Available online at www.sciencedirect.com
SCIENCE DIRECT®
Journal of Computational Physics 199 (2004) 465–502
JOURNAL OF COMPUTATIONAL PHYSICS
www.elsevier.com/locate/jcp

Second-order accurate volume-of-fluid algorithms for tracking material interfaces ☆

James Edward Pilliod Jr., Elbridge Gerry Puckett *

Department of Mathematics, University of California, One Shields Avenue, Davis, CA 95616, USA

Received 1 February 2003; received in revised form 24 December 2003; accepted 31 December 2003
Available online 20 July 2004

Abstract

We introduce two new volume-of-fluid interface reconstruction algorithms and compare the accuracy of these algorithms to four other widely used volume-of-fluid interface reconstruction algorithms. We find that when the interface is smooth (e.g., continuous with two continuous derivatives) the new methods are second-order accurate and the other algorithms are first-order accurate. We propose a design criteria for a volume-of-fluid interface reconstruction algorithm to be second-order accurate. Namely, that it reproduce lines in two space dimensions or planes in three space dimensions exactly. We also introduce a second-order, unsplit, volume-of-fluid advection algorithm that is based on a second-order, finite difference method for scalar conservation laws due to Bell, Dawson and Shubin. We test this advection algorithm by modeling several different interface shapes propagating in two simple incompressible flows and compare the results with the standard second-order, operator-split advection algorithm. Although both methods are second-order accurate when the interface is smooth, we find that the unsplit algorithm exhibits noticeably better resolution in regions where the interface has discontinuous derivatives, such as at corners.

AMS: 65M06; 65M12; 76M20; 76M25; 76-04

Keywords: Advection algorithms; Fluid interfaces; Interface tracking; Material interfaces; Volume-of-fluid; Volume tracking

1. Introduction

There are numerous instances in which it is necessary to reconstruct and/or track the boundary between two materials or states of matter in a numerical computation. Examples include numerical models of fluid

☆ Work of both authors at U.C. Davis supported by the National Science Foundation under grants DMS-9104472 and DMS-9404410 and by the Applied Mathematical Sciences Program in the Mathematical, Information and Computational Sciences (MICS) Division of DOE's Office of Energy Research under contracts DE-FG03-95ER25271, DE-FC02-01ER25473 and DE-FG02-03ER25579. Work of the first author at the Lawrence Berkeley Laboratory was supported by the Applied Mathematical Sciences Program in MICS under contract DE-AC03-76SF00098 and by the Defense Nuclear Agency under contract IACRO 96-3075.
* Corresponding author.
E-mail address: egpuckett@ucdavis.edu (E.G. Puckett).

0021-9991/$ - see front matter © 2004 Published by Elsevier Inc.
doi:10.1016/j.jcp.2003.12.023

jetting devices [1–4], weld pools [5], molten metal [6,7], semiconductor device etching [8–11], thin flame models of combustion [12–15] and clouds [16].

An overview of the state of the field in the early 1980s may be found in [17]. Since that time, many new approaches to tracking the interface between two materials have appeared. A notable example is the level set approach of Osher and Sethian [8,18–22]. There has also been considerable work devoted to developing algorithms that approximate the front as a collection of line segments in two space dimensions or polygons in three space dimensions [23,24] and on boundary integral methods [25,26]. A good idea of the current state of the field may be found in the recent reviews by Scardovelli and Zaleski [27] and Sethian and Smereka [28].

In this article we study a class of interface tracking algorithms known as *volume-of-fluid* methods. In a volume-of-fluid method the motion of the interface itself is not tracked, but rather the volume of each material in each cell is evolved in time and the interface at the new time is reconstructed from the values of the volumes at this new time. For this reason volume-of-fluid methods are sometimes referred to as *volume tracking* methods [29].

The basic idea behind a volume-of-fluid method is as follows. [1] Suppose that we wish to track the interface between two materials, say a dark fluid and a light fluid, in two dimensions. We begin by covering the problem domain with a grid with spacing $h = \Delta x = \Delta y$. With each grid cell we associate a number $f_{i,j}$ that represents the amount of dark fluid in the i,jth cell,

$$f_{i,j}h^2 = \text{volume of dark fluid in the } i,j\text{th cell.}$$

The number $f_{i,j}$ is called the *volume fraction* (of dark fluid) in the i,jth cell. It is apparent that

$$0 \leqslant f_{i,j} \leqslant 1, \tag{1}$$

that the volume fraction associated with the light fluid is $1 - f_{i,j}$ and that a portion of the interface lies in the i,jth cell if and only if $0 < f_{i,j} < 1$. The discrete variable $f_{i,j}$ is a discretization of the characteristic function associated with the dark fluid,

$$f(x,y) \equiv \begin{cases} 1 & \text{if there is dark fluid at the point } (x,y), \\ 0 & \text{if there is light fluid at the point } (x,y), \end{cases} \tag{2}$$

in the sense that

$$f_{i,j}h^2 \approx \int\int_{i,j\text{th cell}} f(x,y)\mathrm{d}x\,\mathrm{d}y.$$

Since the fluid type does not change along particle paths in an incompressible, non-reacting flow, the characteristic function f is passively advected with the flow. Hence, f satisfies the advection equation,

$$f_t + uf_x + vf_y = 0, \tag{3}$$

where $\vec{u} = (u,v)$ denotes the fluid velocity. If the flow is incompressible, then $\vec{u}$ satisfies

$$u_x + v_y = 0. \tag{4}$$

Multiplying (4) by f and adding it to (3) we obtain a conservation law for the characteristic function f,

$$f_t + (uf)_x + (vf)_y = 0. \tag{5}$$

[1] Here and in the remainder of this article we restrict the discussion to uniform square grids and two space dimensions. Neither of these restrictions are necessary. We employ them merely for simplicity and clarity of exposition.

Eq. (5) reflects the fact that in an incompressible flow conservation of mass is equivalent to conservation of volume, and hence conservation of f.

In a compressible flow the velocity field $\vec{u}$ does not satisfy (4) and hence f is not conserved. However the mass of each material is conserved and therefore it is important that a numerical method for modeling this phenomena also conserve the mass of each fluid. It is relatively easy to design a volume-of-fluid interface tracking algorithm that does this [30–32]. Volume-of-fluid algorithms are the basis for most of the large application codes that are used at the national laboratories to model multi-phase, compressible phenomena on Eulerian grids [33–36]. These codes are also used extensively by geophysicists to model meteor impacts and related problems [37–39].

Recently, there have been several important improvements to the basic volume-of-fluid methodology for modeling compressible flows. Colella et al. [30] have developed a model of interface motion in compressible flow in which (5) is modified by the addition of a term that accounts for the effect of isentropic volume changes due to changes in the pressure; i.e., changes in the specific volume V of the form $(\partial V/\partial P)_S$. Their method allows one to model disparities in the compressibility of two materials (e.g., air and water) on a subgrid scale. Puckett and Saltzman [32] have developed an algorithm for tracking gas interfaces in three space dimensions that is based on these ideas Miller and Puckett [31] have developed a similar model for tracking the interface between two solids at very high pressures and temperatures (e.g., magmas) in the hydrostatic limit (i.e., without strength) while Miller and Colella [40] have developed a method for modeling the motion of the interface between a gas and a solid (with realistic constitutive laws) in shock physics applications.

In this article we restrict ourselves to consideration of incompressible flow problems. One might expect the incompressible advection problem to be less difficult than the corresponding problem in compressible flow. However our experience has been that this is generally not true. The difficulty in modeling incompressible flow arises because f is also constrained by the maximum principle (1) but numerical errors in estimating the fluxes in (5) lead to overshoot and undershoot in the values of f. In practice we have found that for the simple advection problems considered here these errors are on the order of machine zero (e.g., see Tables 14 and 20). For more difficult problems they tend to be on the order of one hundredth of a percent (e.g., see the computations in [41] and [21]).

Since in an incompressible flow f satisfies (5), the time update of the discrete variable $f_{i,j}$ can be accomplished with a conservative finite difference method. One can therefore draw on the vast body of knowledge for high-resolution numerical methods for conservation laws [42–45] to devise a method for updating f numerically. In this article we present a volume-of-fluid advection algorithm that is based on ideas developed by Bell et al. [42] to construct a finite difference method for modeling solutions of scalar conservation laws.

Volume-of-fluid methods have been in use for several decades. In one of the earliest implementations of these methods DeBar [46] used a volume-of-fluid algorithm in a two-dimensional Eulerian method to model compressible multi-phase flow. Another early algorithm of this type is the simple line interface calculation (SLIC) method of Noh and Woodward [47]. SLIC and its variants have been very widely used. For example, Colella et al. [48–50] used it to model shock wave refraction at a gas interface. In [12] Chorin developed an improved version of SLIC in order to model flame propagation and combustion. Ghoniem et al. [13] and Sethian [15] used Chorin's version to model turbulent combustion, while Whitaker [51] used it to model Hele–Shaw flow. Another well-known volume-of-fluid algorithm is the VOF algorithm of Hirt and Nichols [52]. [2] Several codes based on the VOF algorithm, namely SOLA-VOF [52,53] and its descendants NASA-VOF2D [54], NASA-VOF3D [55], RIPPLE [56,57] and FLOW3D [58] have been, and continue to be, widely used by researchers to model interfaces and free surfaces in industrial applications. For example,

[2] Many workers use the acronym "VOF" – which stands for "Volume-of-fluid" – to refer generically to any volume-of-fluid algorithm. However, we refrain from doing so since others use it to refer specifically to Hirt and Nichols' algorithm and the associated fluid dynamics code SOLA-VOF [52,53].

researchers at Xerox have used a modified version of these codes to model the flow in thermal ink jet devices [2,3] and they have been used extensively by material scientists to model weld pools [5] and solidifying droplets [6,7]. However, the volume-of-fluid algorithms in all of the methods just referred to are built around relatively crude interface reconstruction algorithms that rely on a piecewise constant or "staircase" representation of the interface, such as the one shown in Fig. 1(c), and advection algorithms that are at best first-order accurate.

More modern volume-of-fluid interface reconstruction methods use a linear approximation to the interface in each multifluid cell [16,29,32,59–62]. This results in a piecewise-linear approximation to the interface as shown in Fig. 1(d). However, as we demonstrate (prove) in Section 3, a linear approximation to the interface in each cell is not sufficient to guarantee a second-order accurate approximation to the interface.

In Section 3 we demonstrate (numerically) that a sufficient condition for a volume-of-fluid interface reconstruction algorithm to be second-order accurate on smooth interfaces, is for the algorithm to reproduce linear interfaces exactly. We show that two interface reconstruction algorithms introduced by the authors (the least squares volume-of-fluid interface reconstruction algorithm (LVIRA) [61] and the efficient least squares volume-of-fluid interface reconstruction algorithm (ELVIRA) [63]) have this property. These second-order accurate piecewise-linear interface reconstruction algorithms have been used extensively to model a variety of compressible and incompressible flows, including the motion of fluid interfaces in variable density incompressible flow [21,41], shock refraction in gases [64–66], shock refraction and impact jetting in solids in the hydrostatic limit [31,39,67], thin flame models for tracking deflagrations and detonations [14,68], and shock waves in coupled gas dynamics-solid mechanics applications [40].

There seems to be a widely held belief in the CFD community that one cannot obtain high-order accuracy with a volume-of-fluid algorithm [24, p. 26]. Perhaps this is due to the widespread use of SLIC and VOF, which are at best first-order accurate and can easily fragment a smooth front (e.g., see Figs. 10 and 11 and Figs. 6 and 8 of [69]). One of the goals of this article is to demonstrate that one can construct high-

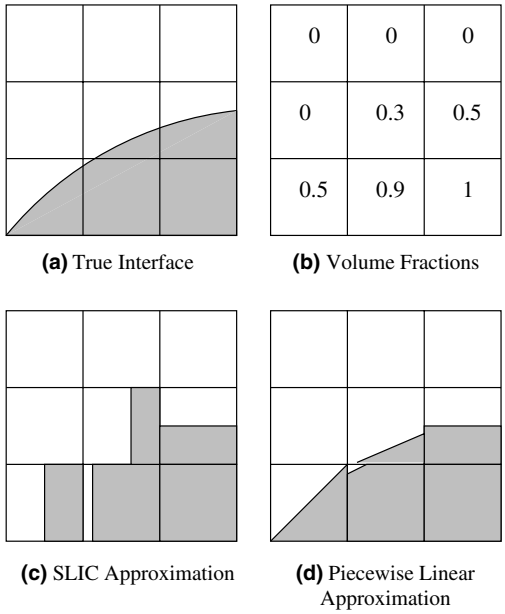

Fig. 1. Volume-of-fluid methods represent an interface (a) by storing volume fractions associated with the interface as shown in (b). An approximation to the interface is produced using an interface reconstruction method such as SLIC, shown in (c), or a more general piecewise linear approximation as in (d).

order accurate volume-of-fluid interface tracking algorithms that are as effective, and for some problems more effective, than competing methods. There are four principal reasons for the effectiveness of volume-of-fluid algorithms:

(1) Volume-of-fluid algorithms naturally conserve the mass of each fluid. For incompressible flow this is because the advection algorithm is a conservative discretization of the conservation law (5) for f, which is equivalent to the mass conservation equation. In a compressible flow the mass of each fluid component must still be conserved even though the characteristic function f is not. In a volume-of-fluid method this can be easily arranged by appending a separate conservation law for the mass of each fluid to the original system of conservations laws [30–32].
(2) In both compressible and incompressible flows it is desirable, if not essential, that the location of the interface as determined by the interface tracking algorithm coincide with the location of the jump in density (and possibly other quantities) as determined by the underlying discretization of the fluid flow equations. Since the flux of a conserved quantity can be written in terms of the fluid volume that crosses a cell edge, it is a simple matter to enforce these constraints in a volume-of-fluid method.
(3) Volume-of-fluid methods automatically handle changes in the global topology of the front, such as fronts that break up into droplets or fronts that collide with themselves and merge. This eliminates the algorithmic complexity that can occur when the front is modeled by a collection of line segments or polygons. Furthermore, the logical structure of the algorithm is not significantly more complicated in three dimensions than in two. This is in contrast to polygonal representations of the front in which the logical complexity increases substantially in going from two to three dimensions. (A discussion of the complexity issue can be found in [11].)
(4) The work required to update the front location is entirely local; typically one needs the velocity and volume fractions in a 3×3 block (or $5 \times 5 \times 5$ block in 3D) of cells to update the volume fraction in the center cell. Since the interface is a codimension 1 set, the computational work required to update the location of the interface is typically $O(N^{d-1})$ for a problem on a grid with N^d cells in $d \geqslant 2$ space dimensions. Thus, the work required to update the front location is small compared to the work required to update the underlying velocity field. The local nature of volume-of-fluid algorithms also makes them amenable to efficient parallelization strategies.

In conclusion, volume-of-fluid methods can be naturally formulated in conservative finite difference form, thereby ensuring that the mass of each material is conserved and that the location of the interface will coincide with jumps in density and other fluid properties, they handle changes in the topology of the front without an increase in algorithmic complexity or computational cost and the work required to update the front is small compared to the work required to update the underlying velocity field.

The remainder of this paper is organized as follows. In Section 2 we describe six volume-of-fluid interface reconstruction algorithms, including the two new second-order accurate algorithms. In Section 3 we study the spatial accuracy of these methods by using them to reconstruct various stationary interfaces. In Section 4 we discuss operator split advection algorithms and study the time accuracy of second-order accurate, operator split advection by using it, in combination with each of the six interface reconstruction algorithms, to approximate various interface shapes undergoing translation and rotation. In Section 5 we describe a second-order accurate, unsplit advection algorithm we have developed and examine the accuracy of this algorithm by applying it to the problems studied in Section 4. We state our conclusions in Section 6.

2. Volume-of-fluid interface reconstruction algorithms

In this section we consider the following problem. Let Ω be a region in the plane $\mathbb{R}^2$ and let $\vec{z}(s) = (x(s), y(s))$ for $0 \leqslant s \leqslant 1$ be a piecewise smooth interface. In all of the examples we study $\vec{z}$ is C^0 and piecewise C^2. In addition, in all but one of these examples $\vec{z}$ is a closed curve $\vec{z}(0) = \vec{z}(1)$, the exception being

when $\vec{z}$ is a line, in which case $\vec{z}(0), \vec{z}(1) \in \partial\Omega$. We think of $\vec{z}$ as separating Ω into two regions of fluids which we refer to as "light" and "dark" fluid. Now cover Ω with a square grid Ω^h where h denotes the grid spacing. For each $0 \leqslant i \leqslant M$ and $0 \leqslant j \leqslant N$ let $f_{i,j}$ represent the fraction of the i,jth cell's volume that is occupied by the dark fluid. The problem is to reconstruct the interface $\vec{z}$, given only the grid Ω^h and the volume fractions $f_{i,j}$, $i = 0, \ldots, M$ and $j = 0, \ldots, N$. We refer to an algorithm for solving this problem as a volume-of-fluid *interface reconstruction algorithm*.

