A VOLUME-OF-FLUID INTERFACE RECONSTRUCTION ALGORITHM THAT IS SECOND-ORDER ACCURATE IN THE MAX NORM

ELBRIDGE GERRY PUCKETT

vol. 5  no. 2  2010
A VOLUME-OF-FLUID INTERFACE RECONSTRUCTION ALGORITHM THAT IS SECOND-ORDER ACCURATE IN THE MAX NORM

Elbridge Gerry Puckett

In an article recently published in this journal the author proved there exists a two-dimensional, volume-of-fluid interface reconstruction method that is second-order accurate in the max norm. However, that article did not include an example of such an algorithm. This article contains a description of a two-dimensional, volume-of-fluid interface reconstruction method that is second-order accurate in the max norm, provided the curve that one is reconstructing is two times continuously differentiable and the length of the sides of the square grid cells is less than a constant divided by the maximum of the absolute value of the curvature of the interface. A computation made with this algorithm is presented that demonstrates the convergence rate is second-order, as expected.

1. Introduction

Let \( \Omega \subseteq \mathbb{R}^2 \) denote a two-dimensional computational domain and take an oriented curve in \( \Omega \) parametrized by \( z(s) = (x(s), y(s)) \), where \( 0 \leq s \leq s_{\text{end}} \) is arc length. Let \( L \) be a characteristic length of the computational domain \( \Omega \). Cover \( \Omega \) with a grid consisting of square cells each of side \( \Delta x \leq L \) and let

\[
h = \frac{\Delta x}{L}
\]  

be a dimensionless parameter that represents the size of a grid cell. Note that \( h \) is bounded above by 1. For the remainder of this article it is to be understood that quantities such as the arc length \( s \) are also nondimensional quantities that have been obtained by division by \( L \) as in (1), and that the curvature \( \kappa \) has been nondimensionalized by dividing by \( 1/L \).


Keywords: volume-of-fluid, interface reconstruction, front reconstruction, piecewise linear interface reconstruction, fronts, two-phase flow, multiphase systems, adaptive mesh refinement, computational fluid dynamics.

Sponsored by the U.S. Department of Energy Mathematical, Information, and Computing Sciences Division contracts DE-FC02-01ER25473 and DE-FG02-03ER25579.
The **volume-of-fluid interface reconstruction problem** is to compute an approximation \( \tilde{z}(s) \) to \( z(s) \) in \( \Omega \) using only the volume fractions \( \Lambda_{ij} \) associated with the curve \( z \) on the grid. In this paper the convention will be that the volume fraction \( \Lambda_{ij} \) in the \( ij \)-th cell is the fraction of material that lies in the \( ij \)-th cell on the “outside” of \( z(s) \)—that is, on the side towards which the outward pointing unit normal vector \( n \) to the interface \( z(s) \) points.

The purpose of this article is to describe a new volume-of-fluid method for reconstructing the interface \( z(s) = (x(s), y(s)) \) between two materials that is second-order accurate in the max norm. The main result in [24; 25] is a proof that

\[
| g(x) - \tilde{g}_{ij}(x) | \leq \left( \frac{50}{3} \kappa_{\text{max}} + C_S \right) h^2 \quad \text{for all } x \in [x_i, x_{i+1}],
\]

(2)

where \((x, \tilde{g}_{ij}(x))\) is the volume-of-fluid approximation to the interface\(^1\) \((x, g(x))\) described in this paper; \(\kappa_{\text{max}}\) is the maximum of the absolute value of the curvature of the interface in the \(3 \times 3\) block of cells \(B_{ij}\) centered on the cell \(C_{ij}\) in which one wishes to reconstruct the interface; \(x = x_i\) and \(x = x_{i+1}\) are the \(x\)-coordinates of the left and right edges of the cell \(C_{ij} = [x_i, x_{i+1}] \times [y_j, y_{j+1}]\) in which one wishes to reconstruct the interface; \(h = \Delta x / L\) is the length defined in (1) of the edges of the square grid cells; and

\[
C_S = \frac{\sqrt{3}}{2} \left\{ (2 \sqrt{2} - 1)4 \sqrt{\kappa_{\text{max}}} + \left( 1 - \frac{7(1+\sqrt{2})}{20} \right) \frac{32}{3} (\sqrt{\kappa_{\text{max}}})^3 \right\}^2.
\]

