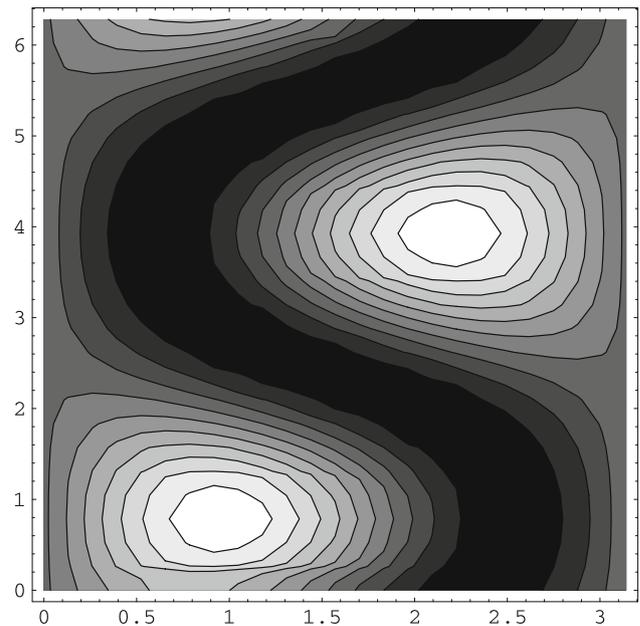


Fig. 12 Contour plot of objective function for a colinearly degenerate planar fit

5.1 Planar smoothing quality measure

A two-dimensional diagram of the planar smoothing quality measure is shown in Fig. 13. The purpose of the quality measure is to develop a quantity that reflects the agreement of the home element's extrapolated interface with the neighboring elements' interfaces. The units of the measure is volume in three-dimensions and area in two-dimensions. The volume of any discrepancy between the extrapolated and actual neighboring interface has two components. The first component is a displacement difference, or how far is the neighboring

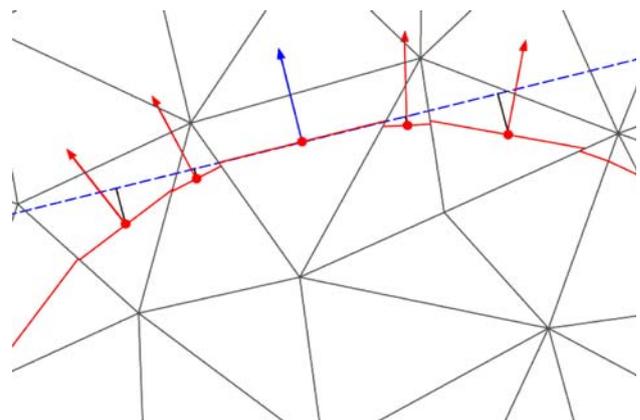


Fig. 13 Planar smoothing quality measure. The blue, dashed interface is the extrapolation of the home element's interface using the planar smoothed normal. The red vectors are the neighboring elements' interface normals. The black segments are the displacements between the extrapolated interface and the neighboring elements' stability points

stability point away from the extrapolated interface. The value of the displacement quality is

$$Q_{disp} = Area_n D_n,$$

in three-dimensions where $Area_n$ is the area of the neighboring element's interface and D_n is the extrapolated displacement which is given by:

$$\hat{\mathbf{h}}_{linear} \cdot \mathbf{S}_n - p = 0$$

where $\hat{\mathbf{h}}_{linear}$ is the planar fit normal in the home element and the neighboring element's stability point position is \mathbf{S}_n . For the two-dimensional expression, the length of the interface is used instead of the area.

The second component of the quality assesses the relative alignment of the neighboring element's normal and the extrapolated normal. If the angle between the two normals is θ the two-dimensional quality contribution is

$$Q_{align} = \frac{1}{2} L_n^2 |\sin \theta|,$$

where L_n is the length of the interface in each neighbor. The three-dimensional quality contribution is

$$Q_{align} = \frac{2A_n}{\pi} |\sin \theta|,$$

where A_n is the area of the interface. The magnitude of the sine of the angle between the two normals is computed using the dot product of the two normals:

$$\sin \theta = \sqrt{1 - (\hat{\mathbf{n}}_{ext} \cdot \hat{\mathbf{n}}_n)^2}.$$

