CONF-9605211--2

Title: A PARALLEL, VOLUME-TRACKING ALGORITHM
FOR UNSTRUCTURED MESHES

Author(s): S. J. MOSSO, B. K. SWARTZ
D. B. KOTHE, R. C. FERRELL

Submitted to: Proceedings of Parallel CFD '96
Capri, Italy
May 20-23, 1996

RECEIVED

AUG 26 1996

OSTI

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

MASTER

Los Alamos
NATIONAL LABORATORY

Los Alamos National Laboratory, an affirmative action/equal opportunity employer, is operated by the University of California for the U.S. Department of Energy under contract W-7405-ENG-36. By acceptance of this article, the publisher recognizes that the U.S. Government retains a nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or to allow others to do so, for U.S. Government purposes. The Los Alamos National Laboratory requests that the publisher identify this article as work performed under the auspices of the U.S. Department of Energy.

DISCLAIMER

**Portions of this document may be illegible
in electronic image products. Images are
produced from the best available original
document.**

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

A Parallel, Volume-Tracking Algorithm for Unstructured Meshes

S. J. Mosso^a, B. K. Swartz^b, D. B. Kothe^c and R. C. Ferrell^d

^aHydrodynamic Methods, Los Alamos National Laboratory, MS F663,
Los Alamos, New Mexico 87545 USA

^bMathematical Modeling and Analysis, Los Alamos National Laboratory,
MS B284, Los Alamos, New Mexico 87545 USA

^cFluid Dynamics, Los Alamos National Laboratory, MS B216, Los Alamos,
New Mexico, 87545 USA

^dCambridge Power Computing Associates, Cambridge, Massachusetts

Abstract

Recent enhancements to multidimensional volume-tracking algorithms are presented. Illustrations in two dimensions are given. The improvements include new, local algorithms for interface normal constructions and a new full remapping algorithm for time integration. These methods are used on structured and unstructured grids.

1. Introduction

1.1 Motivation

Many diverse areas of industry benefit from the use of volume of fluid methods to predict the movement of materials. Casting is a common method of part fabrication. The accurate prediction of the casting process is pivotal to industry. Mold design and casting is currently considered an art by industry. It typically involves many trial mold designs, and the rejection of defective parts is costly. Failure of cast parts, because residual stresses reduce the part's strength, can be catastrophic. Cast parts should have precise geometric details that reduce or eliminate the need for machining after casting. Volume of fluid codes will help designers predict how the molten metal fills a mold and where any trapped voids remain. Prediction of defects due to thermal contraction or expansion will eliminate defective, trial mold designs and speed the parts to market with fewer rejections. Increasing the predictability and therefore the accuracy of the casting process will reduce the "art" that is involved in mold design and parts casting.

1.2 Background

For many years, fluid mechanical simulations with multiple materials have been solved by two methods. The earliest method is referred to as Lagrangian. In this method the different materials occupy separate groups of computational cells. The cells are translated and distorted in space as the velocity field evolves in time. In areas of the problem where material shear is pronounced, cells can become non-convex. The Lagrangian method cannot handle these situations. The second method is referred to as Eulerian. The first part of a time step in this method is to evolve the problem's state variables in a Lagrangian fashion. At the end of the time step, a remapping or advection algorithm redistributes the material variables among the original cells, thus restoring their Eulerian status. A conventional, split advection in two or more spatial dimensions fluxes across alternating pairs of opposing cell faces. The order of spatial fluxing is varied to reduce directional biasing and temporal discretization error.

The algorithms presented here were written in FORTRAN 90. The code employs irregular but logically rectangular grids in two dimensions. The work presented in this paper was done in two dimensions. All communication within the code was isolated within a library module, PGSLib. The communication is implemented with the MPI set of communication primitives. When the code is run on serial machines, the communication calls revert to direct memory references within the code.

2. Review of Locally Linear Interfaces in Volume of Fluid Methods

2.1 Algorithm Overview

An accurate and popular approach to locating material interfaces was developed by R. B. DeBar [2] and D. Youngs [6]. The only information stored about the materials (in addition to the material properties for each) is the volume fraction. The interface is reconstructed during each cycle, so that the flux between cells can be accurately apportioned among the materials that reside in the donor cell. In volume of fluid methods the interfaces are locally linear; namely, the interfaces are line segments in two dimensions and planes in three dimensions. In each cell the interface is described by its own equation:

$$\hat{n} \cdot \vec{x} - p = 0$$

Here, $\hat{n}$ is the unit normal and p is the distance of the interface from the origin.

2.2 Traditional Methods for Estimating the Interface Normal

The typical first step in locating the material interface for a cell is to estimate its unit normal. The linear interface is then moved orthogonal to the normal until the

volume behind the interface matches the material volume specified for that cell.