(3)

The bound in (2) holds whenever the grid size \(h\) and the maximum \(\kappa_{\text{max}}\) of the absolute value of the curvature \(\kappa(s)\) of the interface \(z(s)\)

\[
\kappa_{\text{max}} = \max_s |\kappa(s)|
\]

(4)

satisfies\(^2\)

\[
h \leq \frac{C_h}{\kappa_{\text{max}}},
\]

(5)

where

\[
C_h = \frac{1}{25}.
\]

---

\(^1\)As explained in the following paragraph, it is proven in [24] that the constraint on \(h\) in terms of \(\kappa_{\text{max}}\) in (5) ensures that one can reparametrize the interface as \(y = g(x)\) or \(x = G(y)\) locally about the cell \(C_{ij} = [x_i, x_{i+1}] \times [y_j, y_{j+1}]\) in which one wishes to reconstruct the interface. For convenience, in this article the interface will frequently be written as \(y = g(x)\) instead of \(z(s)\), it being understood that in some cells the parametrizations has to be \(x = G(y)\).

\(^2\)It is only necessary that this condition be satisfied in a neighborhood of the cell \(C_{ij}\); for example, in the \(3 \times 3\) block of cells \(B_{ij}\) centered on the cell \(C_{ij}\).
Section 2 of [24] contains a proof that the constraint in (5) on $h$ is sufficient to ensure that the interface $z(s) = (x(s), y(s))$ can be written as a single valued function $y = g(x)$ or $x = G(y)$ on any $3 \times 3$ block of cells $B_{ij}$ centered on a cell $C_{ij}$ which contains a portion of the interface.\(^3\)

In the algorithm described in this article one uses the row of three cells above and below $B_{ij}$ to determine which columns to use in the approximation to the slope $m_{ij}$ of the piecewise linear approximation

$$\tilde{g}_{ij}(x) = m_{ij} x + b_{ij} \quad (7)$$

...to the interface $y = g(x)$. For this reason one must consider the $5 \times 5$ block of cells $\tilde{B}_{ij}$ centered on the cell $C_{ij}$. However, as shown in Section 5, once one has rotated the block $\tilde{B}_{ij}$ so that the interface only enters (resp., exits) the $3 \times 3$ subblock $B_{ij}$ across its left or top edge (resp., top or right edge); it is not necessary to use the first and last columns of the larger block of cells $\tilde{B}_{ij}$.

The articles [24; 25] consist solely of a collection of proofs showing that there exists a volume-of-fluid interface reconstruction algorithm that is second-order accurate in the max norm; that is, they do not contain an example of such an algorithm. However, the proofs in [24; 25] are constructive, and the algorithm described here is based on those proofs. To date, no other volume-of-fluid interface reconstruction algorithms have been proven to be second-order accurate in the max norm. Section 6 contains a computational example to demonstrate that this algorithm produces an approximation to $\cos x$ for $0 \leq x \leq \pi$ that is a second-order accurate in the max norm.

A detailed statement of the problem. Suppose that one is given a simply connected computational domain $\Omega \in R^2$ that is divided into two distinct, disjoint regions $\Omega_1$ and $\Omega_2$ such that $\Omega_1 \cup \Omega_2 = \Omega$. Let $\Omega_1$ be referred to as material 1 and $\Omega_2$ as material 2. (Although these algorithms have historically been known as “volume-of-fluid” methods, they are frequently used to model the interface between any two materials, including gases, liquids, solids and any combination thereof; for example, see [5; 14; 15; 16].)

Let $z(s) = (x(s), y(s))$, where $s$ is arc length, denote the interface between these two materials and assume that the interface has been oriented so that as one traverses the interface with increasing arc length material 1 lies to the right. Cover $\Omega$ with a uniform square grid of cells, each with side $h$, and let $\Lambda_{ij}$ denote the fraction of material 1 in the $ij$-th cell.