Each of the algorithms described in this section produces a linear approximation to the interface in each multifluid cell; i.e., each cell which satisfies $0 < f_{i,j} < 1$. (We use the terms multi-fluid and multi-material interchangeably.) In general, these piecewise linear approximations are not continuous. All of the algorithms except for SLIC use the volume fractions in a 3×3 block of cells to determine the approximate interface in the center cell of the block. SLIC uses only the volume fractions in a 3×1 block of cells to determine the approximate interface in the center cell of the block.

All of the algorithms described below except for SLIC also return a slope, or equivalently, a vector $\vec{n}$ normal to the interface. In this article we adopt the convention that $\vec{n}$ always points away from the dark fluid. The normal vector $\vec{n}_{i,j}$ in the i,jth cell together with the volume fraction $f_{i,j}$ uniquely determines the approximate linear interface in that cell. Thus, since the volume fraction $f_{i,j}$ is given, all of the algorithms described below (but SLIC) are simply rules for determining a unit normal vector from the values of the volume fractions in some neighborhood of the i,jth cell.

In what follows we often will replace the problem of finding the unit normal $\vec{n}$ to the approximate interface with that of finding its slope $\tilde{m}$, since for many of the interface reconstruction algorithms we study this results in very simple formulas for $\tilde{m}$. However, this approach is problematic when the best linear approximation is a vertical, or nearly vertical, line. This can be remedied by rotating the 3×3 block of cells by 90° and applying the interface reconstruction algorithm in the new coordinate frame. In our discussion of the various interface reconstruction algorithms below we will sometimes omit details related to this coordinate transformation.

Because of the difficulty in representing a vertical line in slope intercept form, we have found that in practice it is usually preferable to represent the approximate interface in each multi-material cell as a unit vector $\vec{n} = (n_x, n_y)$ normal to the approximate interface together with its distance d from the origin. In this case the line satisfies the following equation

$$n_x x + n_y y = d.$$

We have found that this is a better computational representation than the slope intercept form

$$y = \tilde{m}x + b.$$

2.1. Simple Line Interface Calculation

This algorithm is due to Noh and Woodward [47]. Their version of SLIC is a strictly one-dimensional method in which one uses the information in a 3×1 block of cells to reconstruct the interface in the middle cell. This necessitates the use of an operator split advection algorithm (described in Section 4) when one is solving problems in two and three space dimensions. Chorin [12] (see also [51]) proposed a variant of the original SLIC algorithm in which he uses the volume fraction information in a 3×3 block of cells to reconstruct the interface in the center cell. However in general this modified algorithm still does not yield an approximation to the interface that is independent of the sweep direction and hence one is still constrained to use an operator split advection algorithm. Here we study the original version of SLIC as described in [47].

In the SLIC method the reconstructed interface is composed of one or (in two cases) two lines aligned with the grid. The interface geometry and location is based on the values of the volume fractions in the a

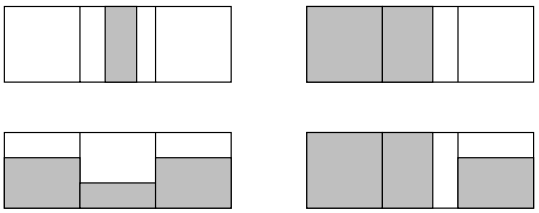

Fig. 2. Four of the nine possible cases in SLIC. The other five cases are obtained by switching light and dark, or by switching left and right.

row of three cells centered on the cell containing the interface. Fig. 2 shows the interface geometry in four of the nine possible cases. The other five cases are obtained by switching light and dark fluid, or by switching left and right. Note that the approximate interface is *not* necessarily perpendicular to the sweep direction. Since SLIC always returns horizontal or vertical lines, it obviously does not exactly reproduce all linear interfaces and hence it is at best first-order accurate.

2.2. The center of mass algorithm

This method is due to Saltzman [32]. In the center of mass algorithm, one considers the dark fluid to have a mass density of 1, and the light fluid to have no mass. To determine the approximate interface in the center of a 3×3 block of cells one first determines the center of mass $(\bar{x}, \bar{y})$ of the 3×3 block and then finds a unit vector that points from this point to the center of the center cell. This vector is taken as the unit normal $\vec{n}$ to the approximate interface.

To see if this method reproduces all lines exactly we consider the exact version of the method. In other words, to find $(\bar{x}, \bar{y})$ we integrate exactly rather than by using a numerical approximation to the integrals as one would in practice. Let h be the cell width of the 3×3 block, and choose a coordinate system in which the center of the center cell is at the origin. If the center of mass algorithm reproduces all lines exactly, then in particular for arbitrary m it must reproduce the line $y = mx$ exactly. Let $(\bar{x}, \bar{y})$ be the coordinate of the center of mass of this 3×3 grid. We can find $(\bar{x}, \bar{y})$ by

$$\begin{aligned}\bar{x} &= \int_{-1.5h}^{1.5h} \int_{-1.5h}^{mx} x \,\mathrm{d}y \,\mathrm{d}x \Big/ \int_{-1.5h}^{1.5h} \int_{-1.5h}^{mx} \mathrm{d}y \,\mathrm{d}x = \int_{-1.5h}^{1.5h} mx^2 + 1.5hx \,\mathrm{d}x \Big/ \int_{-1.5h}^{1.5h} mx + 1.5h \,\mathrm{d}x \\ &= \frac{2.25mh^3}{4.5h^2} = \frac{mh}{2}\end{aligned}$$

and

$$\begin{aligned}\bar{y} &= \int_{-1.5h}^{1.5h} \int_{-1.5h}^{mx} y \,\mathrm{d}y \,\mathrm{d}x \Big/ \int_{-1.5h}^{1.5h} \int_{-1.5h}^{mx} \mathrm{d}y \,\mathrm{d}x = \frac{1}{9h^2} \int_{-1.5h}^{1.5h} m^2x^2 - 2.25h^2 \,\mathrm{d}x \\ &= \frac{1}{9h^2}(2.25m^2h^3 - 6.75h^3) = \frac{m^2h - 3h}{4}.\end{aligned}$$

Thus, the slope of the line given by this method is

$$\frac{\bar{x}}{-\bar{y}} = \frac{mh}{2} \frac{4}{3h - m^2h} = \frac{2m}{3 - m^2}$$

and hence, the center of mass algorithm does not exactly reconstruct the line $y = mx$. We therefore conclude that this method is also at best first-order accurate.

2.3. The central difference algorithm

In this algorithm, one finds the slope $\tilde{m}$ of the approximate interface taking half the difference of the right and left hand column sums of the volume fractions. In other words, the slope $\tilde{m}$ of the approximate interface in the (i,j)th cell is given by

$$\tilde{m} = \frac{1}{2}\sum_{k=-1}^{1} f_{i+1,j+k} - f_{i-1,j+k}.$$

If we let

$$y_i = h\sum_{k=-1}^{1} f_{i,j+k},$$

then y_i can be thought of as being an approximation to the y coordinate of the interface at $x_i = (i+\frac{1}{2})h$. In the central difference algorithm we are determining the slope of the approximate interface by taking a centered difference of the discrete variable y_i.

In order to examine how well this method approximates an arbitrary line, we must consider the following two cases: (a) the line cuts opposite sides of the 3×3 grid, as in Fig. 3(a) or (b) it cuts adjacent sides, as in Fig. 3(b). First consider the case shown in Fig. 3(a). Suppose that the interface is given by $y(x) = mx + b$. Let A_1 be the sum of the volume fractions in the left hand column and A_3 the sum of the volume fractions in the right hand column in Fig. 3(a). We can determine A_1 by noting that it is the area of the trapezoid with sides of length b and $mh + b$ and width h while A_3 is the area of the trapezoid with sides of length $2mh + b$ and $3mh + b$ and width h,

$$A_1 = \frac{1}{2h^2}(mh + b + b)h = \frac{m}{2} + \frac{b}{h},$$
$$A_3 = \frac{1}{2h^2}(3mh + b + 2mh + b)h = \frac{5}{2}m + \frac{b}{h}.$$

Note that these formulas are exact no matter how the line $y(x)$ intersects a given cell, provided only that the line cuts opposite sides of the 3×3 grid. The central difference approximation to $m = y'(x)$ is given by

$$\tilde{m} = \frac{1}{2}\sum_{k=-1}^{1} f_{i+1,j+k} - f_{i-1,j+k} = \frac{A_3 - A_1}{2} = m.$$

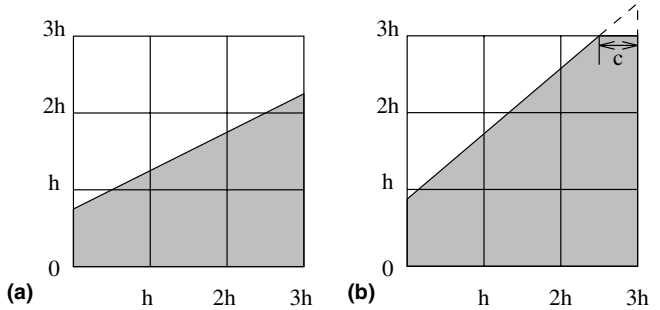

Fig. 3. (a) Center differences will exactly reconstruct a line that cuts opposite sides of a 3×3 block of cells. (b) It will not exactly reconstruct a line that cuts adjacent sides of a 3×3 block of cells.

Since the exact and approximate interfaces have the same slopes and the same volume fraction in the center cell, they are the same line. Thus, when the true interface is a line that intersects opposite sides of the 3×3 block, the approximate interface in the center cell will be precisely this line.

Now consider the case shown in Fig. 3(b), where c is the distance from the point where the line $y(x)$ intersects the top of the 3×3 block to the right-hand side of this block. Again, let A_1 be the sum of the volume fractions in the left hand column, and let A_3 be the sum of the volume fractions in the right hand column. The quantity A_1 is still the area of the trapezoid with sides of length b and $mh + b$, and width h. However, now A_3, is the area of the shape formed by subtracting the right triangle with the sides of length c and $3mh + b - 3h$ from the trapezoid with sides of length $2mh + b$ and $3mh + b$, and width h. Thus, in the second case we have

$$A_1 = \frac{1}{2h^2}(mh + b + b)h = m^2 + bh,$$
$$A_3 = \frac{1}{h^2}\left[\frac{1}{2}(3mh + b + 2mh + b)h - \frac{1}{2}(3mh + b - 3h)ch\right]$$
$$= \frac{5}{2}m + \frac{b}{h} - \frac{3}{2}cm - \frac{bc}{2h} + \frac{3}{2}c.$$

Again these formulas are exact no matter how the line intersects a given cell, provided only that the line $y(x)$ intersects adjacent sides of the 3×3 block. The central difference approximation to the slope m is thus,

$$\frac{A_3 - A_1}{2} = m - \frac{3}{4}cm - \frac{bc}{4h} + \frac{3}{4}c.$$

Thus, when the linear interface intersects adjacent sides of the 3×3 block, the approximate interface in the center cell will not be the original interface. Thus the central difference algorithm is not second-order accurate.

In practice one does not know that the true interface is a valid function of x (e.g., it could be a vertical line). We can address this problem by also determining an approximation $\tilde{m}^y$ to m by differencing the top and bottom rows and choosing the best value of $\tilde{m}$. One way to choose between the two values is to choose the smaller value

$$\tilde{m} = \min\{\tilde{m}^x, \tilde{m}^y\}.$$

This strategy will always return an exact approximation to a line that cuts opposite sides of the 3×3 block, even a vertical line. Another (better) strategy is discussed in Section 2.6.

We note that in their SOLA-VOF method Hirt and Nichols [52] used a centered difference of the volume fractions to determine a location on the interface for the purposes of specifying pressure boundary conditions. However they used an algorithm that is quite similar to SLIC to reconstruct the interface for the purposes of updating the volume fractions in time. Based on the computational tests described below, we believe that their volume fraction advection algorithm would have been more accurate if they had also used the central difference algorithm in the interface reconstruction phase of the volume fraction update.

2.4. Parker and Youngs' method

In this method, due to Parker and Youngs [70], one calculates an approximation to ∇f, which is taken to point in the direction normal to the approximate interface. One calculates ∇f with the following difference scheme

$$\frac{\partial f}{\partial x} = \frac{f_E - f_W}{2},$$
$$\frac{\partial f}{\partial y} = \frac{f_N - f_S}{2}.$$

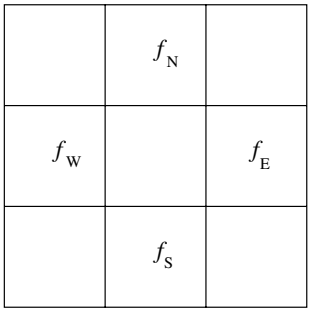

Fig. 4. The stencil that Parker and Youngs use to determine ∇f.

The variables f_E, f_W, f_N, f_S are centered in the cells as shown in Fig. 4 and are given by

$$f_E = \frac{1}{2+\alpha}(f_{i+1,j-1} + \alpha f_{i+1,j} + f_{i+1,j+1}),$$
$$f_W = \frac{1}{2+\alpha}(f_{i-1,j-1} + \alpha f_{i-1,j} + f_{i-1,j+1}),$$
$$f_N = \frac{1}{2+\alpha}(f_{i-1,j+1} + \alpha f_{i,j+1} + f_{i+1,j+1}),$$
$$f_S = \frac{1}{2+\alpha}(f_{i-1,j-1} + \alpha f_{i,j-1} + f_{i+1,j-1}),$$

where α is a free parameter. Parker and Youngs report that $\alpha = 2$ seems to give the best results.

In order to determine how well Parker and Youngs' method approximates straight lines, we consider the line $y = \frac{1}{3}x + h$ shown in Fig. 5(a). The volume fractions due to this line are shown in Fig. 5(b). The values of f_E, f_W, f_N, f_S are

$$f_E = \frac{1}{\alpha+2}\left(\frac{5\alpha}{6} + 1\right),$$
$$f_W = \frac{1}{\alpha+2}\left(\frac{1\alpha}{6} + 1\right),$$
$$f_N = 0,$$
$$f_S = \frac{1}{\alpha+2}(1 + \alpha + 1) = 1,$$

and hence

$$\frac{\partial f}{\partial x} = \frac{(f_E - f_W)}{2} = \frac{\alpha}{3(\alpha+2)},$$
$$\frac{\partial f}{\partial y} = \frac{(f_N - f_S)}{2} = -\frac{1}{2}.$$

The slope of the approximate interface is therefore

$$\tilde{m} = \frac{-\partial f/\partial x}{\partial f/\partial y} = \frac{2\alpha}{3(\alpha+2)}.$$

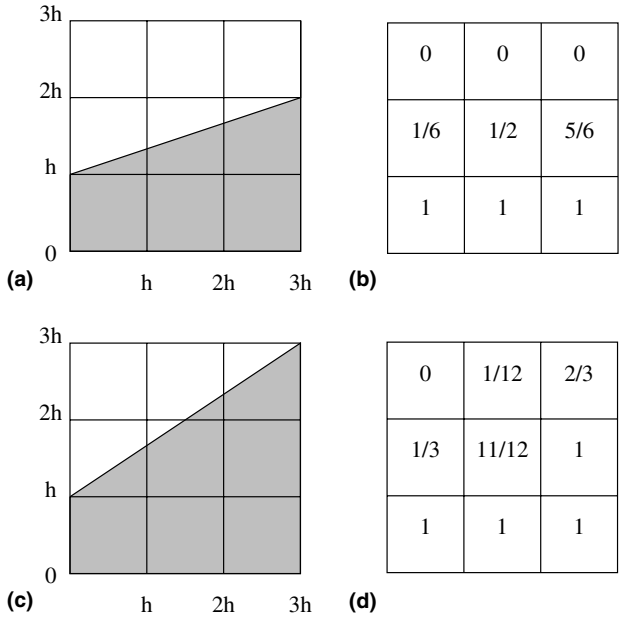

Fig. 5. (a) Parker and Youngs' method will reconstruct this line exactly only if $\alpha = 2$. (b) The volume fractions associated with the line shown in (a). (c) Parker and Youngs' method does not reconstruct this line exactly for $\alpha = 2$. Thus it does not reproduce all linear interfaces exactly, and so we conclude it is at best a first-order method. (d) The volume fractions associated with the line shown in (c).