The extrapolated normal, for the planar quality, is the same as the home interface's planar fit normal, $\hat{\mathbf{h}}_{linear}$ because the planar normal is constant along the plane. The neighboring elements' interface normal is $\hat{\mathbf{n}}_n$. The total quality is the sum of the two components:

$$Q_{linear} = \sum_{i=1}^{ngbrs} (Q_{disp} + Q_{align}). \tag{9}$$

5.2 Spherical smoothing quality measure

The spherical smoothing quality measure is illustrated in Fig. 14. It is computed using the same expressions as the planar smoothing except for two differences. First, the extrapolated neighbor normal is computed by passing a vector from the home element's center of convergence through the neighbor's stability point. The resulting vector is normalized to produce a unit vector:

$$\hat{\mathbf{n}}_{ext} = \frac{\mathbf{S}_n - \mathbf{V}_{home}}{|\mathbf{S}_n - \mathbf{V}_{home}|}.$$

The second difference is the method of calculating the neighbor's displacement from the extrapolated interface. The distance is the magnitude of the difference between the home

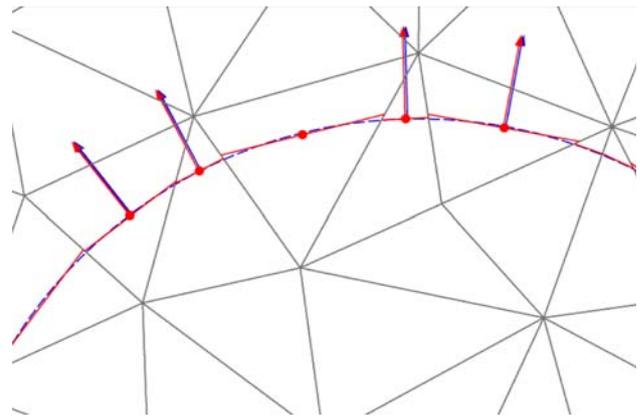


Fig. 14 Spherical smoothing quality measure. The blue, dashed, circular interface is the extrapolation of the home element's fitted interface using the home element's center of convergence. The red vectors are the neighboring elements' interface normals. The blue vectors are the neighboring elements' extrapolated, spherically smoothed normals

element's stability point distance from the center of convergence and the neighboring element's stability point distance:

$$D_n = |R_{home} - R_n|,$$

where

$$R_{home} = |\mathbf{H} - \mathbf{V}_{home}|,$$

$$R_n = |\mathbf{S}_n - \mathbf{V}_{home}|.$$

This results in the two-dimensional spherical quality measure:

$$Q_{CS} = \sum_{i=1}^{ngbrs} \left(L_n |R_n - R_{home}| + \frac{1}{2} L_n^2 \sqrt{1 - (\hat{\mathbf{n}}_{ext} \cdot \hat{\mathbf{n}}_n)^2} \right). \tag{10}$$

6 Algorithm outlines

Two additional parameters are included in the smoothing iterations. Both of these have default values that the user can modify through input. A small number, c , defaulted to 10^{-7} is used as a volume fraction cutoff. The purpose of this threshold is to exclude small volume fractions or volume fractions of very nearly 1. Elements with volume fractions below the threshold and nearly full are excluded from consideration because these usually result from machine roundoff and provide a marginal amount of information to the smoothing. The other parameter is r . The floating point numbers in this paper were implemented as double precision numbers

utilizing 64 bits of machine precision. A value of $r = 10^{-12}$ is used to avoid divisions by zero.

6.1 Two-dimensional planar smoothing algorithm

LinearSmooth (Mesh, home, Interfaces, Vf, VfSum, Fragment, n_{LS} , Q_{LS})

Input: Mesh - the description of elements and nodes
 home - the home element index
 Interfaces - the current interface descriptions for this material
 Vf - volume fractions for current material
 VfSum - sum of volume fractions for current and previous materials
 Fragment - flag indicating if material in an element is a fragment