\(^3\)In that article and this one interfaces that are undergoing topological changes, such as when a droplet separates into two droplets, are not considered. In order for this algorithm to achieve second-order accuracy, the interface must be a single-valued $C^2$ function in the $3 \times 3$ block $B_{ij}$ surrounding the cell $C_{ij}$. 
Each number $\Lambda_{ij}$ satisfies $0 \leq \Lambda_{ij} \leq 1$ and is called the \textit{volume fraction} (of material 1) in the $ij$-th cell.\footnote{Even though in two dimensions $\Lambda_{ij}$ is technically an area fraction, the convention is to refer to it as a \textit{volume fraction}.} Note that

$$0 < \Lambda_{ij} < 1$$

if and only if a portion of the interface $z(s)$ lies in the $ij$-th cell. Similarly, $\Lambda_{ij} = 1$ means the $ij$-th cell only contains material 1, and $\Lambda_{ij} = 0$ means it only contains material 2.

Consider the following problem. Given only the collection of volume fractions $\Lambda_{ij}$ in the grid covering $\Omega$, \textit{reconstruct} $z(s)$; in other words, find a piecewise linear approximation $\tilde{z}$ to $z$ in each cell $C_{ij}$ that contains a portion of the interface $z(s)$. Furthermore, the approximate interface $\tilde{z}$ must have the property that the volume fractions $\tilde{\Lambda}_{ij}$ due to $\tilde{z}$ are identical to the original volume fractions $\Lambda_{ij}$; that is,

$$\tilde{\Lambda}_{ij} = \Lambda_{ij} \quad \text{for all cells } C_{ij}. \quad (9)$$

An algorithm for finding such an approximation is known as a \textit{volume-of-fluid interface reconstruction method}.

The property that $\tilde{\Lambda}_{ij} = \Lambda_{ij}$ is the principal feature that distinguishes volume-of-fluid interface reconstruction methods from other interface reconstruction methods. This ensures that the computational value of the total volume of each material is conserved. In other words, all volume-of-fluid interface reconstruction methods are conservative in that they conserve the volume of each material in the computation. When the underlying numerical method is a conservative finite difference method that one is using to model a system of hyperbolic conservation laws (e.g., gas dynamics) this can be essential since, for example, in order to obtain the correct shock speed it is necessary for all of the conserved quantities to be conserved by the underlying numerical method [13]. More generally, a necessary condition for the numerical method to converge to the correct weak solution of a system of hyperbolic conservation laws is that all of the quantities that are conserved in the underlying partial differential equation must be conserved by the numerical method [12].

Volume-of-fluid methods have been used by researchers to track material interfaces since at least the mid 1970s [18; 19]. Researchers have developed a variety of volume-of-fluid algorithms for modeling everything from flame propagation [3] to curvature and solidification [4]. In particular, the problem of developing high-order accurate volume-of-fluid methods for modeling the curvature and surface tension of an interface has received a lot of attention [1; 2; 4; 7; 32; 22]. Volume-of-fluid methods were among the first interface tracking algorithms to be implemented in codes originally developed at the U.S. National Laboratories, and subsequently
released to the general public, that were capable of tracking material interfaces in a variety of complex material flow problems [6; 9; 17; 30; 31]. They continue to be widely used at these institutions, as well as by the general scientific community.

This article does not contain work concerning the related problem of approximating the movement of the interface in time, for which one would use a *volume-of-fluid advection algorithm*. The interested reader may wish to consult [21; 27; 28] for a detailed description and analysis of several such algorithms. In this article only the accuracy that one can obtain when using a volume-of-fluid interface reconstruction algorithm to approximate a given *stationary* interface \( z(s) \) is considered.

### 2. Assumptions and definitions

**Notation.** The *center cell* \( C_{ij} = [x_i, x_{i+1}] \times [y_j, y_{j+1}] \) is the square grid cell with side \( h \) that contains a portion of the interface \( z(s) = (x(s), y(s)) \) for \( s \) in some interval, say \( s \in (s_l, s_r) \), in which one wishes to reconstruct the interface. This is equivalent to saying that \( 0 < \Lambda_{ij} < 1 \). In what follows the \( 5 \times 5 \) block of square cells — each with side \( h \) — centered on the center cell \( C_{ij} \), as shown, for example, in Figure 1, will be denoted \( \tilde{B}_{ij} = [x_{i-2}, x_{i+3}] \times [y_{j-2}, y_{j+3}] \). In addition, the subblock of \( \tilde{B}_{ij} \) which consists of the \( 3 \times 3 \) subblock of cells centered on the cell