The correct slope of the line is $m = \frac{1}{3}$. Thus if we wish to choose α so that

$$\frac{2\alpha}{3(\alpha+2)} = \frac{1}{3}.$$

We must have $\alpha = 2$. In other words, only the value of $\alpha = 2$ will yield the correct linear interface $y = \frac{1}{3}x + h$. We now show that choosing $\alpha = 2$ does not result in an algorithm that reconstructs all lines exactly. Consider the line $y = \frac{2}{3}x + h$ shown in Fig. 5(c). The volume fractions due to this line are shown in Fig. 5(d). When $\alpha = 2$ the values of f_E, f_W, f_N, f_S are

$$f_E = \frac{1}{4}\left(1 + 2 + \frac{2}{3}\right) = \frac{11}{12},$$
$$f_W = \frac{1}{4}\left(1 + \frac{2}{3} + 0\right) = \frac{5}{12},$$
$$f_N = \frac{1}{4}\left(0 + \frac{2}{12} + \frac{2}{3}\right) = \frac{5}{24},$$
$$f_S = \frac{1}{4}(1 + 2 + 1) = 1,$$

and hence

$$\frac{\partial f}{\partial x} = \frac{1}{2}\left(\frac{11}{12} - \frac{5}{12}\right) = \frac{1}{4},$$
$$\frac{\partial f}{\partial y} = \frac{1}{2}\left(\frac{5}{24} - 1\right) = \frac{-19}{48}.$$

The slope of the approximate interface is therefore

$$\tilde{m} = \frac{-\partial f/\partial x}{\partial f/\partial y} = \frac{1}{4}\frac{48}{19} = \frac{12}{19}. \tag{6}$$

Since the correct slope is $m = 2/3$, we conclude that Parker and Youngs' algorithm does not reconstruct all linear interfaces exactly.

Note that the quantity in (6) is independent of the grid width h. This implies that the approximation to the slope does not improve as $h \to 0$; i.e., in general this algorithm makes an $O(1)$ error in the slope of the interface. We therefore conclude that Parker and Youngs' algorithm is at best first-order accurate. This is consistent with the numerical results presented in Sections 3–5.

2.5. Least squares volume-of-fluid interface reconstruction algorithm

The LVIRA is due to Puckett [61] and has been used to model fluid interfaces in compressible [31,65–67] and incompressible [41,71] flows. Consider the 3×3 block of cells centered on the i,jth cell. Let $f(x)$ be a curve that passes through the i,jth cell and let $f_{k,l}$ for $k = i-1,\ldots,i+1$, $l = j-1,\ldots,j+1$ represent the volume fractions due to the function f, in the 3×3 block. Now let $\tilde{f}$ be a linear approximation to f with slope $\tilde{m}$ and volume fractions $\tilde{f}_{k,l}$ and assume that $\tilde{f}$ has the same volume fraction in the i,jth cell as f; i.e., $f_{i,j} = \tilde{f}_{i,j}$. Define $E^2_{i,j}$ to be the discrete L^2 error between the volume fractions in the 3×3 block of cells centered on the i,jth cell,

$$E^2_{i,j}(\tilde{m}) = \left(\sum_{k,l=-1}^{1} (\tilde{f}_{i+k,j+l}(\tilde{m}) - f_{i+k,j+l})^2 \right)^{\frac{1}{2}}. \tag{7}$$

In the LVIRA algorithm one minimizes $E^2_{i,j}$ as a function of $\tilde{m}$ by rotating the line $\tilde{f}$ under the constraint that this line exactly reproduces the volume fraction in the center cell, $\tilde{f}_{i,j} = f_{i,j}$. [3]

Note that basic design criterion in the LVIRA algorithm is to minimize some measure of the error between the volume fractions given by the true and approximate interfaces. One could instead choose to minimize the discrete L^∞ error

$$E^\infty_{i,j}(\tilde{m}) = \max_{k,l=-1,1} |\tilde{f}_{i+k,j+l}(\tilde{m}) - f_{i+k,j+l}|$$

or the discrete L^1 error

$$E^1_{i,j}(\tilde{m}) = \sum_{k,l=-1}^{1} |\tilde{f}_{i+k,j+l}(\tilde{m}) - f_{i+k,j+l}| \tag{8}$$

in the 3×3 block of cells centered on the i,jth cell, subject to the constraint that $\tilde{f}_{i,j} = f_{i,j}$.

We claim that if the original interface $f(x)$ is a line, then the LVIRA algorithm with any of the norms defined in (7) and (8) will exactly reconstruct the line in the i,jth cell. To see this suppose that $f(x)$ is a line and assume that that the minimization procedure will always find the correct global minimum when given volume fraction data $f_{k,l}$ due to a linear interface in a 3×3 block of cells. (Our test problems below demonstrate that this is a reasonable assumption.) Each of the norms $E^p_{i,j}$ in (7) and (8) has a minimum

[3] In order for (7) to represent our algorithm correctly one must allow $\tilde{m}$ to have the value $\tilde{m} = \infty$. For this reason it is better to express the error $E_{i,j}$ in (7) as a function of the unit normal $\vec{n}$ to the approximate interface. However, we use the formulas for the approximate slope $\tilde{m}$ instead of those for the unit normal vector $\vec{n}$, since we assume that they will be more familiar most readers. We hope that the use of $\tilde{m}$ in (7) this will not cause the reader confusion.

value of 0 that is attained when $f_{k,l} = \tilde{f}_{k,l}$ for each cell in the 3×3 block. This will only occur when $\tilde{f}(x) = f(x)$. Thus the LVIRA algorithm reconstructs linear interfaces exactly.

In the work presented below we determine the slope $\tilde{m}$ by using the central difference algorithm to obtain an initial guess and then using Brent's algorithm to minimize $E^2_{i,j}$. (Brent's algorithm is an iterative method that fits a parabola over the interval, and uses the minimum of the parabola as the next guess for the minimum of the given function. If it cannot fit a parabola, it does a golden section search. The method stops when both the interval and consecutive guesses are within a given tolerance. See [72] for further details.) To help ensure that Brent's method will converge to the global minimum, we slowly expand the interval about the initial guess until the error at the endpoints of the interval is greater than the one given by the initial guess. In Fig. 7 we present an example of Brent's method improving on the initial guess given by the central difference algorithm and finding the minimum of the function $E^2_{i,j}(\tilde{m})$ shown in Fig. 6.

We note that the term "least squares" that has come to be associated with this algorithm may be somewhat misleading. It was chosen because the method was originally designed to minimize the discrete L^2 error defined in (7). Since this is the same measure of error that is minimized in a "least squares" data fit it seemed natural to refer to the algorithm as the "least squares" volume-of-fluid interface reconstruction algorithm. However, one should note that given a *fixed* volume fraction $f_{i,j}$ in the center cell, the function $\mathscr{F}(\vec{n})$ that takes a unit vector $\vec{n}$ normal to the approximate linear interface $\tilde{f}$ in the center cell and returns the volume fractions $\tilde{f}_{i+k,j+l}$ for $k, l = -1, \ldots, 1$ in the 3×3 block of cells surrounding this cell, subject to the constraint that $\tilde{f}_{i,j} = f_{i,j}$, is nonlinear. Thus, unlike the least squares data fitting algorithm, the problem of minimizing (7) can not be formulated as the solution of a system of linear equations.

2.6. Efficient least squares VOF interface reconstruction algorithm

The ELVIRA is due to Pilliod [63] and has been used in thin flame models for tracking deflagrations and detonations [14,68], and to track the motion of fluid interfaces in variable density incompressible flow [21], and to track the motion of the boundary between a gas and a solid in shock physics applications [40].

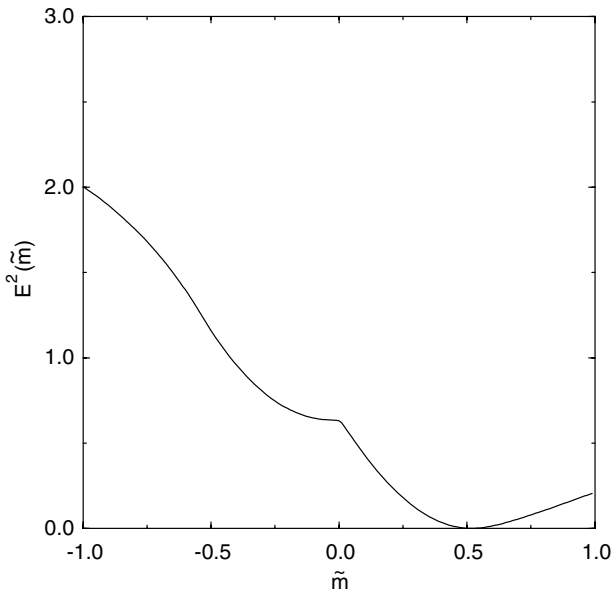

Fig. 6. The error $E^2_{i,j}(\tilde{m})$ between a circle that passes through the cell center, with a tangent line of slope of 0.5 at that point, and the approximate interface $\tilde{f}$ with slope $\tilde{m}$.

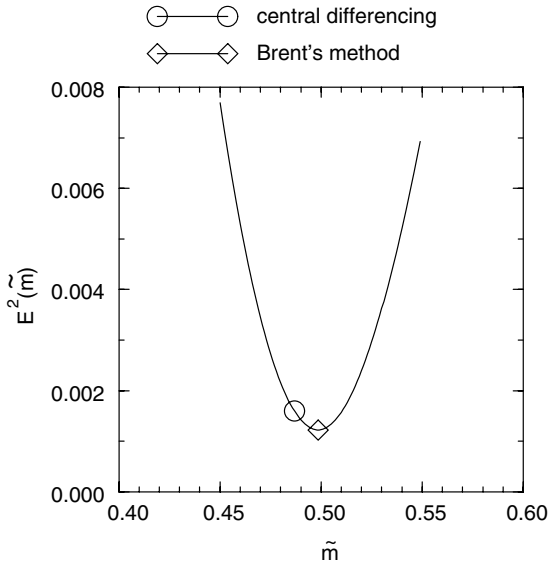

Fig. 7. We use the centered difference algorithm to obtain a starting point for Brent's method, which we then use to find the minimum of the curve shown in Fig. 6.

In the ELVIRA method, one obtains the slope $\tilde{m}$ of the approximate linear interface $\tilde{f}$ by choosing between six candidate values of $\tilde{m}$. The first three of these six candidate values are the backward, central and forward differences of the column sums of the volume fractions. In other words, we consider the following three values

$$\begin{aligned}
\tilde{m}^x_{\mathrm{b}} &= \sum_{l=-1}^{1} f_{i,j+l} - f_{i-1,j+l}, \\
\tilde{m}^x_{\mathrm{c}} &= \frac{1}{2}\sum_{l=-1}^{1} f_{i+1,j+l} - f_{i-1,j+l}, \\
\tilde{m}^x_{\mathrm{f}} &= \sum_{l=-1}^{1} f_{i+1,j+l} - f_{i,j+l}.
\end{aligned} \tag{9}$$

The other three candidate values are the backward, central, and forward differences of the column sums of the volume fractions *in the y-direction*; i.e., the differences of the row sums,

$$\begin{aligned}
\tilde{m}^y_{\mathrm{b}} &= \sum_{k=-1}^{1} f_{i+k,j} - f_{i+k,j-1}, \\
\tilde{m}^y_{\mathrm{c}} &= \frac{1}{2}\sum_{k=-1}^{1} f_{i+k,j+1} - f_{i+k,j-1}, \\
\tilde{m}^y_{\mathrm{f}} &= \sum_{k=-1}^{1} f_{i+k,j+1} - f_{i+k,j}.
\end{aligned} \tag{10}$$

(Note that the slopes $\tilde{m}^y$ are with respect to the coordinate system which is rotated 90° from the original coordinate system. The lines $\tilde{f}(\tilde{m}^y)$ associated with these slopes and the resulting errors $E^p_{i,j}(\tilde{m}^y)$ must be

interpreted appropriately.) To determine the best slope $\tilde{m}$ to use for a given collection of volume fractions $f_{k,l}$ in the 3×3 block we minimize one of the norms in (7) and (8) over the slopes in (9) and (10),

$$\tilde{m} = \min\{E^p_{i,j}(\tilde{m}^x_{\mathrm{b}}), \ldots, E^p_{i,j}(\tilde{m}^x_{\mathrm{f}}), E^p_{i,j}(\tilde{m}^y_{\mathrm{b}}), \ldots, E^p_{i,j}(\tilde{m}^y_{\mathrm{f}}), \tag{11}$$

where the superscript $p = 1, 2, \infty$ denotes one of the discrete norms in (7) and (8).

We claim that this method reconstructs all linear interfaces exactly. In Section 2.4 we saw that if the true interface is a line that intersects opposite sides of the 3×3 block, then a centered difference of the column sums exactly reproduces this line. In this case, (11) will return either $\tilde{m} = \tilde{m}^x_{\mathrm{c}}$ or $\tilde{m} = \tilde{m}^y_{\mathrm{c}}$, since one of these values must result in $E^p_{i,j}(\tilde{m}) = 0$.

Now suppose that the linear interface does not intersects opposite sides of the 3×3 block. Therefore, it must intersect adjacent sides of the 3×3 block. Consider the case shown in Fig. 3(b), where the interface is given by $y = mx + b$ with $m \leqslant 1$. Let A_1 be the sum of the volume fractions in the left hand column and A_2 be the sum of the volume fractions in the middle column. The quantities A_1 and A_2 are given by

$$A_1 = \frac{m}{2} + \frac{b}{h},$$

$$A_2 = \frac{1}{2h^2}(2mh + b + mh + b)h = \frac{3}{2}m + \frac{b}{h},$$

and their difference is,

$$\tilde{m}^x_{\mathrm{b}} = A_2 - A_1 = m.$$

Thus a backward difference of the column sums exactly reconstructs a linear interface that intersects the left hand side of the 3×3 block and has slope $m \leqslant 1$. By considering the mirror image of Fig. 3(b), one can see that a forward difference of the column sums will exactly reconstruct a linear interface that intersects the right hand side of the 3×3 block and has slope $m \geqslant -1$. If the magnitude of the slope m of the true linear interface is greater than one, then the argument above, applied to the 3×3 block in a coordinate frame that has been rotated 90°, shows that either a backward or forward difference of the row sums will produce the correct slope. Thus, at least one of the errors $E^p_{i,j}(\tilde{m}^\alpha_\beta)$ inside the square brackets in (11) will be 0, thereby guaranteeing that the ELVIRA algorithm will reconstruct all linear interfaces that pass through the center cell of the 3×3 block exactly.

If one only considers linear interfaces, the centered difference may seem redundant, since all three difference methods produce the correct slope when the linear interface intersects opposite sides of the 3×3 block. For nonlinear interfaces, however, it appears that a centered difference sometimes produces a more accurate approximation to the interface than the other two methods. For example, consider a circle that is placed in the 3×3 block such that the top of the circle intersects the center of the center cell. Then a backward difference results in a positive slope, a forward difference results in a negative slope, while a centered difference results in a zero slope. The latter is the best approximation to the slope of the tangent to the circle at the center of the center cell.

3. Stationary interface reconstruction

We now examine the accuracy of the reconstruction methods that were introduced in Section 2. All of the test problems in this section are stationary; no advection is performed and hence there is no error due to discretization in time.

3.1. Error measurement

The exact interface $\vec{z}$ separates the plane into two regions, which we refer to as the "dark" fluid and the "light" fluid. Let $f(x, y)$ be the characteristic function associated with the dark fluid as defined in (2). The

approximate interface also separates the plane into regions of dark and light fluid. Let $\tilde{f}(x,y)$ be the characteristic function associated with this partition of the plane. Then a natural measure of the error between the approximate and exact interfaces is

$$L^1 = \frac{1}{l}\int\int |f(x,y) - \tilde{f}(x,y)|\mathrm{d}x\,\mathrm{d}y, \tag{12}$$

where l is the length of the exact interface $\vec{z}$. In what follows, we evaluate (12) by finding the points where the true and approximate interfaces intersect each other and the cell edges, and then evaluating the integrals between these points analytically. We use the exact analytic equation instead of a numerical approximation to the integral in order to avoid truncation error and to minimize execution time.

In what follows we will sometimes average (12) over many (100 or 1000) computations with different initial data. This is done to avoid anomalously small values of L^1 that might arise from a fortuitous alignment of the interface with the grid. (For example, SLIC will reproduce a line exactly if the line is parallel to one of the axes.)

In all of the tables below we calculate a the *convergence rate* of each algorithm between successive grid spacings h. These convergence rates were calculated based on the assumption that the truncation error of each method is of the form of $C_1 h^q$, where q is the order of the method (i.e., the "Rate" and C_1 is a constant that is *independent* of both the grid size h and the time step Δt).