Output: n_{LS} - the new linear smoothed normal for home element
 Q_{LS} - the linear smoothed quality for home element

vector Ngbrs ← Mesh fetch list of elements sharing a node with home
 vector SmoothingNgbrs.clear()

```

for (ngbr of Ngbrs) {
    if (c < Vf[ngbr] < 1-c &&
        VfSum[ngbr] < 1-c && !Fragment[ngbr]) {
        DotProduct ← Interfaces.Normal[home] o
                    Interfaces.Normal[ngbr]
        if (DotProduct > DotProductThreshold)
            SmoothingNgbrs.push(ngbr)
    }
}
for (ngbr of SmoothingNgbrs)
    Evaluate Sxx, Syy, Sxy, D, and t [eqns 3]

if (abs(t) ≥ r) {
    Evaluate n̂1, n̂2, n̂3, and n̂4 [eqns 4]
    Evaluate Obj(n̂1), Obj(n̂2), [eqn 2]
    Obj(n̂3), and Obj(n̂4)
    n̂LS ← n̂ of min(Obj(n̂1), Obj(n̂2),
                  Obj(n̂3), Obj(n̂4))
    if (n̂LS o Interfaces.Normal[home] < 0)
        n̂LS = -n̂LS
    Evaluate QLS [eqn 9]
}
else
    QLS = Large
    
```

6.2 Two-dimensional spherical smoothing algorithm

CircularSmooth (Mesh, home, Interfaces, Vf, VfSum, Fragment, n_{CS} , Q_{CS})

Input: Mesh - the description of elements and nodes
 home - the home element index
 Interfaces - the current interface descriptions for this material
 Vf - volume fractions for current material
 VfSum - sum of volume fractions for current and previous materials
 Fragment - flag indicating if material in an element is a fragment

Output: n_{CS} - the new circular smoothed normal for home element
 Q_{CS} - the circular smoothed quality for home element

vector Ngbrs ← Mesh fetch list of elements sharing a node with home
 vector SmoothingNgbrs.clear()

```

for (ngbr of Ngbrs) {
    if (c < Vf[ngbr] < 1-c &&
        VfSum[ngbr] < 1-c && !Fragment[ngbr]) {
        DotProduct ← Interfaces.Normal[home] o
                    Interfaces.Normal[ngbr]
        Evaluate Cdisp [eqns 8]
        if (DotProduct > DotProductThreshold &&
            Cdisp > 1/2 Lhome)
            SmoothingNgbrs.push(ngbr)
    }
}
for (ngbr of SmoothingNgbrs) {
    Evaluate ChordLength between ngbr and home
    Evaluate MaxChord = max of ChordLength
    Evaluate Cxx, Cyy, Cxy, Cxp, Cyp, and D [eqns 6]
}

if (abs(D) > r) {
    Evaluate V [eqns 5]
    R = sqrt(Vx^2 + Vy^2)
    if (R ≤ 1000*MaxChord) {
        Evaluate n̂CS [eqn 7]
        if (n̂CS o Interfaces.Normal[home] ≤ 0)
            n̂CS = -n̂CS
        Evaluate QCS [eqn 10]
    }
}
else
    QCS = Large
    
```

```

}
else {
     $Q_{CS} = Large$ 
}
    
```

6.3 Iteration outline

Using the volume fraction gradient normal as the initial interface normal, a material’s interfaces are positioned in each mixed element containing the material and the stability points are calculated. For the first iteration, only the planar smoothing is employed because the gradient normal is quite noisy for unstructured grids. After the interfaces have been repositioned and new stability points are calculated, both the planar and spherical smoothings are performed and the quality measure is calculated for each. The smoothing method that produces the best quality (lowest magnitude) for a mixed element is used for the following iterations for this material in this mixed element. The quality of the initial iteration usually is predictive of the quality that would be produced in later iterations; however, by eliminating the less accurate smoothing, significant computational expense is saved. After each iteration, the interface is repositioned in the mixed elements and the stability point location is updated. Each mixed element will use its smoothing method to improve its normal until either the change in the previous iteration’s normal and the current normal is minimal or a maximum number of iterations has been reached. The iteration maximum is usually reached only when there is a significant discontinuity in the neighboring interfaces, such as if the home element is the central element for a corner. The maximum number of iterations is ten and the successive normal convergence occurs when the dot product of the previous normal and the current normal is greater than 1 minus an epsilon:

$$\hat{\mathbf{h}}^{prev} \cdot \hat{\mathbf{h}}^{curr} > 1 - 10^{-10}.$$

In the first demonstration of the following section, no test of the 180 orientations took more than the minimum number of three iterations.