![Figure 1](image-url)

*Figure 1.* In this example the interface is \( g(x) = \tanh(x) \) and material 1 lies *below* the curve. Note that all three of the column sums are *exact*, but that for the inverse function \( x = g^{-1}(y) \), only the (horizontal) center column sum is exact. (Exact column sums are defined in Section 4 below.) The cell in which one wishes to reconstruct the interface is \( C_{ij} \), the \( 3 \times 3 \) block of cells centered on \( C_{ij} \) is \( B_{ij} \), and the \( 5 \times 5 \) block of cells centered on \( C_{ij} \) is \( \tilde{B}_{ij} \).
$C_{ij}$ will be denoted $B_{ij} = [x_{i-1}, x_{i+2}] \times [y_{j-1}, y_{j+2}]$, and the $3 \times 5$ subblock of $\tilde{B}_{ij}$, which is $B_{ij}$ together with the row of 3 cells $C_{i-1,j-2}$, $C_{i,j-2}$ and $C_{i+1,j-2}$ added to its bottom and the row of 3 cells $C_{i-1,j+2}$, $C_{i,j+2}$ and $C_{i+1,j+2}$ added to its top, will be denoted $\tilde{\tilde{B}}_{ij} = [x_{i-1}, x_{i+2}] \times [y_{j-2}, y_{j+3}]$. The coordinates of the vertical edges of the cells in $\tilde{B}_{ij}$ are denoted $x_{i-2}, x_{i-1}, x_{i}, x_{i+1}, x_{i+2}$ and $x_{i+3}$ and the horizontal edges by $y_{j-2}, y_{j-1}, y_{j}, y_{j+1}, y_{j+2}$ and $y_{j+3}$ as shown, for example, in Figure 1. It will always be the case that $x_{i+1} - x_{i} = h$, $y_{j+1} - y_{j} = h$, etc.

**Assumptions concerning the interface.** In this article the exact interface $z(s) = ((x(s), y(s))$ is assumed to satisfy the following conditions:

I. The interface $z$ is two times continuously differentiable; in other words,

$$z(s) \in C^2(\Omega).$$

II. The grid size $h$ and the maximum value

$$\kappa_{\text{max}} = \max_{s} |\kappa(s)|$$

of the absolute value of the curvature $\kappa(s)$ of the interface satisfy the following constraint in terms of each other:

$$h \leq \frac{C_{h}}{\kappa_{\text{max}}},$$

where

$$C_{h} = \frac{1}{25}.$$

**Remark.** One can show that the constraint in (11) prevents configurations in which the interface enters the center cell $C_{ij}$, exits it, and then enters it again, before exiting the $5 \times 5$ block of cells $\tilde{B}_{ij}$, as shown in Figure 2. In particular, one can use the constraint in (11) to show that the interface does not have hairpin turns which are on the order of a grid cell. See [24] for a proof of these facts.

Note that it may be possible for the interface to pass through the center cell, then exit the $3 \times 3$ block $B_{ij}$, “wander around the computational domain”, and then reenter the $3 \times 3$ block $B_{ij}$ and the center cell $C_{ij}$ again. The constraint in (11) simply guarantees that given a point on the interface that lies in the center cell $C_{ij}$, one can find an orientation of the $3 \times 3$ block $B_{ij}$ such that locally the interface can be written as a single-valued function on the interval $[x_{i-1}, x_{i+2}]$ such that the

---

5Whenever possible, the same notation is used in this article as in [24; 25]. One significant change, however, is that the lower left corner of the center cell $C_{ij}$ is now $(x_{i}, y_{j})$ rather than $(x_{i-1}, y_{j-1})$ as it was denoted in [24].