Historically there have been several methods for estimating this normal. For uniform, multidimensional grids there is the notion of a volume fraction function associated with an interface moving smoothly over a fixed cell. An approximation of its gradient estimates the interface's normal. Sadly, the function depends upon the orientation of the interface with respect to the grid. Worse, the size of the gradient varies inversely with the mesh size, confounding the convergence of the gradient's numerical approximation. Even worse: the function's nature varies wildly from cell to cell for irregular grids. See Swartz [Reference 4, pp. 680-681, pp. 707, and Appendix 4].

Nevertheless, a stencil of about nine cells in 2D and a stencil of about twenty-seven cells in 3D has been used to estimate the gradient from a Barth least-squares gradient [1]. (The number of cells here is approximate for an unstructured grid. A cell doesn't necessarily have a fixed number of immediate neighbors.)

2.3 A Simple, Straight Line Test

Figure 1. Angle of "gradient" normal versus volume fraction

To check the accuracy of this "Barth gradient", a test problem was constructed. The test consisted of a three-by-three stack of cells of uniform size with a 10% randomization of the cell edges. Volume fractions associated with a linear interface with a prescribed normal were placed into each of the cells. The interface passed through the center cell at a number of different positions, yielding various volume fractions from zero to one. A plot of the estimated normal's angle obtained using data from a 30 degree normal is shown in Figure 1. The exact answer is, of course, the horizontal line. The angle estimated by the "Barth gradient" method is the dotted curve. Notice that for this example that the error in the calculated normal can be as much as several degrees. This error would lead to a significant error in the advection of material between

cells. Note also that refinement of the grid would not decrease this error. The cause of this error is the nature of the volume fraction field as compared to a

smoothly varying function. The true volume fraction function [Swartz [4] pp. 680-681] associated with a 30-degree-oriented square is a quadratic spline along the normal, rising like x^2 from zero until the interface crosses opposite sides, then varying linearly until it again crosses adjacent sides, when quadratic variation carries it to be constantly 1 far from the square.

2.4 A Circular Interface Test

Figure 2. "Gradient" normal for circular interface

Another test of the "Barth gradient" interface normal is shown in Figure 2. The volume fractions are appropriate to a circular interface in the unstructured, logically rectangular grid. The dotted curve is the exact material interface. The solid lines indicate the calculated interface. Note that the interfaces visually match the circle in a number of cells. At several points in the figure, it is observed that the calculated interface crosses the exact material interface at only one point. More symmetric interfacial error is desirable for accurate advection to take place.

Given an interface normal, the volume behind the interface always reproduces the material volume specified. The preceding discussion shows that the "gradient normal" can contain significant errors. Two new methods for calculating the interface normal will be presented in the following sections.

3. Improved Accuracy Interface Normal Algorithms

3.1 A New Algorithm Reproducing Linear Interfaces

If the volume fraction is fixed but the normal varies, a linear interface moves more at a cell's boundary than near the center. This exemplifies a 200-year-old result in hydrostatics: the envelope of the lines that cut off a prescribed volume from some given figure is tangent to any particular one of these lines at that line's center of mass. More easily [5], the following m -dimensional fact underlies all our iterative algorithms: the mass fraction cut off from some possibly nonuniform body by hyperplanes hinged through the center of mass of one of them is stationary at that

hyperplane. This fact justifies our basic iteration to improve a guessed normal n for a hyperplane in m dimensions cutting off m specified fractions from m nonuniform bodies: for each body, move a hyperplane orthogonal to n to cut off its appropriate mass. Next, connect the centers of mass of the m sections, determining a hyperplane. This hyperplane's normal is the next guess. Swartz and Mosso [5] prove this iteration is quadratically convergent near an "isolated" solution of the problem.

For a linear interface in an unstructured 2D mesh, two iterations are typically coupled: an "inner iteration" for a fixed pair of interfacial cells and an "outer iteration" over all cells and their neighbors. We actually reverse the inner and outer,

Figure 3. Comparison of interfaces produced by "gradient" normal and linear normal outlined in this paper

proceeding as follows. Initial normals are "gradient normals." Next, each interfacial segment is moved to cut out its appropriate volume fraction. Next, fix on interfacial cell C. Its segment has a center of mass, its midpoint. Each nearby interfacial cell with a common vertex has another interfacial midpoint - connect them, yielding a normal. Do this for each such neighbor of C. Average the unit normals so obtained by finding the "closest" (least squares) unit vector to them all. This vector is C's new normal. Record it. Do the same for all interfacial cells, and then move all segments according to their new normals and given volume fractions. This procedure completes one stage of our outer/inner iteration. Repeat to

three digit convergence, at most four stages. Linear interfaces are reproduced exactly.

3.2 Interfacial Normal Estimates via Circular Interpolation

Figure 4. Circular normal construction.

Instead of vertex neighbors and least squares averages of unit vectors, the following technique seems equally appropriate for curved interfaces. Begin as before. Midpoints of three neighboring interfacial segments are interpolated by a circle. See Figure 4. Its normal, at the central of the three midpoints, is chosen for the central cell's new normal, and recorded. Proceed through the cells, and move the linear segments. This completes one stage of the iteration. This method is used when the center of the circle is outside the cell of interest and the sphere's radius is less than a maximum value. Outside this range, the linear interface calculation is used. It

should be noted that this method also converges in three or four iterations.