Please note that both the LVIRA and ELVIRA algorithms from Section 2 which we claim are second-order accurate sometimes exhibit rates of convergence between successive grid spacings h are 2.00 or above and sometimes lower than 2.00, but not lower than 1.9 (to two significant places). The fluctuations in the convergence rate (aka the order of the methods) shown in Table 1 below are typical of a second-order accurate method.

3.2. Test problems

Test problem 3.1. We begin by examining the accuracy with which each interface reconstruction algorithm approximates a line. The errors reported here were averaged over 1000 lines with randomly generated slopes and intercepts. It is apparent from the data presented in Table 1 that the errors produced by the SLIC, center of mass, Parker and Youngs' and centered difference algorithms decrease at a first-order rate. However, it should be noted that the amplitudes of these errors differ by more than two orders of magnitude, with SLIC consistently having the largest error for a given grid width h.

For all intents and purposes, the LVIRA and ELVIRA algorithms *reproduce the lines exactly*. The magnitude of the ELVIRA algorithm's error is machine zero, which here is $O(10^{-16})$, while the magnitude the LVIRA algorithm's error is determined by the tolerance of the root finding algorithm that one uses with it. Here that tolerance is 10^{-10} leading to an error that is $O(10^{-12})$. In particular, when we use a tolerance of 10^{-10} in Brent's method the error due to the LVIRA algorithm is $O(10^{-12})$. When we changed the tolerance

Table 1
The average L^1 error and convergence rate when approximating 1000 random lines

h	SLIC	Rate	CM	Rate	P&Y	Rate	CD	Rate	LVIRA	Rate	ELVIRA	Rate
$\frac{1}{2}$	1.3e+0		1.5e−1		1.3e−2		5.2e−3		4.2e−12		3.2e−17	
$\frac{1}{4}$	6.5e−1	1.0	7.6e−2	1.0	1.1e−2	0.3	2.8e−3	0.9	2.0e−12	–	2.1e−17	–
$\frac{1}{8}$	3.3e−1	1.0	3.7e−2	1.0	6.6e−3	0.8	1.6e−3	0.8	1.1e−12	–	1.9e−17	–
$\frac{1}{16}$	1.6e−1	1.0	1.9e−2	1.0	3.6e−3	0.8	7.2e−4	1.2	5.2e−13	–	9.8e−17	–
$\frac{1}{32}$	8.2e−2	1.0	9.2e−3	1.0	1.8e−3	1.0	4.1e−4	0.8	2.8e−13	–	2.6e−16	–
$\frac{1}{64}$	4.1e−2	1.0	4.8e−3	0.9	9.9e−4	0.9	1.8e−4	1.2	1.4e−13	–	7.3e−17	–

to 10^{-14}, the error reduced to $O(10^{-16})$, which is machine zero. *In other words, one can set the tolerance of the root finder in the LVIRA algorithm so as to achieve an error that is machine zero.*

Remark. In Test problem 3.1 the LVIRA and ELVIRA algorithms reproduce the linear interfaces *exactly*; i.e., up to a given tolerance which is machine zero for the ELVIRA algorithm and is $O(10^{-12})$ for the LVIRA algorithm. This is the reason that no convergence rates are reported for these two algorithms in Table 1. Please also note that – as we discuss in the paragraph immediately above – one can also arrange for the error due to the LVIRA algorithm to be machine zero by setting the tolerance of the root finding algorithm (i.e., Brent's method) to 10^{-14}.

Test problem 3.2. Next we examine the accuracy with which each interface reconstruction algorithm approximates a circle. The data presented in Table 2 is the error L^1 defined in (12) averaged over 1000 unit circles with randomly generated centers. It is apparent from this data that the convergence rate of the SLIC algorithm is *precisely* first-order rate with respect to h, while the convergence rate of the center of mass algorithm oscillates about 1.0 on the coarser grids ($h = 1/2$ and $1/4$) and then is exactly 1.0 on the finer grids. The convergence rate for Parker and Youngs' method exhibits a similar behavior, but has a better overall decrease; i.e., a larger convergence rate on the coarser grids. However note that the convergence rate for Parker and Youngs' method asymptotes to 1.0 as $h \to 0$. (In this regard, please read our remark below.)

The convergence rate of the LVIRA algorithm is *precisely* 2.0 for all values of h, while the convergence rate of the ELVIRA algorithm oscillates about 2.0. This indicates that both the LVIRA and ELVIRA algorithms are second-order accurate on this test problem. Finally we note that the convergence rate of the centered difference algorithm is 2.0 on the coarse grids, but begins to decrease ($2.0 \to 1.9 \to 1.8$) as the grid is refined. This is evidence that the centered difference algorithm is *not* second-order. See the remark in the next paragraph and the discussion accompanying Test problem 3.3 for details of our reasoning.

Remark. We caution the reader against drawing conclusions concerning an algorithm's accuracy from the error and convergence rate data on grids that are too coarse; i.e., h too large, such as $h = 1/2$, $1/4$ and $1/8$ in Table 2. The behavior of an algorithm on an *under-resolved* grid is not a good indicator of its behavior in the limit as the grid spacing $h \to 0$, since such grids can give rise to anomalously large or small errors and hence affect the quality of the convergence rate data. For example, the error due to Parker and Youngs' method in Table 6 actually *increases* as h goes from $\frac{1}{2}$ to $\frac{1}{4}$ yielding a negative convergence rate. This is *not* a flaw in the algorithm, but rather a symptom of computing on an under-resolved grid. The errors due to Parker and Youngs' method in Tables 2 and 6 are an *excellent* example of the effect of *under-resolution* on a numerical method. In order to avoid confusion that may occur in interpreting our results on under-resolved grids, we usually only present the results of computations for which $h \geqslant \frac{1}{8}$. However, we do display some data from computations on coarser grids, because we think that it is a valuable illustration of the effects of computing on an under-resolved grid.

Table 2
The average L^1 error and convergence rate when approximating 1000 random circles

h	SLIC	Rate	CM	Rate	P&Y	Rate	CD	Rate	LVIRA	Rate	ELVIRA	Rate
$\frac{1}{2}$	5.8e−3		6.7e−3		8.4e−3		7.8e−3		1.1e−2		9.2e−3	
$\frac{1}{4}$	2.8e−3	1.0	3.2e−3	1.1	2.1e−3	2.0	2.0e−3	2.0	2.6e−3	2.0	2.0e−3	2.2
$\frac{1}{8}$	1.4e−3	1.0	1.8e−3	0.8	6.6e−4	1.7	5.1e−4	2.0	6.6e−4	2.0	5.2e−4	2.0
$\frac{1}{16}$	6.8e−4	1.0	9.9e−4	0.9	2.6e−4	1.3	1.4e−4	1.9	1.7e−4	2.0	1.4e−4	1.9
$\frac{1}{32}$	3.4e−4	1.0	5.1e−4	1.0	1.2e−4	1.2	3.9e−5	1.8	4.3e−5	2.0	3.6e−5	1.9
$\frac{1}{64}$	1.7e−4	1.0	2.5e−4	1.0	5.6e−5	1.1	1.1e−5	1.8	1.1e−5	2.0	9.0e−6	2.0

It is surprising that a centered difference exhibits nearly second-order accuracy when we use it to reconstruct a circle, yet it is clearly not second-order accurate when we use it to reconstruct a line. We believe that this is because the data in Table 2 is an average measure of the error and that, for a given circle, the centered difference algorithm returns a second-order accurate approximation to a tangent to the circle in a large proportion of the cells that contain a portion of the circle. Our reasoning is as follows. As we noted in Section 2.4 a centered difference of the column sums of the volume fractions will reconstruct a line exactly if the line passes through the opposite sides of the 3×3 grid. It is likely that if the interface is circular, then in a large number of cases there will be at least one second-order accurate linear approximation to the interface (e.g., a tangent) that passes through opposite sides of the 3×3 block and that the centered difference algorithm may return a second order approximation to this line. However, if this is only true for most, but not all cells that are occupied by the circular interface, then there must be some cells in which the centered difference algorithm is only returning a first-order approximation to the interface.

Test problem 3.3. We tested this conjecture by examining the *pointwise* error produced by each algorithm, rather than an integral norm of the error such as that defined in (12). For our purposes here we define the error in the discrete L^∞ or "sup" norm by

$$L^\infty = \max_{i,j}\{L^\infty_{i,j}\}, \tag{13}$$

where the maximum is only taken over those cells (i, j) that contain a portion of the interface and

$$L^\infty_{i,j} = \max_{(x,y)\in\Omega_{i,j}} |f(x,y) - \tilde{f}(x,y)|, \tag{14}$$

where $\Omega_{i,j}$ denotes the i,jth cell.

In the work presented here we evaluate the error $L^\infty_{i,j}$ in a given cell by determining the maximum distance between the two curves f and $\tilde{f}$ analytically and comparing these values with those at the cell edges.

Note that the definition in (13) and (14) is reasonable only when the true and approximate interfaces occupy precisely the same cells. This precludes us from using it in later sections when we study the accuracy of volume-of-fluid advection algorithms.

In Table 3 we present the error L^∞ when we used each method to reconstruct one randomly generated circle. It is apparent that in this norm SLIC is $O(\sqrt{h})$, LVIRA and ELVIRA are second-order accurate and centered difference and the other algorithms are first-order accurate. This data supports our conjecture concerning the near second-order accurate behavior of the central difference algorithm in the averaged L^1 norm when it is used to reconstruct a circular interface. Namely that all of the algorithms *except* the LVIRA and ELVIRA produce errors that are first-order in the grid spacing h in some of the cells that contain the interface, whereas the LVIRA and ELVIRA algorithms produce errors that are second-order in grid spacing h in *all* such cells.

Test problem 3.4. Next we study the accuracy of these methods when we use them to reconstruct a continuous interface that has several discontinuities in its first derivative. In the first test problem the exact interface is a cross formed by removing four unit squares from the corners of a square three units on a side. In Table 4 we present the results of using the various interface reconstruction methods to reconstruct this shape. The data presented in Table 4 is the error L^1 averaged over 1000 crosses with randomly generated centers and orientations.

It is apparent from Table 4 that the SLIC algorithm exhibits precisely first-order accuracy, the center of mass algorithm exhibits somewhat better than first order accuracy for the range of h tested, Parker and Youngs' algorithm reduces the error by a factor of 500 over the same range. The centered difference, LVIRA and ELVIRA algorithms exhibit precisely second-order accuracy to two significant digits.

Table 3
Maximum pointwise error L^∞ and convergence rate when approximating a circle

h	SLIC	Rate	CM	Rate	P&Y	Rate	CD	Rate	LVIRA	Rate	ELVIRA	Rate
$\frac{1}{8}$	3.1e−1		5.6e−3		5.8e−3		2.1e−2		5.8e−3		5.8e−3	
$\frac{1}{16}$	2.3e−1	0.4	2.6e−3	1.0	2.3e−3	1.3	9.2e−3	1.1	1.4e−3	2.0	1.4e−3	2.0
$\frac{1}{32}$	1.6e−1	0.5	1.3e−3	1.0	1.1e−3	1.0	4.3e−3	1.1	3.5e−4	2.0	3.5e−4	2.0
$\frac{1}{64}$	1.1e−1	0.5	6.4e−4	1.0	5.6e−4	1.0	2.0e−3	1.0	8.6e−5	2.0	8.6e−5	2.0

Table 4
The average L^1 error and rate when approximating 1000 random crosses

h	SLIC	Rate	CM	Rate	P&Y	Rate	CD	Rate	LVIRA	Rate	ELVIRA	Rate
$\frac{1}{8}$	1.6e−2		5.6e−3		4.5e−3		4.4e−3		4.4e−3		4.2e−3	
$\frac{1}{16}$	7.9e−3	1.0	1.9e−3	1.5	1.2e−3	1.7	1.1e−3	2.0	1.1e−3	2.0	1.1e−3	2.0
$\frac{1}{32}$	3.9e−3	1.0	7.5e−4	1.4	3.7e−4	1.7	2.9e−4	1.9	2.8e−4	2.0	2.7e−4	2.0
$\frac{1}{64}$	1.9e−3	1.0	3.1e−4	1.3	1.2e−4	1.7	7.3e−5	2.0	6.8e−5	2.0	6.6e−5	2.0

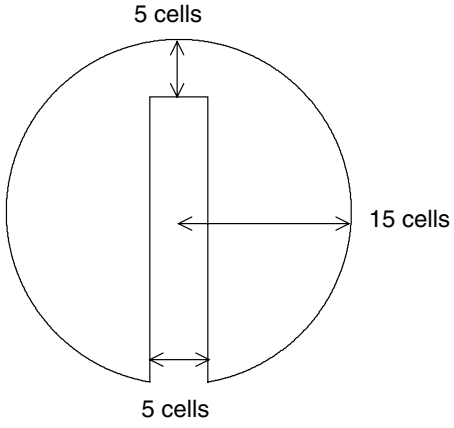

Fig. 8. The notched circle, first introduced by Zalesak [73] to study the accuracy of advection algorithms, which we use in several of our test problems.

Table 5
The average L^1 error and rate when approximating 1000 random notched circles

h	SLIC	Rate	CM	Rate	P&Y	Rate	CD	Rate	LVIRA	Rate	ELVIRA	Rate
$\frac{1}{8}$	1.2e−2		3.0e−3		2.6e−3		2.6e−3		2.5e−3		2.6e−3	
$\frac{1}{16}$	5.3e−3	1.2	9.1e−4	1.7	6.9e−4	1.9	6.5e−4	2.0	6.3e−4	2.0	6.6e−4	2.0
$\frac{1}{32}$	2.7e−3	1.0	3.0e−4	1.6	1.9e−4	1.9	1.7e−4	2.0	1.6e−4	2.0	1.7e−4	2.0
$\frac{1}{64}$	1.2e−3	1.2	1.1e−4	1.5	5.4e−5	1.8	4.1e−5	2.0	3.9e−5	2.0	4.0e−5	2.0

Test problem 3.5. Finally we study the problem of reconstructing the shape shown in Fig. 8. This shape is produced by cutting a rectangle 1/3 units in width from a unit circle, starting 1/3 of a unit length from the top of the circle. This shape was introduced by Zalesak [73] to study the accuracy of flux-corrected

transport (FCT) on a simple advection problem. It has subsequently been used by many other authors to study the accuracy of various advection algorithms [42,44].

In Table 5 we show the average L^1 error that we obtained after approximating 1000 of these shapes with randomly generated centers and orientations. We see that the SLIC algorithm exhibits roughly first-order accuracy, the center of mass algorithm exhibits slightly better than first-order accuracy, Paker and Youngs' algorithm exhibits slightly less than second-order accuracy and the enter difference, LVIRA and ELVIRA algorithms exhibit clear second-order accuracy for this problem. [4]

4. Volume-of-fluid advection algorithms

In order to approximate solutions of the advection equation (5) we need an algorithm for evolving the volume fractions in time. Let $u^n_{i-\frac{1}{2},j}$ (resp. $v^n_{i,j-\frac{1}{2}}$) denote the value of u (resp. v) at the center of the left (resp. bottom) edge of the i,jth cell and suppose that these velocities satisfy a discrete form of (4),

$$\frac{(u^n_{i+\frac{1}{2},j} - u^n_{i-\frac{1}{2},j})}{\Delta x} + \frac{(v^n_{i,j+\frac{1}{2}} - v^n_{i,j-\frac{1}{2}})}{\Delta y} \approx 0.$$

Given an approximation to the interface in each cell for which $0 < f^n_{i,j} < 1$ we wish to determine the volume fractions $f^{n+1}_{i,j}$ at the new time $t^{n+1} = (n+1)\Delta t$. We refer to algorithms for doing this as *volume-of-fluid advection algorithms*.

In this article we study two types of advection algorithms. Both are based on the standard conservative finite difference update of (5),

$$f^{n+1}_{i,j} = f^n_{i,j} + \frac{\Delta t}{\Delta x}[F^n_{i-\frac{1}{2},j} - F^n_{i+\frac{1}{2},j}] + \frac{\Delta t}{\Delta y}[G^n_{i,j-\frac{1}{2}} - G^n_{i,j+\frac{1}{2}}], \tag{15}$$

where $F^n_{i-\frac{1}{2},j} = (fu)^n_{i-\frac{1}{2},j}$ denotes the flux of f across the left-hand edge of the i,jth cell and $G^n_{i,j-\frac{1}{2}} = (fv)^n_{i,j-\frac{1}{2}}$ denotes the flux across the bottom edge of the i,jth cell, etc.