6See footnote 2.
Figure 2. In this example, $h = 1$ and the interface is the parabola $g(x) = a(x - x_c)^2 - \frac{1}{2}$ with $a = 9$. Consequently, the maximum curvature of the interface is

$$\kappa_{\text{max}} = 2a = 18 > C_h h^{-1} = \frac{1}{25},$$

and hence the constraint on the cell size $h$ in (5) is not satisfied. As one can see from the figure, the interface enters the $3 \times 3$ block of cells $B_{ij}$ through the top edge of the left column, passes through the center cell $C_{ij}$, exits the $3 \times 3$ block of cells $B_{ij}$ through the bottom edge of the center column (i.e., the line $y = y_{j-1}$), and then passes through $B_{ij}$ again; the second path being a reflection about the line $x = x_c$ of the first. The constraint on $h$ with respect to the maximum curvature $\kappa_{\text{max}}$ in (5) ensures that the interface does not have sharp or “hairpin” turns that are on the scale of the $3 \times 3$ block of cells $B_{ij}$, such as the one illustrated here. A finer grid (i.e., smaller $h$) is required in order to resolve curves such as this one.

curve enters the $3 \times 3$ block $B_{ij}$, passes through the center cell once — and only once — and then exits $B_{ij}$. It does not prevent the curve from eventually reentering the center cell after traversing the domain for a large number of cell lengths. In this article it is assumed that this latter case does not occur.
3. Rotation of the 5 by 5 block

Given a cell $C_{ij}$ that contains a portion $y = g(x)$ of the interface, or equivalently, a cell $C_{ij}$ in which $0 < \Lambda_{ij} < 1$, assume that the $5 \times 5$ block of cells $\hat{B}_{ij}$ centered on $C_{ij}$ has been rotated so that in the rotated coordinate frame the bottom row of cells in the $3 \times 5$ subblock of cells $\hat{B}_{ij}$ satisfy

$$\Lambda_{i-1,j-2} = 1, \quad \Lambda_{ij-2} = 1, \quad \Lambda_{i+1,j-2} = 1.$$ 

This ensures that the interface does not exit the $3 \times 5$ subblock of cells $\hat{B}_{ij}$ across its bottom edge. Not only does this reduce the number of cases that one must consider in the description of the algorithm, but it also reduces the number of cases one must consider in the implementation of the algorithm. This is because the Symmetry Lemma of [24, page 119] states that all configurations of the interface with respect to the $3 \times 3$ block $B_{ij}$ are equivalent to the following two cases:

- **Configuration A:** The interface enters $B_{ij}$ across the left edge of $B_{ij}$ and exits across the right edge of $B_{ij}$ (as shown, for example, in Figure 1).

- **Configuration B:** The interface enters $B_{ij}$ across the left edge of $B_{ij}$ and exits across the top edge of $B_{ij}$ (as shown, for example, in Figure 4).

Provided only that the interface satisfies the conditions in (5) and (10), these two cases are equivalent to all of the other ways in which the interface can enter the $3 \times 3$ block of cells $B_{ij}$, pass through the center cell $C_{ij}$, and exit the block $B_{ij}$. In other words, rotating the block $B_{ij}$ by 0, 90, 180 or 270 degrees and/or interchanging the direction traversed by the arc length parameter $s \rightarrow -s$, one can arrive at a configuration that is identical to one of the two configurations listed above. This is a consequence of the Symmetry Lemma cited above.

In the implementation of this algorithm, for the purposes of producing the results shown in Section 6, the case in which — after the $5 \times 5$ block of cells $\hat{B}_{ij}$ has been rotated — the interface enters $B_{ij}$ across the top edge and exits it across the right edge is also included. In other words, the symmetric image of the example shown in Figure 4 is also included in this implementation of the algorithm, although according to the Symmetry Lemma this is not strictly necessary.

During the course of proving the results in [24; 25], or in developing a second-order accurate volume-of-fluid interface reconstruction method such as the one described here, it is often necessary to rotate the $5 \times 5$ block of cells $\hat{B}_{ij}$ centered on $C_{ij}$ by 90, 180, or 270 degrees and/or reflect the coordinates about one of the coordinate axes: $x \rightarrow -x$ or $y \rightarrow -y$. No other coordinate transformations besides one of these three rotations and a possible reversal of one or both of the variables $x \rightarrow -x$ and/or $y \rightarrow -y$ are required in order for the algorithm studied in this article and the articles in [24; 25] to converge to the exact interface as $h \rightarrow 0$. Furthermore,
these coordinate transformations are only used to determine a first-order accurate approximation $m_{ij}$ to $g'(x_c)$ in the center cell. The grid covering the domain $\Omega$ always remains the same.