MaterialReconstruction (Mesh, Vf, VfSum, Fragment, Interfaces)

Input:	Mesh	- the description of elements and nodes
	Vf	- volume fractions for current material
	VfSum	- sum of volume fractions for current and previous materials
	Fragment	- flag indicating if material in an element is a fragment
Output:	Interfaces	- the current interface descriptions for this material

```

for (mixed elements of Mesh) {
    Evaluate  $\hat{\mathbf{h}}_{grad}$  [eqn 1]
    Interfaces.Normal[element]  $\leftarrow \hat{\mathbf{h}}_{grad}$ 
    Position interface to match element material volume
    Interfaces.Position[element]  $\leftarrow$  position
    Interfaces.StabilityPt[element]  $\leftarrow$  midPoint
}

for (mixed elements of Mesh) {
    LinearSmooth(Mesh, element, Interfaces, Vf,
                VfSum, Fragment,  $n_{LS}$ ,  $Q_{LS}$ )
    if ( $Q_{LS} < Large$ ) {
        Interfaces.Normal[home]  $\leftarrow n_{LS}$ 
        Position interface to match element mat volume
        Interfaces.Position[element]  $\leftarrow$  position
        Interfaces.StabilityPt[element]  $\leftarrow$  midPoint
    }
}

vector SmoothingMethod  $\leftarrow$  Linear
for (mixed elements of Mesh) {
    LinearSmooth(Mesh, element, Interfaces, Vf,
                VfSum, Fragment,  $n_{LS}$ ,  $Q_{LS}$ )
    CircularSmooth(Mesh, element, Interfaces, Vf,
                  VfSum, Fragment,  $n_{CS}$ ,  $Q_{CS}$ )
    if ( $Q_{LS} < Q_{CS}$ ) {
        SmoothingMethod[element]  $\leftarrow$  Linear
        Interfaces.Normal[home]  $\leftarrow n_{LS}$ 
    }
    else if ( $Q_{CS} < Large$ ) {
        SmoothingMethod[element]  $\leftarrow$  Circular
        Interfaces.Normal[home]  $\leftarrow n_{CS}$ 
    }
    Position interface to match element mat volume
    Interfaces.Position[element]  $\leftarrow$  position
    Interfaces.StabilityPt[element]  $\leftarrow$  midPoint
}

vector Converged  $\leftarrow$  True
for (iteration 3–10)
    for (mixed elements of Mesh) {
        PreviousNrml  $\leftarrow$  Interfaces.Normal[element]
        if (SmoothingMethod[element] = Linear) {
            LinearSmooth(Mesh, element, Interfaces, Vf,
                        VfSum, Fragment,  $n_{LS}$ ,  $Q_{LS}$ )
            Interfaces.Normal[home]  $\leftarrow n_{LS}$ 
        }
        else if (SmoothingMethod[element] = Circular) {
            CircularSmooth(Mesh, element, Interfaces, Vf,
                          VfSum, Fragment,  $n_{CS}$ ,  $Q_{CS}$ )
            Interfaces.Normal[home]  $\leftarrow n_{CS}$ 
        }
        Position interface to match element mat volume
    }
}
    
```

```

Interfaces.Position[element] ← position
Interfaces.StabilityPt[element] ← midPoint
if (PreviousNrml ◦ Interfaces.Normal[element] <
    1 - ε)
    Converged[element] ← False
}
if (All(Converged)) break

```

7 Two-dimensional planar test

To demonstrate the method, an unstructured grid of triangular elements is constructed and the volume fractions for two materials separated by a planar interface are constructed analytically. The orientation of the interface was varied incrementally for 180 test cases. The interface was pivoted about the center of the mesh. The results shown in Fig. 15 are representative of the entire range of angles. The error for the test is defined as the area between the analytic interface and the PIR interface. The sum of the error over the mesh is 2.04651×10^{-12} and is representative of the entire range of orientations. The area of the mesh is one. The error is due to machine roundoff and convergence bounds and can be considered zero. The maximum error in any mixed element in the mesh is 4.30989×10^{-13} .