4.1. Operator split advection

The simplest advection algorithm for approximating solutions of (5) is the *fractional step* or *operator split* method,

$$f^*_{i,j} = f^n_{i,j} + \frac{\Delta t}{\Delta x}[F^n_{i-\frac{1}{2},j} - F^n_{i+\frac{1}{2},j}], \tag{16}$$

$$f^{n+1}_{i,j} = f^*_{i,j} + \frac{\Delta t}{\Delta y}[G^*_{i,j-\frac{1}{2}} - G^*_{i,j+\frac{1}{2}}]. \tag{17}$$

where the superscript * represents an intermediate value for the volume fractions and fluxes. There is a simple geometric interpretation of the fluxes in (16) and (17). Suppose that $u^n_{i+\frac{1}{2},j}$ is positive. Divide the (i,j)th cell into two disjoint rectangles, with areas $u^n_{i+\frac{1}{2},j}\Delta t\,\Delta y$ on the right and $(\Delta x - u^n_{i+\frac{1}{2},j}\Delta t)\Delta y$ on the left as shown in Fig. 9(a).

[4] We caution the reader against drawing conclusions concerning an algorithm's accuracy from the convergence rate data on the coarser grids, namely $h = 1/2$, $1/4$ and $1/8$. The behavior of an algorithm on an under resolved grid is not a good indicator of its behavior in the limit as the grid spacing $h \to 0$.

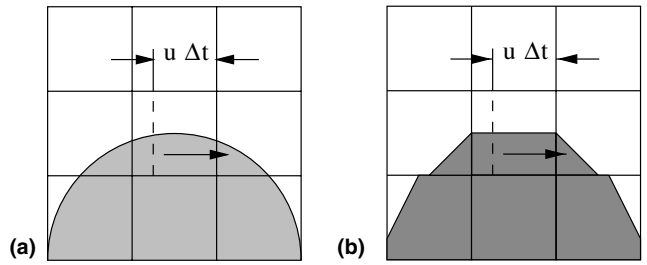

Fig. 9. (a) In operator split advection, the fluid to the right of the dotted line crosses the right cell edge. (b) In a volume-of-fluid method, we use the reconstructed interface to determine the amount of fluid that crosses each edge.

All of the fluid to the right of the dotted line in Fig. 9(a) will cross the right-hand edge during this time step. In particular, the flux of dark fluid across this edge is equal to the amount of dark fluid contained in this rectangle. In a volume-of-fluid method, this can be determined by the location of the reconstructed interface as shown in Fig. 9(b). Thus, if $V_{i+\frac{1}{2},j}$ denotes the volume of dark fluid in the center cell to the right of the dotted line in Fig. 9(b), then the (approximate) volume fraction flux across the right hand cell edge is given by

$$F^n_{i+\frac{1}{2},j} = u^n_{i+\frac{1}{2},j} V_{i+\frac{1}{2},j} / (u^n_{i+\frac{1}{2},j} \Delta t \, \Delta y) = V_{i+\frac{1}{2},j} / (\Delta t \, \Delta y). \tag{18}$$

After using (18) in (16) to determine the intermediate volume fractions $f^*_{i,j}$, one then uses these values to reconstruct the interface in all cells that satisfy $0 < f^*_{i,j} < 1$. The vertical fluxes $G^*_{i,j+1/2}$ are then determined by a geometric construction analogous to the one described for the horizontal fluxes, and the volume fractions at the new time level $f^{n+1}_{i,j}$ are found by inserting these vertical fluxes into (17). This procedure can be made second-order accurate simply by alternating the sweep direction at each time step.

4.1.1. The CFL constraint

It is apparent from geometric considerations that one must choose the CFL number σ so that the amount of fluid that leaves a cell in one time step is no more than the amount of fluid that was originally in the cell. In other words, one must choose σ so that

$$V_{i+\frac{1}{2},j} - V_{i-\frac{1}{2},j} \leqslant f_{i,j} \Delta x \, \Delta y \tag{19}$$

for all i, j. One way to ensure that (19) is always satisfied is to choose $\sigma \in (0, 1]$ so that

$$|u^n_{i+\frac{1}{2},j}| \Delta t \leqslant \Delta x/2 \quad \text{and} \quad |v^n_{i,j+\frac{1}{2}}| \Delta t \leqslant \Delta y/2 \qquad \text{for all } i, j. \tag{20}$$

An alternative, is to choose $\sigma \in (0, 1]$ so that

$$\left(u^n_{i+\frac{1}{2},j} - u^n_{i-\frac{1}{2},j}\right) \Delta t \leqslant \Delta x \quad \text{and} \quad \left(v^n_{i,j+\frac{1}{2}} - v^n_{i,j-\frac{1}{2}}\right) \Delta t \leqslant \Delta y.$$

This latter condition is less restrictive than (20) and will usually result in a larger time step

4.2. Test problems

Test problem 4.1. We begin by studying the accuracy with which second-order operator splitting combined with each of the interface reconstruction methods approximates a line that is translating in a constant velocity field. We obtained the errors reported in Table 6 by translating 100 randomly generated lines with

Table 6
The average L^1 error after translating 100 randomly generated lines one time unit

h	SLIC	Rate	CM	Rate	P&Y	rate	CD	Rate	LVIRA	Rate	ELVIRA	Rate
$\frac{1}{2}$	9.6e−1		1.5e−1		6.7e−3		3.4e−3		4.0e−12		6.7e−17	
$\frac{1}{4}$	5.1e−1	0.9	7.9e−2	0.9	9.9e−3	−0.6	2.2e−3	0.6	1.3e−12	–	6.8e−17	–
$\frac{1}{8}$	2.7e−1	1.0	3.6e−2	1.1	6.4e−3	0.6	1.5e−3	0.6	1.1e−12	–	1.1e−16	–
$\frac{1}{16}$	1.2e−1	1.1	1.8e−2	1.0	3.3e−3	0.9	4.0e−4	1.9	4.0e−13	–	9.0e−17	–
$\frac{1}{32}$	6.6e−2	0.9	9.5e−3	1.0	1.9e−3	0.9	3.1e−4	0.4	2.1e−13	–	4.2e−17	–
$\frac{1}{64}$	3.0e−2	1.1	4.4e−3	1.1	8.4e−4	1.1	1.1e−4	1.5	8.2e−14	–	7.9e−17	–

unit velocity in a randomly generated direction for one unit of time and averaging the error L^1 between the approximate and exact solutions. It is apparent from the data in Table 6 that the SLIC, center of mass, centered difference, and Parker and Youngs' algorithms are all first-order accurate.

As with a stationary line, the LVIRA and ELVIRA methods essentially reproduce the interface exactly. The error produced by the LVIRA method is entirely due to the tolerance we used in Brent's algorithm. The ELVIRA method is accurate to machine zero. One can use this as a design criterion for constructing a formally second-order accurate interface tracking algorithm. Namely, require that it must propagate a straight line with any slope in a uniform velocity field in any direction, *exactly*.

Remark. Please note that in Test problem 4.1 the LVIRA and ELVIRA algorithms translate lines *exactly*; i.e., up to a given tolerance which is machine zero for the ELVIRA algorithm and is $O(10^{-12})$ for the LVIRA algorithm. This is the reason that no convergence rates are reported for these two algorithms in Table 6. Please also note that – as we discuss in the paragraph immediately preceding Table 1 – one can also arrange for the error due to the LVIRA algorithm to be machine zero by setting the tolerance of the root finding algorithm (e.g., Brent's method) to 10^{-14}.

Test problem 4.2. Next we present three tests with circles. In the first test we translate a unit circle in the x-direction with unit velocity for one unit of time using various CFL numbers σ. In Table 7 we present the errors when we use $\sigma = 1$, while in Table 8 we present the errors when we use $\sigma = 1/32$. Note that errors due to the SLIC, center of mass, Parker and Youngs algorithms decrease at a first-order rate for both values of σ, while the errors due to the LVIRA and ELVIRA algorithms decrease at a second-order rate for $\sigma = 1$.

Table 7
The L^1 error after translating a circle one unit in time with CFL number $\sigma = 1$

h	SLIC	Rate	CM	Rate	P&Y	Rate	CD	Rate	LVIRA	Rate	ELVIRA	Rate
$\frac{1}{8}$	1.8e−2		4.7e−3		1.5e−3		6.1e−4		1.0e−3		6.1e−4	
$\frac{1}{16}$	9.1e−3	0.9	2.6e−3	0.9	8.7e−4	0.8	1.5e−4	2.0	2.8e−4	1.8	1.6e−4	1.9
$\frac{1}{32}$	4.7e−3	1.0	1.4e−3	0.9	4.6e−4	0.9	4.2e−5	1.9	6.6e−5	2.1	4.0e−5	2.0
$\frac{1}{64}$	2.4e−3	1.0	7.5e−4	0.9	2.3e−4	1.0	1.4e−5	1.6	1.6e−5	2.0	1.0e−5	2.0

Table 8
The L^1 error after translating a unit circle one unit in time with $\sigma = 1/32$

h	SLIC	Rate	CM	Rate	P&Y	Rate	CD	Rate	LVIRA	Rate	ELVIRA	Rate
$\frac{1}{8}$	1.8e−2		6.0e−3		2.4e−3		1.4e−3		1.5e−3		1.4e−3	
$\frac{1}{16}$	9.1e−3	0.9	3.0e−3	1.0	1.3e−3	0.9	4.3e−4	1.7	5.0e−4	1.5	4.4e−4	1.7
$\frac{1}{32}$	4.7e−2	1.0	1.6e−3	1.0	6.6e−4	1.0	1.3e−4	1.7	1.5e−4	1.7	1.3e−4	1.8
$\frac{1}{64}$	2.4e−3	1.0	8.0e−4	1.0	3.2e−4	1.0	4.2e−5	1.6	4.8e−5	1.7	3.8e−5	1.7

Also note that it is apparent from the data presented in these two tables that, in general, decreasing the CFL number did not reduce the error. In fact, the amplitude of the error is generally larger when $\sigma = 1/32$ than when $\sigma = 1$. Furthermore, for the two second-order accurate algorithms LVIRA and ELVIRA, as well as for the centered difference method, the convergence rate is adversely affected by the very small CFL number. The increase in the amplitude of the error seen in Table 8 is almost certainly due to the accumulation of local truncation error over 32 times as many time steps, and we believe that this is also the cause of the degradation in the convergence rate of the second-order accurate algorithms.

Unless noted otherwise, we set $\sigma = 0.5$ in all of the remaining test problems.

Test problem 4.3. Next we translate 100 unit circles with randomly generated centers in a randomly generated direction with unit velocity for one unit of time. In Table 9 we present the L^1 error averaged over 100 randomly chosen circles. It is apparent that the center of mass, SLIC, Parker and Youngs' and centered difference methods are first-order accurate, and the LVIRA and ELVIRA methods are second-order accurate.

Test problem 4.4. In our final test with circles we place a unit circle with its center at cell center and rotate it with unit angular velocity for one rotation. Here we used a CFL number of $\sigma = \pi/6$. It is apparent from the data presented in Table 10 that SLIC, the center of mass and Parker and Youngs' algorithms exhibit first-order accuracy while the other three algorithms exhibit second-order accuracy. Starting with the left-hand column and moving right, the overall decrease in the error for each algorithm when the grid was reduced from $h = 1/2$ to $1/64$ was 20, 47, 185, 827, 893 and 1146, respectively. A precisely second-order accurate decrease in the error would be by a factor of 1024.

Test problem 4.5. In the next collection of test problems we study the accuracy of the operator split advection algorithm when we use it to model the motion of a cross rotating and translating in a uniform incompressible velocity field. We average the error over 100 crosses with randomly generated centers and randomly generated orientations. In the first test problem we translate each cross in a randomly generated direction with unit velocity for one unit time. It is apparent from the data presented in Table 11 that none of the algorithms reduce the error at a second-order accurate rate. This is to be expected, since the cross has

Table 9
The average L^1 error after translating 100 random unit circles in random directions

h	SLIC	Rate	CM	Rate	P&Y	Rate	CD	Rate	LVIRA	Rate	ELVIRA	Rate
$\frac{1}{8}$	3.1e−2		9.0e−3		1.9e−3		9.6e−4		1.5e−3		1.0e−3	
$\frac{1}{16}$	1.8e−2	0.8	4.8e−3	0.9	1.1e−3	0.9	2.5e−4	2.0	4.1e−4	1.9	2.5e−4	2.0
$\frac{1}{32}$	1.1e−2	0.8	2.4e−3	1.0	5.8e−4	0.9	1.0e−4	1.2	1.1e−4	1.9	6.6e−5	1.9
$\frac{1}{64}$	6.3e−3	0.7	1.2e−3	1.1	2.8e−4	1.1	5.8e−5	0.8	2.7e−5	2.0	2.0e−5	1.7

Table 10
The L^1 error after rotating a circle once with the operator split advection algorithm

h	SLIC	Rate	CM	Rate	P&Y	Rate	CD	Rate	LVIRA	Rate	ELVIRA	Rate
$\frac{1}{8}$	3.0e−2		2.9e−3		1.0e−3		7.8e−4		8.6e−4		7.7e−4	
$\frac{1}{16}$	1.9e−2	0.7	1.1e−3	1.4	4.0e−4	1.3	2.9e−4	1.4	2.5e−4	1.8	2.3e−4	1.7
$\frac{1}{32}$	8.8e−3	1.1	6.2e−4	0.9	2.0e−4	1.0	5.5e−5	2.4	5.9e−5	2.1	5.7e−5	2.0
$\frac{1}{64}$	4.6e−3	0.9	3.1e−4	1.0	7.8e−5	1.3	1.8e−5	1.6	1.5e−5	2.0	1.3e−5	2.1

Table 11
The average L^1 error after translating 100 random crosses in random directions

h	SLIC	Rate	CM	Rate	P&Y	Rate	CD	Rate	LVIRA	Rate	ELVIRA	Rate
$\frac{1}{8}$	2.8e−2		1.1e−2		9.4e−3		9.5e−3		9.5e−3		9.4e−3	
$\frac{1}{16}$	1.5e−2	1.0	4.1e−3	1.4	3.1e−3	1.6	3.0e−3	1.7	2.9e−3	1.7	3.0e−3	1.7
$\frac{1}{32}$	8.1e−3	0.8	1.7e−3	1.3	1.0e−3	1.6	9.3e−4	1.7	9.1e−4	1.7	9.2e−4	1.7
$\frac{1}{64}$	4.9e−3	0.7	7.8e−4	1.1	3.4e−4	1.5	3.0e−4	1.6	3.1e−4	1.6	2.9e−4	1.6

discontinuities in its' first derivative at the corners, and hence a typical analysis of the error (e.g., by Taylor series expansion) would fail at these points. However, it is also apparent, both from the data presented in Table 11 that the convergence rates for the centered difference, LVIRA and ELVIRA methods are somewhat better than those for Parker and Youngs' method, the center of mass method and especially the SLIC method.

Test problem 4.6. Next we take a cross, centered on a cell center, with its sides initially parallel to the grid, and rotate it with unit angular velocity for one rotation. It is apparent from the data in Table 12 that all of the methods are first-order accurate. The reduction in the accuracy of the LVIRA and ELVIRA methods to first-order is presumably due to the discontinuities in the first derivative of the interface at the corners. This conjecture is consistent with all of the data presented in this paper.

In Fig. 10 we compare the shape of a cross after it has been rotated using each of the various interface reconstruction methods and compare it with the exact solution in Fig. 8. In this example the grid width was $h = 1/64$. Notice that the approximate and true solutions differ the most at the corners, where the derivative of the function that describes the interface is discontinuous. This is consistent with our conjecture above concerning why the second-order accurate algorithms LVIRA and ELVIRA are only first order accurate on these problems.

Note also note the degree to which the SLIC computation has broken up the interface; i.e., it is no longer even remotely akin to a continuous function. This behavior is commonly seen in computations with volume-of-fluid methods that are based on the SLIC [47] and so-called "VOF" [52] interface reconstruction algorithms. In fact, this artifact is so common that users of these methods have a name for it: "flotsam", and have devised various ad-hoc methods for reducing its occurrence [69, p. 138]. It is apparent from Figs. 10 and 11 that the flotsam problem is completely eliminated when SLIC is replaced by any one of the piecewise linear interface reconstruction algorithms. We also note that Chorin [74] has developed improvements to SLIC that reduce the amount of fluid trailing behind the true interface.

Test problem 4.7. Finally we tested the second-order accurate operator split advection method with the various interface reconstruction methods on Zalezak's test problem. Here we revolved the shape shown in Fig. 8 about a point 5/3 units below its center for one revolution. It is apparent from the data in Table 13 that all of the methods exhibit an $O(h)$ decrease in the error.