Thus, if one is using the interface reconstruction algorithm as part of a numerical method to solve a more complex problem than the one posed here (e.g., the movement of a fluid interface where the underlying fluid flow is a solution of the Euler or Navier–Stokes equations), it is not necessary to perform these coordinate transformations on the underlying numerical fluid flow solver. Therefore, unless noted otherwise, in what follows the interface will always be written $y = g(x)$ and the coordinates of the edges of the cells in the $3 \times 3$ block $B_{ij}$ will be denoted by $x_{i-1}, x_i, x_{i+1}, y_{j-1}, y_j, y_{j+1}, y_{j+2}$, it being implicitly understood that a transformation of the coordinate system as described above may have been performed in order for this representation of the interface to be valid, and that the names of the variables $x$ and $y$ might have been interchanged in order to write the interface as $y = g(x)$.

4. Column sums

Let $S_{i-1}$, $S_i$ and $S_{i+1}$ represent the left, center and right column sums respectively in the $3 \times 3$ subblock of cells $B_{ij}$ centered on $C_{ij}$:

$$S_{i-1} = \sum_{j'=j-1}^{j+1} \Lambda_{i-1,j'}, \quad S_i = \sum_{j'=j-1}^{j+1} \Lambda_{ij'}, \quad S_{i+1} = \sum_{j'=j-1}^{j+1} \Lambda_{i+1,j'}.$$  \hspace{1cm} (12)

The volume fraction $\Lambda_{ij}$ in the $ij$-th cell $C_{ij}$ is a nondimensional way of storing the volume of material 1 in that cell, while the $i$-th column sum $S_i$ defined above is a nondimensional way of storing the total volume of material 1 in the column of three cells centered on the $ij$-th cell, and similarly for $S_{i-1}$ and $S_{i+1}$.

Now consider an arbitrary column consisting of three cells with left edge $x = x_i$ and right edge $x = x_{i+1}$. Furthermore, assume that the interface can be written as a function $y = g(x)$ on the interval $[x_i, x_{i+1}]$. Assume also that the interface enters the column through its left edge, exits the column through its right edge and does not cross the top or bottom edges of the column, as is the case with each of the columns $S_{i-1}$, $S_i$ and $S_{i+1}$ in the $3 \times 3$ subblock of cells $B_{ij}$ centered on the cell of interest $C_{ij}$ shown in the example in Figure 1. Then, in particular, the total volume of material 1 that occupies the three cells of the center column and lies below the interface $g(x)$ is equal to the integral of $(g(x) - y_{j-1})$ over the interval $[x_i, x_{i+1}]$. This leads to the following relationship between the column sum and the normalized volume of material 1 in the column:

$$S_i = \sum_{j'=j-1}^{j+1} \Lambda_{ij'} = \frac{1}{h^2} \int_{x_i}^{x_{i+1}} (g(x) - y_{j-1}) \, dx.$$  \hspace{1cm} (13)
This in turn leads to the following definition.

**Definition.** Assume that the interface \( y = g(x) \) enters the \( i \)-th column through its left edge and exits the \( i \)-th column through its right edge and does not cross the top or bottom edges of the column. Then the column sum \( S_i \) is exact whenever \((13)\) holds. Integrals such as the one on the right in \((13)\) will be referred to as the normalized integral of \( g \) in the \( i \)-th column.\(^7\)

Given the \( 3 \times 3 \) block of cells \( B_{ij} \) surrounding a cell \( C_{ij} \) that contains a portion \( y = g(x) \) of the interface, the accuracy of the algorithm described in this paper is based on how well the column sums \( S_{i-1}, S_i \) and \( S_{i+1} \) approximate the normalized integral of \( g \) in the \((i - 1)\)-st, \( i \)-th, and \((i + 1)\)-st column. This is because if two of the column sums \( S_{i+α} \) and \( S_{i+β} \) with \( α, β = 1, 0, -1 \) and \( α \neq β \) are exact, then the slope

\[
m_{ij} = \frac{S_{i+β} - S_{i+α}}{β - α} \tag{14}\]

will be a first-order accurate approximation to \( g'(x_c) \), where \( x_c = \frac{1}{2}(x_{i+1} - x_i) \), as shown in \((18)\) below. (This is Theorem 23 in [24].) It then follows that the piecewise linear approximation

\[
\tilde{g}_{ij}(x) = m_{ij} x + b_{ij} \tag{15}\]

to the portion of the interface \( g(x) \) in \( C_{ij} \) is pointwise second-order accurate, as shown in \((19)\) below. (This is Theorem 24 in [24].) Therefore, one should use one of the following three slopes for \( m_{ij} \) in \((15)\):

\[
m^l_{ij} = (S_i - S_{i-1}), \quad m^c_{ij} = \frac{1}{2}(S_{i+1} - S_{i-1}), \quad m^r_{ij} = (S_{i+1} - S_i). \tag{16}\]