There were no explicit boundary conditions used or needed by this model. Interface reconstruction boundary conditions used by many codes involve a layer of ghost elements around the mesh. This is inaccurate because the volume fractions in

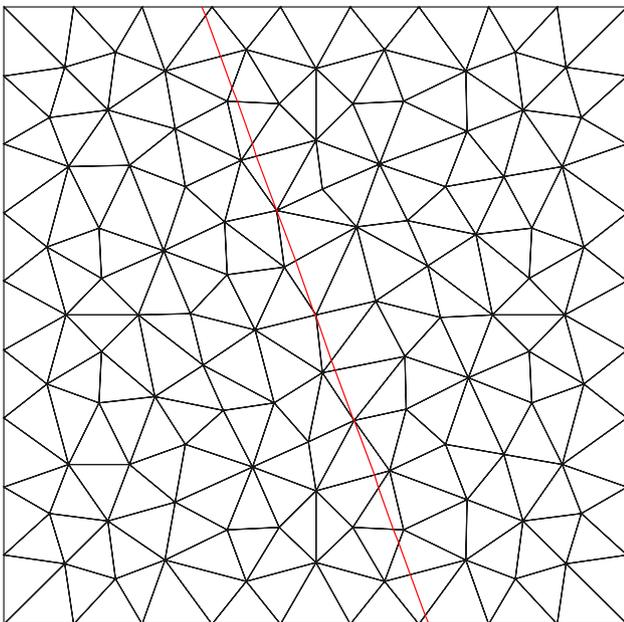


Fig. 15 Two-dimensional planar test on an unstructured grid with triangular elements

the ghost layer are somewhat difficult to robustly specify. The layer of ghost elements are usually described as a reflection of the physical elements along the mesh boundary. The material volume fractions are copied from the physical layer to the ghost layer. This leads to interface orientations that are nearly perpendicular to the problem boundary. For the test case shown the interfaces are continuous and did not require any special treatment to achieve this continuity. This is an advantage in accuracy, algorithm simplicity, and computational expense.

8 Three-dimensional slab test

The three-dimensional, planar smoothing is demonstrated for a slab inclined with respect to the Cartesian mesh. The results shown in Figs. 16 and 17 are representative of the entire range of angles. The error for the test is defined as the volume between the analytic interface and the PIR interface.

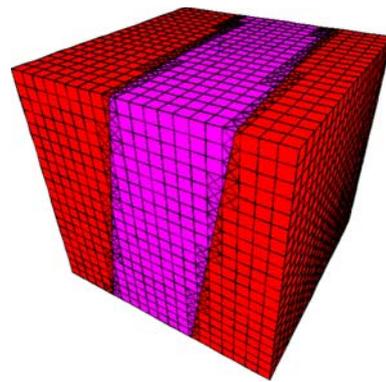


Fig. 16 Three-dimensional planar smoothing test on a structured grid with hexahedral elements

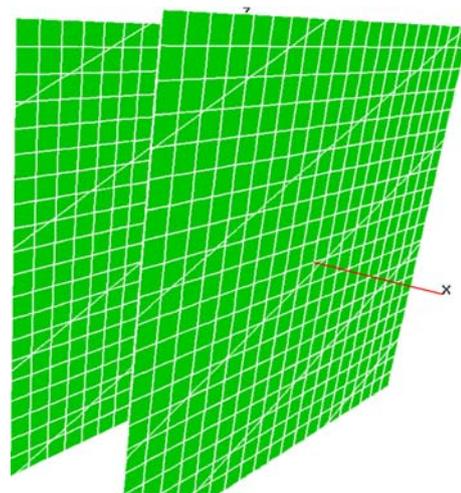


Fig. 17 Three-dimensional planar smoothing test on a structured grid with hexahedral elements

The sum of the error over the mesh is 5.736368×10^{-10} and is representative of the entire range of orientations. The volume of the mesh is one. The error is due to machine roundoff and iteration convergence bounds and can be considered zero. The maximum error in any mixed element in the mesh is 1.04547×10^{-10} . The three-dimensional mesh displays the same boundary accuracy as the previous two-dimensional demonstration. In Fig. 17 the interface polygons for the mesh are displayed. The polygonal edges are very continuous between neighboring elements. In three spatial dimensions this continuity is a very good visual indication that the interfaces are continuous.