Table 12
The L^1 error after rotating a cross counterclockwise one revolution

h	SLIC	Rate	CM	Rate	P&Y	Rate	CD	Rate	LVIRA	Rate	ELVIRA	Rate
$\frac{1}{8}$	4.8e−2		3.9e−2		3.3e−2		3.3e−2		3.4e−2		3.3e−2	
$\frac{1}{16}$	3.3e−2	0.5	1.8e−2	1.1	1.6e−2	1.0	1.6e−2	1.0	1.6e−2	1.1	1.6e−2	1.0
$\frac{1}{32}$	2.5e−2	0.4	1.0e−2	0.8	8.5e−3	0.9	8.5e−3	0.9	8.5e−3	0.9	8.5e−3	0.9
$\frac{1}{64}$	1.5e−2	0.7	6.4e−3	0.7	5.3e−3	0.7	5.3e−3	0.7	5.3e−3	0.7	5.3e−3	0.7

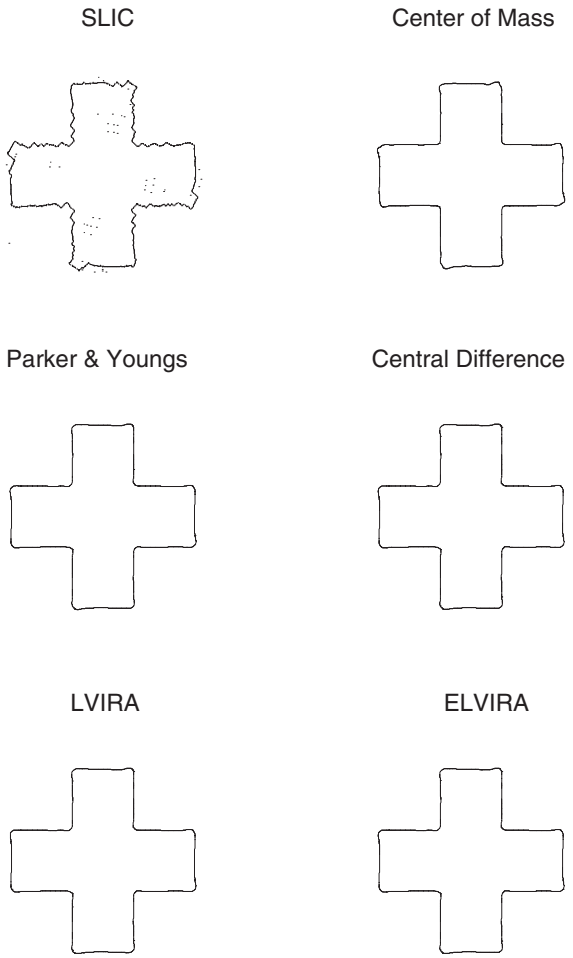

Fig. 10. A cross that has been rotated one revolution using the operator split advection algorithm and the various interface reconstruction methods. Notice that only the SLIC algorithm produces flotsam.

In Table 14 we present the difference between the volume (area) of the initial shape and the final shape for the computations presented in Table 13. It is apparent that all of the methods conserve the volume (or equivalently the mass) of the shape to better than 1.1e−14 which is nearly machine zero.

In Fig. 11 we present the results of using the operator split advection method and the various interface reconstruction methods on Zalesak's test problem on a grid with $h = 1/15$. This grid size was chosen to facilitate direct comparison with other published results of the same test problem [42,73]. Note that, as expected, the error is greatest at the corners. We believe this is caused by the discontinuity in the first derivative of the function that describes the interface at these points.

5. Unsplit advection

For many problems one will obtain satisfactory results with the second-order accurate, fractional step method described in Section 4. However for some problems, such as unstable displacements in porous

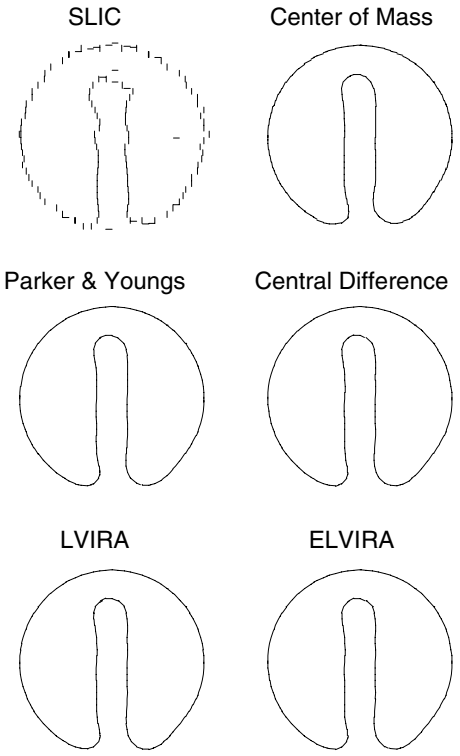

Fig. 11. The result of using the operator split advection algorithm and the various interface reconstruction methods on Zalesak's test problem. Again notice that only the SLIC algorithm produces flotsam.

Table 13
The average L^1 error after translating 100 random notched circles in random directions

h	SLIC	Rate	CM	Rate	P&Y	Rate	CD	Rate	LVIRA	Rate	ELVIRA	Rate
$\frac{1}{8}$	4.8e−2		1.9e−2		1.6e−2		1.7e−2		1.7e−2		1.6e−2	
$\frac{1}{16}$	2.5e−2	0.9	7.2e−3	1.4	6.2e−3	1.4	6.3e−3	1.4	6.4e−3	1.4	6.2e−3	1.4
$\frac{1}{32}$	1.4e−2	0.9	4.1e−3	0.8	2.9e−3	1.1	2.8e−3	1.2	2.9e−3	1.1	2.8e−3	1.2
$\frac{1}{64}$	6.9e−3	1.0	2.3e−3	0.8	1.4e−3	1.1	1.3e−3	1.1	1.3e−3	1.1	1.3e−3	1.1

Table 14
The difference between the intial volume and final volume of the shape in Fig. 8

h	SLIC	C of M	P&Y	C Diff	LVIRA	ELVIRA
$\frac{1}{4}$	0.0e+0	0.0e+0	4.4e−16	8.9e−16	8.9e−16	4.4e−16
$\frac{1}{8}$	1.3e−15	2.7e−15	3.1e−15	3.6e−15	2.7e−15	3.6e−15
$\frac{1}{16}$	−3.1e−15	−2.7e−15	−1.3e−15	−2.7e−15	−1.3e−15	−4.4e−16
$\frac{1}{32}$	−3.1e−15	−4.9e−15	−4.0e−15	−4.0e−15	−2.7e−15	−4.0e−15
$\frac{1}{64}$	7.1e−15	4.4e−15	3.1e−15	8.9e−16	1.1e−14	4.0e−15

media, fractional step methods can distort the interface (e.g., see the discussion in [42]). A characteristic feature of this problem is the so-called "push-pull" or "staircase" phenomenon. For problems such as these it is preferable to use an unsplit advection algorithm. In this section we present an unsplit, volume-of-fluid advection algorithm that is based on the approach used by Bell et al. [42] to develop a second-order accurate, unsplit, finite difference method for approximating solutions of scalar hyperbolic conservation laws. We then present the results of applying this advection algorithm to the test problems studied in Section 4. Since SLIC is an inherently one-dimensional method, we will not use it in this section.

5.1. A first-order accurate unsplit advection algorithm

In order to present the basic idea behind the unsplit algorithm we begin by describing a first-order accurate version. We wish to use a conservative finite difference method of the form (15) to approximate solutions of the conservation law (5). To illustrate our approach we assume that $u > 0$ and $v > 0$, and describe how one determines the flux $F^n_{i+1/2,j}$. The other cases are analogous.

The flux through the right-hand edge of the (i,j)th cell in the time interval (t^n, t^{n+1}) is

$$F^n_{i+1/2,j} = \int_{t^n}^{t^{n+1}} \int_{y_{i,j-1/2}}^{y_{i,j+1/2}} u(x_{i+1/2}, y, t) f(x_{i+1/2,j}, y, t)\,\mathrm{d}y\,\mathrm{d}t = u_{i+1/2,j} \int_{t^n}^{t^{n+1}} \int_{y_{i,j-1/2}}^{y_{i,j+1/2}} f(x_{i+1/2,j}, y, t)\,\mathrm{d}y\,\mathrm{d}t \tag{21}$$

where we have assumed in our numerical discretization that $u_{i+1/2,j}$ is constant on the space time interval $(y_{i,j-1/2}, y_{i,j+1/2}) \times (t^n, t^{n+1})$. The integral in (21) is the amount of dark fluid in the space-time rectangle $BCEF$ shown in Fig. 12. We can find this amount by tracing back along the characteristics that originate from the rectangle $BCEF$. This gives us the solid region $ABCEFGH$ shown in Fig. 12. The domain of dependence of these characteristics at time t^n is the shaded region $AHBEG$ in Fig. 13. Note that this region is the rectangle $ABDE$, plus the triangle ABH, minus the triangle DEG,

$$F^n_{i+1/2,j} \approx \int\int_{ABDE} f\,\mathrm{d}x\,\mathrm{d}y + \int\int_{ABH} f\,\mathrm{d}x\,\mathrm{d}y - \int\int_{DEG} f\,\mathrm{d}x\,\mathrm{d}y. \tag{22}$$

In order to approximate the right hand side of (22) we use one of the volume-of-fluid interface reconstruction algorithms described in Section 2 to determine an approximation to the interface in cell (i,j) and cell $(i,j-1)$. We then compute the area of the intersection of the dark fluid with the rectangle $ABDE$ and the triangles DEG and ABH. This yields an approximation to each of the terms on the right hand side of (22) and hence an approximation to $F^n_{i+1/2,j}$.

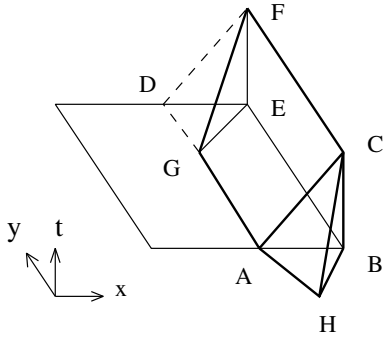

Fig. 12. The fluid inside the object outlined in bold passes through the right cell edge EB between time t^n and t^{n+1}.

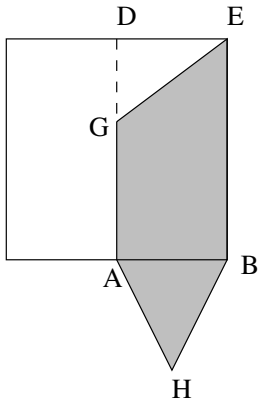

Fig. 13. The domain of dependence for characteristics that pass through the righthand edge EB of the (i,j)th cell.

The fluxes through the other three edges of the cell are found in an analogous manner. This method for calculating the flux, which Colella [43] calls corner transport upwind (CTU) is first-order accurate. See [44] for a detailed analysis of the accuracy of this method and [42] and [44] for the results of tests when this method is implemented as a finite difference algorithm for linear advection problems.

5.2. A second-order accurate unsplit advection algorithm

We now describe a second-order accurate, unsplit, volume-of-fluid advection algorithm. Our approach is based on the finite difference method for scalar conservation laws in multiple dimensions originally described in [42]. We approximate the flux $F^n_{i+1/2,j}$ in (21) by integrating (5) over the prism $ABCDEF$ shown in Fig. 12 and integrating by parts. We begin by writing (5) in the form

$$f_t + u_x f + u f_x + (vf)_y = 0,$$

and setting $u = u_{i+1/2,j}$ and $u_x = (u_x)_{i,j}$. Integrating over $ABCDEF$ yields

$$\int\int\int_{ABCDEF} (f_t + (u_x)_{i,j} f + u_{i+1/2,j} f_x + (vf)_y)\,\mathrm{d}x\,\mathrm{d}y\,\mathrm{d}t = 0.$$

Integrating the above expression by parts, and noting that $u_{i+1/2,j}$ is constant, we find that the flux $F^n_{i+1/2,j}$ is given by

$$F^n_{i+1/2,j} = \int\int_{ABDE} f\,\mathrm{d}x\,\mathrm{d}y + \int\int_{ABC} vf\,\mathrm{d}x\,\mathrm{d}t - \int\int_{DEF} vf\,\mathrm{d}x\,\mathrm{d}t + \int\int\int_{ABCDEF} (u_x)_{i,j} f\,\mathrm{d}x\,\mathrm{d}y\,\mathrm{d}t. \tag{23}$$

The integral over $ABDE$ is the volume of dark fluid in this rectangle. As above we use an interface reconstruction algorithm to determine an approximation to the interface in the (i,j)th cell and use this approximation to compute the area of the intersection of the dark fluid with rectangle $ABDE$ to compute this quantity.

Now let R_1 be the ratio of the volume of dark fluid in $ABDE$ to the area of $ABDE$, and let V_1 be the volume of the prism $ABCDEF$. We approximate the volume integral in (23) by

$$\int\int\int_{ABCDEF} (u_x)_{i,j} f\,\mathrm{d}x\,\mathrm{d}y\,\mathrm{d}t \approx R_1 V_1 (u_x)_{i,j}.$$

In order to evaluate the integral over *DEF* in Eq. (23) we integrate (5) in the quasilinear form

$$f_t + u_x f + u f_x + v_y f + v f_y = 0$$

over the tetrahedron *DEFG*

$$\int\int\int_{DEFG} (f_t + (u_x)_{i,j} f + u_{i+1/2,j} f_x + (v_y)_{i,j}) f + v_{i,j+1/2} f_y \,\mathrm{d}x\,\mathrm{d}y\,\mathrm{d}t = 0. \tag{24}$$

The domain of dependence of *DEF* is the triangle *DEG*. Integrating the expression on the left hand side of (24) we find that the tetrahedron *DEFG* is related to the triangle *DEF* through

$$\int\int_{DEF} f \,\mathrm{d}x\,\mathrm{d}t = \int\int_{DEG} f \,\mathrm{d}x\,\mathrm{d}y + \int\int\int_{DEFG} ((u_x)_{i,j} + (v_y)_{i,j}) f \,\mathrm{d}x\,\mathrm{d}y\,\mathrm{d}t. \tag{25}$$

The integral over *DEG* is the volume of dark fluid in the triangle *DEG*. We approximate this quantity by using an interface reconstruction algorithm to determine an approximation to the interface in the (i,j)th cell and then computing the area of the intersection of the dark fluid with triangle *DEG*.

To obtain an approximation to the volume integral on the right hand side of (25), we begin by letting R_2 be the ratio of the volume of dark fluid in *DEG* to the area of *DEG*, and let V_2 be the volume of the tetrahedron *DEFG*. Our approximation to the volume integral in (25) is then

$$\int\int\int_{DEFG} \Big((u_x)_{i,j} + (v_y)_{i,j}\Big) f \,\mathrm{d}x\,\mathrm{d}y\,\mathrm{d}t = R_2 V_2 \Big((u_x)_{i,j} + (v_y)_{i,j}\Big).$$

We evaluate integral over *ABC* in a similar manner. Thus we are able to evaluate each term of (23), and hence determine the flux of dark fluid through the right hand edge *BEFC* of the (i,j)th cell.

Note that if $v_{i,j+1/2} < 0$, then the point *G* will lie in the $(i, j+1)$st cell. Thus the tetrahedron *DEFG* will lie in the $(i, j+1)$st cell instead of the (i,j)th cell. In this case we add the tetrahedron *DEFG* to the prism *ABCDEF*, instead of subtracting it as we did above. Hence we add the integral over *DEF* instead of subtracting it. In order to avoid the distorted region that arises when $u_{i+1/2,j}$ and $u_{i+1/2,j+1}$ are of opposite sign we determine the x-coordinate of the vertex *G* by

$$x = \min\left(\mathrm{i}\Delta x + \frac{\Delta x}{2}, \mathrm{i}\Delta x + \frac{\Delta x}{2} - \Delta t u_{i+1/2,j+1}\right).$$

In this way we are assured that *G* lies in the $(i, j+1)$st cell.

5.3. Test problems

In this section we use the second-order accurate unsplit advection algorithm just described to compute most of the test problems presented in Section 4.

Test problem 5.1. We begin with the translation of a smooth interface, the unit circle. We take 100 unit circles with randomly generated centers, translate each circle with unit velocity in a randomly generated direction and average the L^1 error in approximating each circle. It is apparent from the data shown in Table 15 that the errors associated with the center of mass and Parker and Youngs' algorithms decrease at a rate that is not quite $\mathrm{O}(h)$. The overall decrease is by a factor of 5.8 and 12.9 respectively, whereas a precisely first-order accurate method would have diminished the error by a factor of 16. [5] On the other hand the

[5] Again however, we caution the reader against drawing conclusions concerning an algorithm's accuracy from the convergence rate data on under-resolved grids (e.g., $h = 1/4$ and 1/8) in Table 15.