Under reasonable assumptions, the segments' centers of mass converge as the mesh size decreases with second-order accuracy to a curved interface [4 Section 8] which can lead to first-order accuracy for these "circularly" averaged normals.

4. Time Integration with a Full Remap

To evolve accurately the material motion in time, the time step is broken into a Lagrangian phase and a remap. In the Lagrangian portion of the time step, the cells are moved in space, but no material is transferred between cells. At the end of the time step, a remapping is performed. During the remap, the material is redistributed to copies of the cells in their starting position. The remap is traditionally done through a direction split advection. The order of the directions is alternated to avoid biasing. But material which moves to cells that only share a common vertex with a donor cell must receive material from the donor cell after it has been transferred through an intermediate cell that shares a face with the donor cell. By passing material through intermediate cells, the material is combined with the material in the intermediate cell, and material information is lost. Sharp gradients tend to be dissipated. In this section, a complete remap algorithm is presented. In this way, material is directly transported from the donor cell to all of its recipients. The remap saves interface reconstruction steps because the material interfaces are only reconstructed once per remap step. The single reconstruction of the full remap is contrasted with direction split schemes, where the

interfaces are reconstructed before each of the advection directions. The remap algorithm is more complex, hence the savings due to the reduced number of reconstructions may be offset by the increased complexity of the remap.

The first step in the remap is to determine the position of the cells at the end of the Lagrangian phase. To do this, the velocities are evolved from their starting values to the values at the end of the time step. A suitable average of the starting and ending velocities is used to move the cell vertices. The volume fractions are, as yet, unchanged. The material interfaces are now constructed on these new cells. They are divided into material subcells. Each material in a mixed cell is represented by its own polyhedron. Then, each of the original Eulerian cells that overlaps it constructs the intersection of these polyhedra with itself. One thereby calculates a new volume fraction for each material in the Eulerian cells to begin the next time step. The algorithm for constructing this intersection is based upon the algorithm developed by Patrick O'Rourke [3]. The algorithm uses a set of simple, primitive operations. "Intersection" is the operation of constructing the location of intersection between two cell edges. "Inside" is the operation of determining if a point is within a polygon. "Left" is the operation of determining if a given point is to the left of an edge: if the vector product of the edge-vector with the vector joining the edge origin with the point in question is greater than zero, the point is to the left of the edge. The two polygons are passed to the intersection algorithm as a series of ordered vertices. The vertices are arranged such that as we traverse the perimeter of the polygon, the vector product formed by successive edges is positive. The algorithm starts with a point on both polygons. It then proceeds by determining which point is to the "left" and "inside" of the other polygon. In this way, only a single pass is made through each polygon's list of vertices. This single pass makes the algorithm efficient and of linear complexity.

To show the accuracy of the interface reconstruction and the remap, a circular interface was placed in an irregular mesh. The starting position is shown in Figure 5. The disc is next translated at constant velocity through the mesh for

Starting Position

Mid Problem

Ending Position

Figure 5. Comparison of interfaces for the circular interface at 3 problem times

roughly ten cells, and the velocity is reversed so that the disc is advected back to its starting position. The Courant number was 0.4. The ending configuration is shown in Figure 5. The comparison of the starting and ending volume fractions is one measure of the fidelity of the remap. This average error is 0.6 percent per mixed cell.

5. Conclusion

An efficient, iterative method for calculating interface normals was presented. It is exact for linear interfaces. More generally, the algorithms will produce a second-order accurate interface, a large improvement over the standard first order accurate, approximate gradient method. A remap algorithm was presented which very accurately transports materials in a volume of fluid code. The underlying, iterative algorithm to find a linear interfacial normal is proven to be quadratically convergent in m dimensions [5]. The interface algorithms and the remapping algorithm each have extensions to three dimensions (and to r - z coordinates, for example, via variable density r for the centers of mass of the interfacial segments). These extensions will be explored in future work.

6. References

1. T. J. Barth, "Recent Developments in High Order K-Exact Reconstruction on Unstructured Meshes", AIAA-93-0668, 31st Aerospace Sciences Meeting and Exhibit.
2. R. B. DeBar, "Fundamentals of the KRAKEN code", Report UCID-17366, Lawrence Livermore Laboratory, Livermore, Ca, 1974, 17pp.
3. J. O'Rourke, Computational Geometry in C, Cambridge University Press.
4. B. Swartz, "The Second-order Sharpening of Blurred Smooth Borders," Mathematics of Computation, vol. 52, pp. 675-714, 1989.
5. B. Swartz and S. Mosso, "Swiftly Slicing Specific Fractions from n Bodies in n -Space", Report LA-UR-96-2353, 1996.
6. D. L. Youngs, "Time-Dependent Multi-Material Flow with Large Fluid Distortion", Numerical Methods for Fluid Dynamics, K.W. Morton and M.L. Norman, editors, pp 187-221 (1986).