9 Three material, two-dimensional, planar test with triangular elements

For this demonstration three materials with planar interfaces that are arranged in an intersecting pattern are tested. In this problem the interface for the red material is calculated first. For the purposes of positioning the blue and green materials within mixed elements containing red material, the red material interface is used to cut off its portion of the mixed elements. Accurate placement of the blue and green materials is simplified by removing this portion of the mixed element. When the gradient normal for the green material is computed rather than using the green material's volume fraction alone, the sum of the red and green materials' volume fractions is used to compute the gradient normal. By summing the vol-

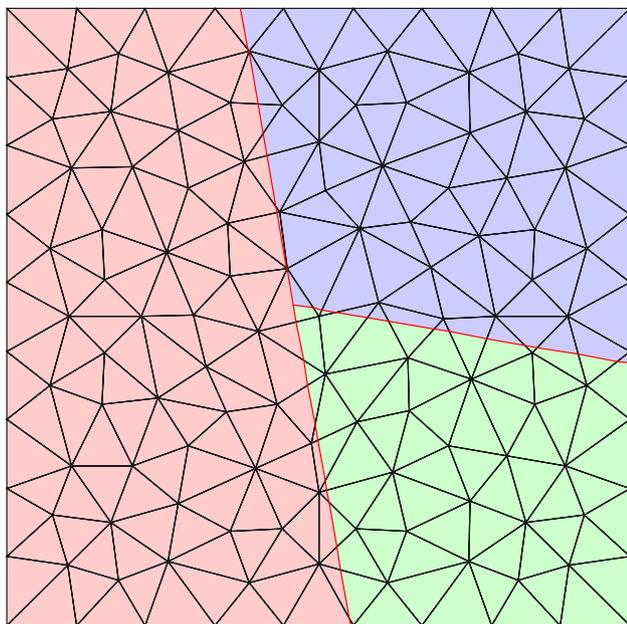


Fig. 18 Mesh for two-dimensional, three material planar smoothing test of intersecting materials on an unstructured grid with triangular elements

ume fractions the interfaces are built on top of each other in multiple material elements. Youngs' refers to this as the "onion skin" model [9]. The "onion skin" interface normal treatment has advantages for the gradient normal; however, it produces problems if the interface is positioned within the element based upon the sum of the volume fractions. The PIR algorithm avoids the positioning problem by removing previous materials from the element's polygonal description for positioning purposes. The interface at the junction of the three materials retains its planar nature.

10 Three material, three-dimensional, planar test in a Cartesian grid

As in the previous two-dimensional test a three material arrangement of materials is placed in a mesh. The mesh for this test is a Cartesian mesh of hexahedral elements. The reconstructed interfaces are shown in Fig. 19 and the interface polygons are shown in Fig. 20. The interfaces have been transported through the mesh using the reconstructed interfaces and the reconstructed interfaces have remained planar and the junction has remained sharp through the translation.

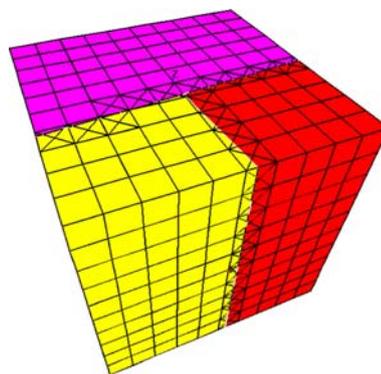


Fig. 19 Mesh for three-dimensional, three material planar smoothing test of intersecting materials on a structured grid with hexahedral elements

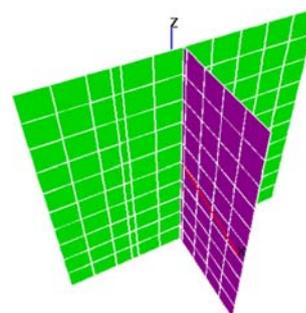


Fig. 20 Interface polygons of three-dimensional, three material planar smoothing test of intersecting materials on a structured grid with hexahedral elements

11 Two-dimensional spherical smoothing test on unstructured grid

The volume fractions for a circle were constructed in four successively refined unstructured meshes to assess the performance of the spherical smoothing. In Fig. 21 the results of the PIR for the three coarser meshes of varying refinement are shown. Several error measures are possible; however, none are useful in calculating an order of convergence for the method. A possible error measure is the area between the input circle and the reconstructed piecewise linear interface. The results of this error measure for the five meshes are presented in Table 1. The mesh resolution, Δ , is computed as:

$$\Delta = \sqrt{1/n},$$

where n is the number of elements in each mesh. A least-squares fit of the error for the respective mesh dimensions yields an accuracy order of 1.98.