Table 15
The average L^1 error and rate after translating 100 random circles in random directions

h	CM	Rate	P&Y	Rate	CD	Rate	LVIRA	Rate	ELVIRA	Rate
$\frac{1}{4}$	1.1e−2		4.0e−2		1.8e−2		2.1e−2		1.9e−2	
$\frac{1}{8}$	1.0e−2	0.0	2.0e−2	1.0	3.8e−3	2.3	4.4e−3	2.2	3.9e−3	2.3
$\frac{1}{16}$	6.8e−3	0.6	1.1e−2	0.9	7.4e−4	2.4	1.1e−3	2.1	7.7e−4	2.3
$\frac{1}{32}$	3.6e−3	0.9	5.6e−3	0.9	1.8e−4	2.0	2.8e−4	1.9	1.8e−4	2.1
$\frac{1}{64}$	1.8e−3	1.0	3.1e−3	0.9	6.7e−5	1.4	7.5e−5	1.9	5.0e−5	1.9

errors associated with the centered difference, LVIRA and ELVIRA algorithms decrease at a rate that is somewhat better than $O(h^2)$. The overall decrease in the error being 274.6, 276 and 376, respectively, whereas a precisely second-order accurate method would have diminished the error by a factor of 256. (But in this regard, please see the footnote concerning drawing conclusions about an algorithm's accuracy from the convergence rate data on under-resolved grids.)

In summary, the center of mass and Parker and Youngs' algorithms appear to be first-order accurate, while the other three appear to be second-order accurate, although the centered difference algorithm shows some evidence that its convergence rate may asymptote to 1.0 as $h \to 0$. As in Section 4.2 we conjecture that the apparent second-order accurate behavior of the centered difference algorithm is again due to the fact that *on average* it will produce a second-order approximation to a tangent to the circle in each cell. It is important to note that this will *not* be the case when the interface is more nearly linear, in which case the centered difference algorithm is on average first-order accurate as shown in Tables 1 and 3.

Test problem 5.2. In the next test problem we rotate a unit circle, centered on a cell center, with unit angular velocity for ten rotations. It is apparent from the data in Table 16 that the rate of decrease of the errors associated with each algorithm is comparable with the rate of decrease seen in the previous test problem. In this problem however, the center of mass and Parker and Youngs' algorithms exhibit a somewhat better than $O(h)$ decrease in the error. However, the same conclusions continue to apply.

In Fig. 14, we present five unit circles that have been rotated ten times with the various interface reconstruction methods on a grid with $h = 1/32$ and compare the results with the true solution. At this level of graphical resolution all of the approximate solutions are indistinguishable from the true solution. In fact, even when magnified by a factor of 64, the approximate interfaces still appear to be continuous – although not smooth – in spite of the fact that they are actually composed of a collection of discontinuous line segments.

Table 16
The L^1 error after rotating a circle 10 times with the unsplit advection algorithm

h	CM	Rate	P&Y	Rate	CD	Rate	LVIRA	Rate	ELVIRA	Rate
$\frac{1}{2}$	1.6e−2		1.4e−2		1.4e−2		1.4e−2		1.4e−2	
$\frac{1}{4}$	5.3e−3	1.6	3.0e−3	2.2	3.3e−3	2.1	3.9e−3	1.9	3.3e−3	2.1
$\frac{1}{8}$	3.1e−3	0.8	9.7e−4	1.6	1.7e−3	0.9	9.8e−4	2.0	1.7e−3	0.9
$\frac{1}{16}$	1.4e−3	1.2	2.9e−4	1.7	7.1e−4	1.3	2.4e−4	2.0	7.2e−4	1.3
$\frac{1}{32}$	6.3e−4	1.1	1.4e−4	1.0	1.5e−4	2.2	5.9e−5	2.0	1.5e−4	2.3
$\frac{1}{64}$	3.0e−4	1.1	6.8e−5	1.1	1.7e−5	3.2	1.5e−5	2.0	1.6e−5	3.2

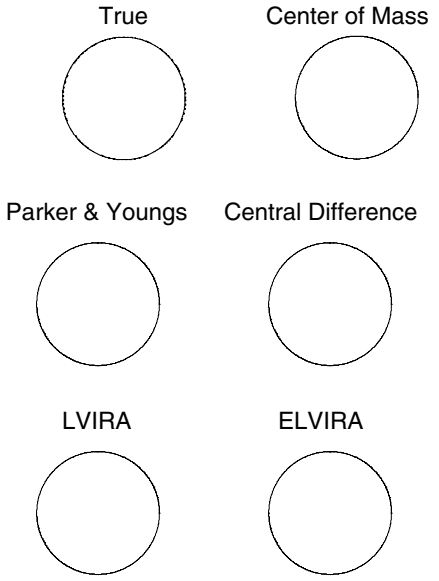

Fig. 14. A unit circle that has been rotated for ten revolutions with the unsplit advection algorithm and various reconstruction methods.

Test problem 5.3. In our next test we repeat the linear advection test problem with crosses described in Section 4.2. As before, we averaged the error obtained after advecting 100 crosses with randomly generated centers with unit velocity for one unit in time in a randomly generated direction. As one can see from Table 17, all of the interface reconstruction algorithms produce comparable errors, which decay at a rate that is somewhat better than first-order, but is certainly not second-order. This is presumably due to the lack of smoothness in the interface shape. Starting with the left-hand column and ending with the right, the overall decrease in the error is by a factor of 70, 108, 120, 123 and 124, respectively. The tendency of the center of mass algorithm (or SLIC in Sections 3 and 4) to produce the smallest overall decrease in the error and for the ELVIRA algorithm to produce the largest overall decrease in the error, even when all of the algorithms are performing at a nominally first-order accurate rate, was consistently displayed in all of the test problems we studied.

Test problem 5.4. Our conjecture that all of the interface reconstruction algorithms are first-order accurate when one uses them to advect a non-smooth interface is confirmed by our next test. In this case we take *one* cross centered on a grid point and rotate it once with unit angular velocity. It is clear from the data in Table 18 that all of the algorithms produce comparable errors and that these errors are decreasing at a rate which appears to asymptote to 1.0 as $h \to 0$.

Table 17
The average L^1 error after translating 100 random crosses in random directions

h	CM	Rate	P&Y	Rate	CD	Rate	LVIRA	Rate	ELVIRA	Rate
$\frac{1}{2}$	1.0e−1		9.5e−2		9.8e−2		9.9e−2		9.9e−2	
$\frac{1}{4}$	2.8e−2	1.8	2.7e−2	1.8	2.7e−2	1.9	2.7e−2	1.9	2.7e−2	1.9
$\frac{1}{8}$	9.7e−3	1.5	8.5e−3	1.7	8.6e−3	1.7	8.5e−3	1.7	8.5e−3	1.7
$\frac{1}{16}$	3.6e−3	1.4	2.7e−3	1.6	2.6e−3	1.7	2.7e−3	1.7	2.6e−3	1.7
$\frac{1}{32}$	1.4e−3	1.3	8.8e−4	1.6	8.1e−4	1.7	8.0e−4	1.7	8.0e−4	1.7

Table 18
The L^1 error after rotating a cross one revolution with the unsplit advection algorithm

h	CM	Rate	P&Y	Rate	CD	Rate	LVIRA	Rate	ELVIRA	Rate
$\frac{1}{2}$	1.7e−1		1.7e−1		1.7e−1		1.7e−1		1.7e−1	
$\frac{1}{4}$	6.8e−2	1.4	6.3e−2	1.4	6.3e−2	1.4	6.3e−2	1.4	6.0e−2	1.4
$\frac{1}{8}$	2.6e−2	1.4	2.5e−2	1.4	2.5e−2	1.3	2.6e−2	1.3	2.5e−2	1.3
$\frac{1}{16}$	1.0e−2	1.3	1.0e−2	1.3	1.1e−2	1.3	1.1e−2	1.3	1.0e−2	1.3
$\frac{1}{32}$	3.9e−3	0.9	3.8e−3	1.4	4.0e−3	1.4	3.9e−3	1.5	4.0e−3	1.4
$\frac{1}{64}$	1.8e−3	0.8	1.4e−3	1.4	1.5e−3	1.5	1.5e−3	1.4	1.5e−3	1.5

In Fig. 15, we present the approximate interfaces from Test problem 5.4 with grid width $h = 1/64$ and compare the results with the exact solution. If one compares images produced with the same interface reconstruction in Figs. 10 and 15, it is apparent that there is a discernable improvement in the resolution of the corners of the cross in Fig. 15. We conjecture that this improved resolution is due to an increase in the accuracy with which the unsplit method resolves a portion of the overall error, such as the phase error associated with finite difference solutions of the advection Eq. (3) [75, Chapter 1]. In particular, based on the results presented here, we conjecture that for both advection methods, the order of this portion of the error is as high or higher than that of the underlying advection algorithm, and that the phase error associated with the unsplit advection algorithm is one or more degrees higher than that associated with the

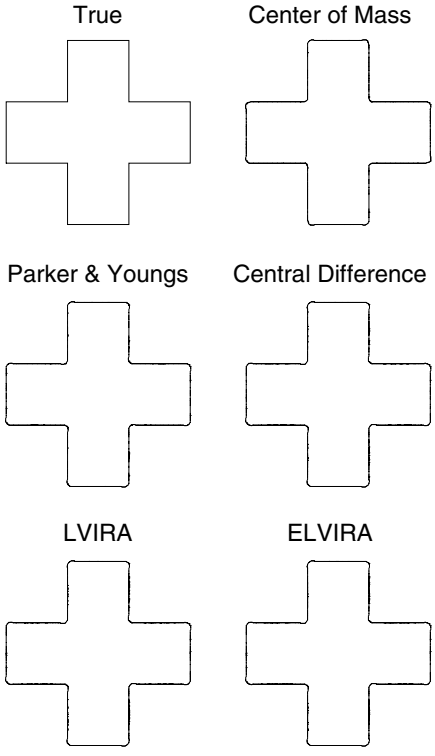

Fig. 15. A cross that has been rotated one revolution with the unsplit advection algorithm and various reconstruction methods. By comparing the results shown here with those shown in Fig. 10, one can see that the unsplit advection algorithm produces noticeably better resolution of the corners.

operator split advection algorithm. Thus, although both the fractional step and unsplit methods are second-order accurate, the unsplit method produces better overall results.

Test problem 5.5. Finally we test the unsplit advection method on Zalezak's test problem. As one can see from the data in Table 19 all of the algorithms – with the possible exception of the center of mass algorithm – produce comparable errors and that these errors are decreasing at a rate which appears to asymptote to 1.0 as $h \to 0$.

In Fig. 16 we present the results of computing Zalesak's test problem on a grid with $h = 1/15$. Again, we chose this relatively coarse grid in order to facilitate a direct comparison with other published results of the same problem such as in [42] and [73]. The coarseness of the grid prevents one from detecting the increased resolution at the corners we expect with the unsplit advection algorithm. Higher resolution computations of

Table 19
The average L^1 error for Zalesak's test problem

h	CM	Rate	P&Y	Rate	CD	Rate	LVIRA	Rate	ELVIRA	Rate
$\frac{1}{8}$	1.8e−2		1.6e−2		1.6e−2		1.6e−2		1.6e−2	
$\frac{1}{16}$	7.0e−3	1.3	5.8e−3	1.5	5.9e−3	1.4	5.8e−3	1.5	5.7e−3	1.5
$\frac{1}{32}$	3.7e−3	0.9	2.6e−3	1.1	2.7e−3	1.2	2.7e−3	1.1	2.6e−3	1.1
$\frac{1}{64}$	2.2e−3	0.8	1.3e−3	1.1	1.2e−3	1.1	1.3e−3	1.1	1.2e−3	1.1

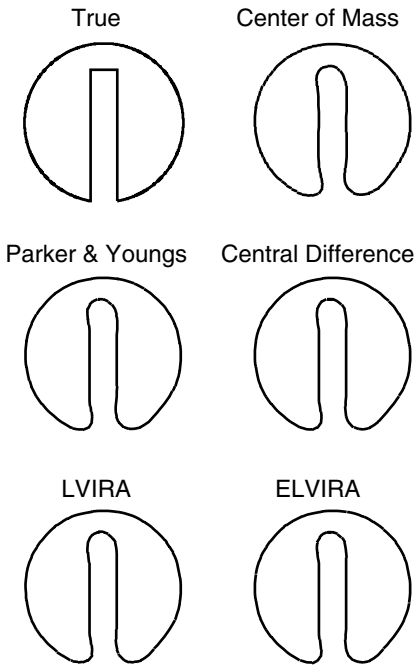

Fig. 16. Here we present the results of using the unsplit advection algorithm and the various reconstruction methods to compute Zalesak's test problem. Note that the computations shown here and in Fig. 11 were conducted on the very coarse grid shown in Fig. 8. Consequently one cannot detect the increased resolution at the corners expected from the unsplit advection algorithm. Higher resolution computations of this problem do exhibit better resolution with unsplit algorithm.

Table 20
The difference between the intial volume and final volume of the shapes in Fig. 16

h	C of M	P&Y	Center difference	LVIRA	ELVIRA
$\frac{1}{4}$	0.0e+0	2.2e−15	−4.4e−16	−8.8e−16	4.4e−16
$\frac{1}{8}$	2.7e−15	5.3e−15	6.2e−15	3.6e−15	3.1e−15
$\frac{1}{16}$	8.9e−16	2.2e−15	−1.8e−15	−1.8e−15	−6.7e−15
$\frac{1}{32}$	−1.9e−14	1.6e−14	−1.0e−14	−2.2e−15	−1.1e−14
$\frac{1}{64}$	7.2e−14	4.9e−14	3.3e−14	3.3e−14	1.0e−14

this problem do exhibit better resolution at the corners of Zalesak's shape with unsplit algorithm. We conclude that, although both the fractional step and unsplit methods are second-order accurate, the unsplit method produces better overall results.

In Table 20 we present the difference between the volume (area) of the initial shape and the final shape for the computations presented in Table 19. It is apparent that all of the methods conserve the volume (and hence the mass) of the shape to better than one part in 10^{13}, which is nearly machine zero.

6. Concluding remarks

We have presented a comprehensive framework for the design and implementation of modern volume-of-fluid interface tracking algorithms and conducted an extensive computational study of the accuracy of several commonly used versions of these algorithms. Our presentation is based on separating the interface reconstruction phase from the advection or time update phase of the overall tracking algorithm and studying the accuracy of the interface reconstruction algorithm independently of the advection algorithm. In our study of volume-of-fluid interface reconstruction algorithms, we have identified several key properties – or design criteria – that we believe will ensure that the method is second-order accurate on smooth interfaces; e.g., interfaces that have two or more continuous derivatives. In particular, we have found that if a volume-of-fluid interface reconstruction algorithm is designed so that it always reproduces lines (or planes in 3D) exactly, then it will be second-order accurate on smooth interfaces – in both the L^1 and the L^∞ norms – when used to reconstruct stationary interfaces. We have introduced two new volume-of-fluid interface reconstruction algorithms that have this property and demonstrated that they consistently exhibit second-order accuracy when we use them to reconstruct smooth stationary interfaces, whereas the other algorithms we tested overall exhibit first-order accuracy.

In our study of volume-of-fluid advection algorithms we have demonstrated that one can obtain second-order accuracy (in space and time) by combining one of our second-order accurate interface reconstruction algorithms with a standard fractional step or operator split solution of the time evolution equation and alternating the sweep directions at each time step (i.e., Strang splitting). We have also introduced a new unsplit volume-of-fluid advection algorithm that is second-order accurate in space and time when combined with one of the second-order accurate interface reconstruction algorithms. Furthermore we have shown that the unsplit algorithm exhibits noticeably better resolution of the interface near discontinuities in the derivatives of the interface (e.g., corners). Since this improved resolution does not manifest as an increase in the order of accuracy of the advection algorithm, we conjecture that it is a higher order effect due to an increase in the accuracy with which the algorithm resolves a portion of the error, such as the phase error (e.g., see the discussion on phase errors in [75]).

Another conclusion that can be drawn from our study is that piecewise linear interface reconstruction algorithms that reconstruct lines exactly will revert to first-order accuracy when the interface fails to be sufficiently smooth (e.g., remains continuous but has discontinuities in the first-derivative). We conjecture

that the constraint that a volume-of-fluid interface reconstruction method must *always* reproduce the correct fluid volume in each cell is sufficient to guarantee first-order accuracy in the L^1 norm for time dependent advection problems – at least when the advection algorithm is formally second-order accurate as is the case in our studies. This conclusion appears to be true even for reconstruction algorithms that do not exhibit second-order accuracy on smooth interfaces, such as SLIC. However it is apparent from the results presented in Table 3. that something more than this constraint is needed in order to guarantee first-order accuracy in the L^∞ norm.