12 Three-dimensional spherical smoothing test on a Cartesian grid

A spherically shaped material was placed in a three-dimensional Cartesian grid. The problem was run on a coarse and a fine mesh. The interface plot of the reconstructed interfaces on a coarse mesh are shown in Fig. 22 and the interface plot of the finer mesh run is shown in Fig. 23. The interfaces were used to translate the material through the mesh with

Table 1 Comparison of the area error for a circle centered at $\{\frac{1}{2}, \frac{1}{2}\}$ with a radius of 0.3 for various mesh resolutions

n	Δ	PIR area error
228	0.066	9.62238×10^{-4}
838	0.0345	3.23664×10^{-4}
3,278	0.0175	7.44814×10^{-5}
13,306	0.00867	1.71750×10^{-5}
52,994	0.00435	7.52342×10^{-6}

Fig. 21 Interface polygons of two-dimensional spherical smoothing test of sphere in three, unstructured grids with triangular elements

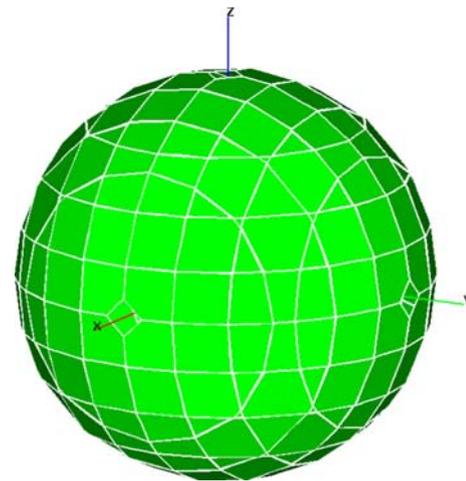
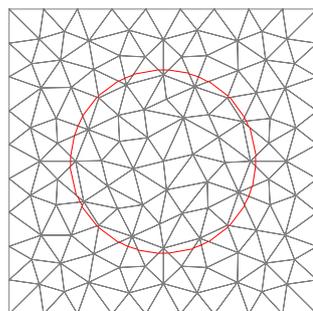


Fig. 22 Interface polygons of three-dimensional spherical smoothing test of sphere in a coarse, structured grid with hexahedral elements

both resolutions. The continuity of the interfaces was very good for both the coarse and fine meshes and this continuity was maintained as the body was moved through the mesh. A mesh convergence study for the three-dimensional problem is not presented because an intersection calculation, similar to the method used in two-dimensions, is not yet available in three dimensions.

13 Two-dimensional oval test of spherical smoothing

The PIR algorithm can be shown to perform well on a curved two-dimensional shape that is not a circle. To address this question volume fractions for a Cassini Oval [7] are computed for two unstructured meshes. The reconstructed interfaces are shown as red line segments and the input profile of the Cassini Oval is shown as the blue curve in Fig. 24. The agreement is excellent. The blue, analytic interface is barely visible under the reconstructed interfaces. The error is the area between the Cassini oval and the piecewise linear interface and is tabulated for five successively refined meshes in Table 2. The least squares fit of the error and mesh dimension gives an order of accuracy of 2.01.

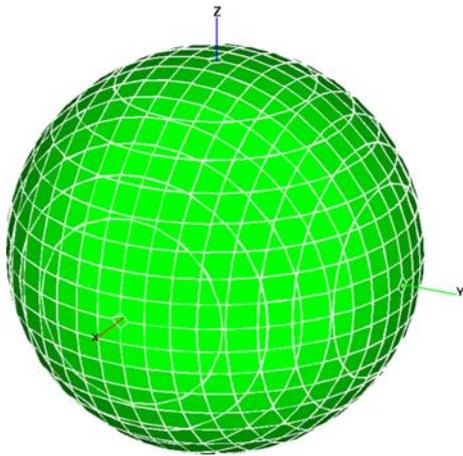


Fig. 23 Interface polygons of three-dimensional spherical smoothing test of sphere in a fine, structured grid with hexahedral elements