In summary, we have presented two new volume-of-fluid interface reconstruction algorithms and demonstrated that they are more accurate than the most commonly used volume-of-fluid interface reconstruction algorithms. These new interface reconstruction algorithms are currently being used in a number of application codes for modeling the motion of material interfaces in compressible gas dynamics [65,66], high-pressure solids in the hydrostatic limit [31,39,67], combined gas dynamics/solid mechanics shock physics [40] and variable density incompressible fluid flow [21,41,71]. We have also introduced a new, unsplit volume-of-fluid advection algorithm, demonstrated that it is second-order accurate in space and time and shown that it exhibits superior resolution of kinks or corners in the interface as compared to the fractional step advection algorithm, which is currently the most widely used advection algorithm. This unsplit advection algorithm has been used by Puckett et al. [41] to model the motion of an interface between two immiscible fluids *without surface tension* moving in a two dimensional velocity field that satisfies the incompressible Euler equations. We refer the reader to the papers by Sussman and Puckett [21] and Sussman [22] for studies concerning the accuracy and convergence rate of these methods when they are used to model the motion of an interface between two immiscible fluids *with surface tension* moving in a velocity field that satisfies either the incompressible Euler or Navier–Stokes equations. The work in [21] and [22] also includes computations in both radially symmetric coordinates and three space dimensions.

Acknowledgements

The authors would like to thank John Bell for many helpful discussions and especially for his suggestions concerning unsplit advection algorithms. The second author (EGP) would also like to acknowledge Bill Noh's role introducing him to these problems and Phil Colella's role in encouraging him to pursue them.

References

[1] I.D. Aleinov, E.G. Puckett, M.M. Sussman, Formation of droplets in microscale jetting devices, in: Proceedings of the Third ASME/JSME Joint Fluids Engineering Conference, American Society of Mechanical Engineers, San Francisco, CA, 1999, fEDSM99-7106.

[2] N.V. Deshpande, Fluid mechanics of bubble growth and collapse in a thermal ink-jet printhead, in: SPSE/SPIES Electronic Imaging Devices and Systems Symposium, 1989.

[3] P.A. Torpey, Prevention of air ingestion in a thermal ink-jet device, in: Proceedings of the Fourth International Congress on Advances in Non-Impact Print Technologies, 1988.

[4] D.B. Wallace, A method of characteristics model of a drop-on-demand ink-jet device using an integral method drop formation model, in: Proceedings of the ASME Winter Annual Meeting, 89-WA/FE-4, San Francisco, CA, 1989.

[5] C. Chan, J. Mazumder, M.M. Chen, A two-dimensional transient model for convection in a laser melted pool, Metall. Trans. A 15A (1984) 2175–2184.

[6] H. Liu, E.J. Lavernia, R.H. Rangel, Numerical investigation of micropore formation during substrate impact of molten droplets in plasma spray processes, Atomizat. Sprays 4 (1994) 369–384.

[7] G. Trapaga, E.F. Matthys, J.J. Valencia, J. Szekely, Fluid flow, heat transfer, and solidificaton of molten metal droplets impinging on substrates – comparison of numerical and experimental results, Metall. Trans. B 23 (6) (1992) 701–718.

[8] D. Adalsteinsson, J.A. Sethian, An overview of level set methods for etching, deposition, and lithography development, IEEE Trans. Semicond. Manuf. 10 (1) (1997) 167–184.

[9] J.J. Helmsen, P. Colella, E.G. Puckett, Non-convex profile evolution in two dimensions using volume of fluids, Technical Report LBNL-40693, Lawrence Berkeley National Laboratory, 1997.
[10] J.J. Helmsen, P. Colella, E.G. Puckett, M. Dorr, Two new methods for simulating photolithography development in three dimensions, in: Proceeedings of the 10th SPIE Optical/Laser Microlithography Conference, vol. 2726, SPIE, San Jose, CA, 1996, pp. 253–261.
[11] J.J. Helmsen, A comparison of three-dimensional photolithography simulators, Ph.D. thesis, Department of Electrical Engineering, University of California, Berkeley, Berkeley, CA, 1994.
[12] A.J. Chorin, Flame advection and propagation algorithms, J. Comput. Phys. 35 (1980) 1–11.
[13] A.F. Ghoniem, A.J. Chorin, A.K. Oppenheim, Numerical modeling of turbulent flow in a combustion tunnel, Phil. Trans. R. Soc. Lond. A 304 (1982) 303–325.
[14] J.E. Pilliod, A second-order unsplit method for modeling flames in two-dimensional compressible flow, Phd thesis, Department of Mathematics, University of California, Davis, September 1996.
[15] J.A. Sethian, Turbulent combustion in open and closed vessels, J. Comput. Phys. 54 (1984) 425–456.
[16] L. Margolin, J.M. Reisner, P.K. Smolarkiewicz, Application of the volume-of-fluid method to the advection-condensation problem, Mon. Wea. Rev. 125 (1997) 2265–2273.
[17] A.R. Bishop, L.J. Campbell, P.J. Channell, Fronts, Interfaces and Patterns, North-Holland, Amsterdam, 1984.
[18] S. Osher, J.A. Sethian, Fronts propagating with curvature-dependent speed: algorithms based on Hamilton–Jacobi formulations, J. Comput. Phys. 79 (1) (1988) 12–49.
[19] J.A. Sethian, Tracking interfaces with level sets, AMERICAN SCIENTIST 85 (3) (1997) 254–263.
[20] M. Sussman, P. Smereka, S. Osher, A level set approach for computing solutions to incompressible two-phase flow, J. Comput. Phys. 114 (1) (1994).
[21] M. Sussman, E.G. Puckett, A coupled level set and volume of fluid method for computing 3d and axisymmetric incompressible two-phase flows, J. Comput. Phys. 162 (2000) 301–337.
[22] M.M. Sussman, A second order coupled level set and volume of fluid method for computing growth and collapse of vapor bubbles, J. Comput. Phys. 187 (2003) 110–136.
[23] J. Glimm, O. McBryan, A computational model for interfaces, Adv. Appl. Math. 6 (1985) 422–435.
[24] S.O. Unverdi, G. Tryggvason, Computations of multi-fluid flows, Physica D 60 (1992) 70–83.
[25] H.N. Oguz, A. Prosperetti, Dynamics of bubble growth and detachment from a needle, J. Fluid Mech. 257 (1993) 111–145.
[26] J. Strain, A boundary integral approach to unstable solidification, J. Comput. Phys. 85 (1989) 342–389.
[27] R. Scardovelli, S. Zaleski, Direct numerical simulation of free-surface and interfacial flow, Annu. Rev. Fluid Mech. 31 (1999) 567–603.
[28] J.A. Sethian, P. Smereka, Level set methods for fluid interfaces, Annu. Rev. Fluid Mech. 35 (2003) 341–372.
[29] W.J. Rider, D.B. Kothe, Reconstructing volume tracking, J. Comput. Phys. 141 (2) (1998) 112–152.
[30] P. Colella, H.M. Glaz, R. Fergusona, E.G. Puckett, Multifluid algorithms for Eulerian finite difference methods, in preparation.
[31] G.H. Miller, E.G. Puckett, A high-order Godunov method for multiple condensed phases, J. Comput. Phys. 128 (1) (1996) 134–164.
[32] E.G. Puckett, J.S. Saltzman, A 3-d adaptive mesh refinement algorithm for multimaterial gas dynamics, Physica D 60 (1992) 84–104.
[33] F.L. Addessio, D.E. Carroll, J.K. Dukowicz, F.H. Harlow, J.N. Johnson, B.A. Kashiwa, M.E. Maltrud, H.E. Rupple, CAVEAT: a computer code for fluid dynamics problems with large distortion and internal slip, Technical Report LA-10613-MS, Los Alamos National Laboratory, 1986.
[34] K.S. Holian, S.J. Mosso, D.A. Mandell, R. Henninger, MESA: a 3-d computer code for armor/anti-armor applications, Technical Report LA-UR-91-569, Los Alamos National Laboratory, 1991.
[35] D.B. Kothe, J.R. Baumgardner, S.T. Bennion, J.H. Cerutti, B.J. Daly, K.S. Holian, E.M. Kober, S.J. Mosso, J.W. Painter, R.D. Smith, M.D. Torrey, PAGOSA: a massively-parallel, multi-material hydro-dynamics model for three-dimensional high-speed flow and high-rate deformation, Technical Report LA-UR-92-4306, Los Alamos National Laboratory, 1992.
[36] J.M. McGlaun, S.L. Thompson, C.N. Kmetyk, M.G. Elrick, A brief description of the three dimensional shock wave physics code CTH, Technical Report SAND92-1916 UC-410, Sandia National Laboratory, September 1992.
[37] J.G. Blank, G.H. Miller, The fate of organic compounds in cometary impacts, in: A.F.P. Houwing, A. Paull, R.R. Boyce, P.M. Danehy, H. Hannemann, J.J. Kurtz, T.J. McIntyre, S.J. McMahon, D.J. Mee, R.J. Sandeman, H. Tanno (Eds.), Proceedings of the 21st International Symposium on Shock Waves, vol. II, Panther Publishing, Fyshwick, Australia, 1998, pp. 1205–1210.
[38] G.H. Miller, Jetting in oblique, asymmetric impacts, Icarus 134 (1998) 163–175.
[39] E.G. Puckett, G.H. Miller, The numerical computation of jetting impacts, in: B. Sturtevant, J. Sheperd, H. Hornung (Eds.), Proceedings of the 20th International Symposium on Shock Waves, World Scientific, New Jersey, 1996, pp. 1467–1472.
[40] G.H. Miller, P. Colella, A conservative three-dimensional Eulerian method for coupled fluid-solid shock capturing, J. Comput. Phys. 183 (2002) 26–82.
[41] E.G. Puckett, A.S. Almgren, J.B. Bell, D.L. Marcus, W.J. Rider, A high-order projection method for tracking fluid interfaces in variable density incompressible flows, J. Comput. Phys. 130 (2) (1997) 269–282.

[42] J.B. Bell, C.N. Dawson, G.R. Shubin, An unsplit, higher order Godunov method for scalar conservation laws in multiple dimensions, J. Comput. Phys. 74 (1988) 1–24.
[43] P. Colella, Multidimensional upwind methods for hyperbolic conservation laws, J. Comput. Phys. 87 (1990) 171–200.
[44] R.J. LeVeque, High-resolution conservative algorithms for advection in incompressible flow, SIAM J. Numer. Anal. 33 (2) (1996) 627–665.
[45] B. van Leer, Towards the ultimate conservative difference scheme. V. A second-order sequel to Godunov's method, J. Comput. Phys. 32 (1979) 101–136.
[46] R. DeBar, A method in two-d Eulerian hydrodynamics, Technical Report UCID-19683, Lawrence Livermore National Laboratory, March 1974.
[47] W.F. Noh, P.R. Woodward, SLIC (Simple Line Interface Calculation), in: A.I. van der Vooren, P.J. Zandbergen (Eds.), Lecture Notes in Physics, vol. 59, Springer-Verlag, New York, 1976, pp. 330–340.
[48] P. Colella, L.F. Henderson, E.G. Puckett, A numerical study of shock wave refraction at a gas interface, in: Proceedings of the AIAA 9th Computational Fluid Dynamics Conference, Buffalo, New York, 1989, pp. 426–439, aIAA-89-1973.
[49] L.F. Henderson, P. Colella, E.G. Puckett, On the refraction of shock waves at a slow-fast gas interface, J. Fluid Mech. 224 (1991) 1–27.
[50] E.G. Puckett, A numerical study of shock wave refraction at a CO_2/CH_4 interface, in: J. Glimm, A.J. Majda (Eds.), Multidimensional Hyperbolic Problems and Computations, IMA Volumes in Mathematics and Its Applications, vol. 29, Springer-Verlag, 1991, pp. 261–280.
[51] N. Whitaker, Numerical solution of the Hele–Shaw equations, J. Comput. Phys. 90 (1990) 176–199.
[52] C.W. Hirt, B.D. Nichols, Volume of Fluid (VOF) method for the dynamics of free boundaries, J. Comput. Phys. 39 (1981) 201–225.
[53] B.D. Nichols, C.W. Hirt, R.S. Hotchkiss, SOLA-VOF: a solution algorithm for transient fluid flow with multiple free boundaries, Technical Report LA-8355, Los Alamos National Laboratory, August 1980.
[54] M.D. Torrey, L.D. Cloutman, R.C. Mjolsness, C.W. Hirt, NASA-VOF2D: a computer program for incompressible flows with free surfaces, Technical Report LA-10612-MS, Los Alamos National Laboratory, December 1985.
[55] M.D. Torrey, R.C. Mjolsness, L.R. Stein, NASA-VOF3D: a three-dimensonal computer program for incompressible flows with free surfaces, Technical Report LA-11009-MS, Los Alamos National Laboratory, July 1987.
[56] D.B. Kothe, R.C. Mjolsness, RIPPLE: a new model for incompressible flows with free surfaces, AIAA J. 30 (11) (1992) 2694–2700.
[57] D.B. Kothe, R.C. Mjolsness, M.D. Torrey, RIPPLE: a computer program for incompressible flows with free surfaces, Technical Report LA-12007-MS, Los Alamos National Laboratory, April 1991.
[58] C.W. Hirt, Flow-3D Users Manual, Flow Sciences, Inc, 1988.
[59] N. Ashgriz, J.Y. Poo, FLAIR, Flux line-segment model for advection and interface reconstruction, J. Comput. Phys. 93 (1991) 449–468.
[60] B. LaFaurie, C. Nardone, R. Scardovelli, S. Zaleski, G. Zanetti, Direct numerical simulation of interface breakup and atomisation, in: Proceedings of ICLASS-94, Rouen, France, 1994.
[61] E.G. Puckett, A volume-of-fluid interface tracking algorithm with applications to computing shock wave refraction, in: H. Dwyer (Ed.), Proceedings of the Fourth International Symposium on Computational Fluid Dynamics, Davis, CA, 1991, pp. 933–938.
[62] D.L. Youngs, Time-dependent multi-material flow with large fluid distortion, in: K.W. Morton, M.J. Baines (Eds.), Numerical Methods for Fluid Dynamics, Academic Press, New York, 1982.
[63] J.E. Pilliod, An analysis of piecewise linear interface reconstruction algorithms for volume-of-fluid methods, M. S. Thesis, University of California, Davis, September 1992.
[64] L.F. Henderson, E.G. Puckett, P. Colella, Anomalous refraction of shock waves, in: K. Takayama (Ed.), Shock Waves, Springer-Verlag, Berlin, 1992, pp. 283–286.
[65] E.G. Puckett, L.F. Henderson, The anomalous refraction of shock waves in gases, Proc. Royal Soc. London A, submitted for publication.
[66] E.G. Puckett, L.F. Henderson, P. Colella, A general theory of anomalous refraction, in: R. Brun, L.Z. Dumitrescu (Eds.), Shock Waves @ Marseilles, vol. 4, Springer-Verlag, Berlin, 1995, pp. 139–144.
[67] G.H. Miller, E.G. Puckett, Edge effects in molybdenum-encapsulated molten silicate shock wave targets, J. Appl. Phys. 75 (3) (1994) 1426–1434.
[68] J.E. Pilliod, E.G. Puckett, An unsplit, second-order accurate Godunov method for tracking deflagrations and detonations, in: A.F.P. Houwing, A. Paull, R.R. Boyce, P.M. Danehy, H. Hannemann, J.J. Kurtz, T.J. McIntyre, S.J. McMahon, D.J. Mee, R.J. Sandeman, H. Tanno (Eds.), Proceedings of the 21st International Symposium on Shock Waves, vol. II, Panther Publishing, Fyshwick, Australia, 1998, pp. 1053–1058.
[69] B. LaFaurie, C. Nardone, R. Scardovelli, S. Zaleski, G. Zanetti, Modelling merging and fragmentation in multiphase flows with surfer, J. Comput. Phys. 113 (1994) 134–147.

[70] B.J. Parker, D.L. Youngs, Two and three dimensional Eulerian simulation of fluid flow with material interfaces, Technical Report 01/92, UK Atomic Weapons Establishment, Aldermaston, Berkshire, February 1992.
[71] I. Aleinov, E.G. Puckett, Computing surface tension with high-order kernels, in: K. Oshima (Ed.), Proceedings of the Sixth International Symposium on Computational Fluid Dynamics, Lake Tahoe, CA, 1995, pp. 6–13 .
[72] W. Press, B. Flannery, S. Teukolsky, W. Vetterling, Numerical Recipes in C, Cambridge University Press, Cambridge, 1988.
[73] S.T. Zalesak, Fully multidimensional flux-corrected transport algorithms for fluids, J. Comput. Phys. 31 (1979) 335–362.
[74] A.J. Chorin, Curvature and solidification, J. Comput. Phys. 57 (1985) 472–490.
[75] P. Colella, E.G. Puckett, Advanced Numerical Methods for Fluid Flow, Cambridge University Press, in press.