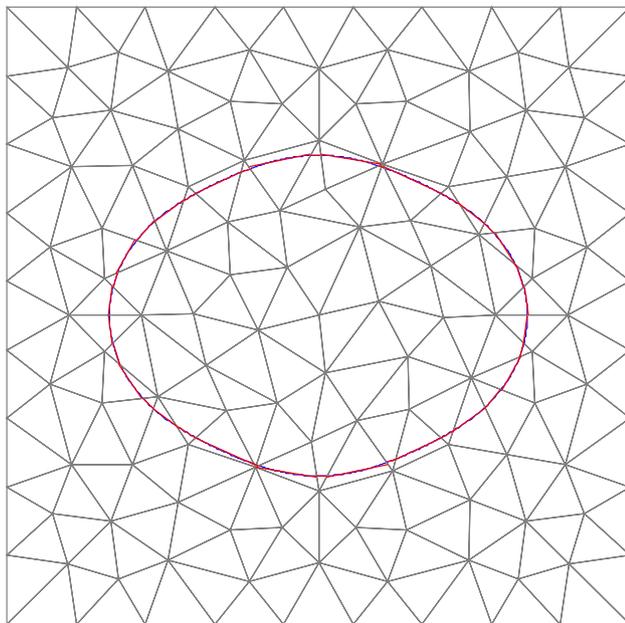


Fig. 24 Interface polygons for two-dimensional spherical smoothing test of Cassini Oval in a coarse unstructured grid with triangular elements

Table 2 Comparison of the area error for a Cassini Oval at $\{\frac{1}{2}, \frac{1}{2}\}$ for various mesh resolutions

n	Δ	PIR area error
228	0.066	9.55529×10^{-4}
838	0.0345	3.20342×10^{-4}
3,278	0.0175	7.46090×10^{-5}
13,306	0.00867	1.60791×10^{-5}
52,994	0.00435	4.47963×10^{-6}

14 Conclusion

A second-order accurate, interface reconstruction algorithm has been presented. The method can be used to reconstruct piecewise linear interfaces in structured and unstructured, two- and three-dimensional meshes. The algorithm is more computationally expensive than other gradient normal based, non-iterative reconstruction techniques; however, the added expense results in higher spatial accuracy.

Acknowledgments The authors acknowledge the essential involvement of Thomas Voth, Joshua Robbins, and John Dolbow in the development of the algorithm on two-dimensional unstructured meshes. This work would not have been possible without their patient observations. Several of the plots were produced by the General Mesh Viewer (GMV) package that was provided by Frank Ortega at Los Alamos National Laboratory.

References

- Dukowicz, J.K.: Conservative rezoning (remapping) for general quadrilateral meshes. *J. Comput. Phys.* **54**, 411–424 (1984)
- Mosso, S.J., Swartz, B.K., Kothe, D.B., Ferrell, R.C.: A parallel, volume-tracking algorithm for unstructured meshes. In: Shiano, P., Ecer, A., Periaux, J., Satofuka, N. (eds.) *Parallel Computational Fluid Mechanics*, pp. 368–375, Elsevier, Amsterdam (1997)
- Nocedal, J., Wright, S.W.: *Numerical Optimization*. Springer, Heidelberg (1999)
- Noh, W.F., Woodward, P.: SLIC—simple line interface calculation. In: van der Vooren, A.I., Zandbergen, P.J. (eds.) *Lecture Notes in Physics*, pp. 330–340, Springer, Heidelberg (1976)
- Scardovelli, R., Zaleski, S.: Analytical relations connecting linear interfaces and volume fractions in rectangular grids. *J. Comput. Phys.* **164**, 228–237 (2000)
- Swartz, B.: The second-order sharpening of blurred smooth borders. *Math. Comput.* **52**, 675–714 (1989)
- Weisstein, E.W.: *CRC Concise Encyclopedia of Mathematics*. CRC Press, Boca Raton (1998)
- Yang, X., James, A.: Analytic relations for reconstructing piecewise linear interfaces in triangular and tetrahedral grids. *J. Comput. Phys.* **214**, 41–54 (2006)
- Youngs, D.L.: Time-dependent multi-material flow with large fluid distortion. In: Morton, K.W., Baines, J.J. (eds.) *Numerical Methods for Fluid Dynamics*, pp. 273–285. Academic Press, New York (1982)