**Example 1.** In order to see why one of the three slopes in \((16)\) will be the best choice for \( m_{ij} \), consider the case when the interface is a line \( g(x) = m x + b \). In this case the \( 3 \times 3 \) subblock of cells \( B_{ij} \) has two exact column sums as shown in Figure 3. Note that in this particular orientation of \( B_{ij} \), \( g \) has two exact column sums; namely the sums in the first and second columns. It is easy to check that

\[
m = \frac{1}{h^2} \int_{x_{i-1}}^{x_i} (g(x) - y_{j-1}) \, dx - \frac{1}{h^2} \int_{x_{i-2}}^{x_{i-1}} (g(x) - y_{j-1}) \, dx
\]

\[= (S_i - S_{i-1}) = m^l_{ij}.
\]

\(^7\)In [24] an exact column sum was mistakenly defined as

\[
S_i \equiv \sum_{j' = j-1}^{j+1} A_{ij'} = \frac{1}{h^2} \int_{x_i}^{x_{i+1}} (g(x) - y_{j-1} h) \, dx;
\]

that is, \( y_{j-1} h \) appears in the integrand, rather than just \( y_{j-1} \). The correct definition appears here.
Figure 3. Here the interface $g$ is a line $g(x) = m \ x + b$ that has two exact column sums; namely the sums in the first and second columns. In this case the slope $m_{ij}$ from (16) is exactly equal to the slope $m$ of the interface: $m_{ij} = m$. It is always the case that when the exact interface is a line on a grid of square cells one can find an orientation of the $3 \times 3$ block of cells $B_{ij}$ such that at least one of the divided differences of the column sums in (16) is exact.

In this example the divided difference $m_{ij}$ of the column sums $S_{i-1}$ and $S_{i}$ is exactly equal to the slope $m$ of the exact interface. In fact, it is always the case that when the exact interface is a line and the grid consists of square cells, one can find an orientation of the $3 \times 3$ subblock of cells $B_{ij}$ such that at least one of the divided differences of the column sums in (16) is exact. For example, note that in the case shown in Figure 3 one could rotate the $3 \times 3$ block of cells 90 degrees clockwise and then the correct slope to use when forming the piecewise linear approximation $\tilde{g}_{ij}(x) = m_{ij} \ x + b_{ij}$ would be $m_{ij} = m_{ij}^r$ in the rotated coordinate frame. One can easily check that this choice for $m_{ij}$ would again be exactly equal to the slope $m$ of the exact interface (in the rotated coordinate frame).

Example 2. However, as demonstrated in Example 2 of [25], there are instances in which the interface satisfies (5) but the center column sum $S_{i}$ is not exact. An example was shown in Figure 4. All of the work in [25] is devoted to showing that when the interface satisfies (5), the center column sum $S_{i}$ will still be exact to
Figure 4. An example of a circular interface \( g(x) \) that satisfies (5), but for which the center column sum is not exact in any of the four standard orientations of the grid. Consequently, any approximation \( m_{ij} \) to the slope \( g'(x_c) \) of the form (14) must have a center column sum \( S_i \) that is not exact. (See [25, Example 2] for more details.) Theorem 4 of [25] states that if (5) is satisfied, then the error between the column sum \( S_i \) and the normalized integral of \( g \) over the center column is \( O(h) \); that is, (5) implies that (17) holds. This suffices to ensure that (18) is true, and hence that (19) is true.

\[ O(h) :^8 \]

\[
\left| S_i - \frac{1}{h^2} \int_{x_i}^{x_{i+1}} (g(x) - y_{j-1}) \, dx \right| \leq C_S h, \tag{17}
\]

where \( C_S \) is defined in (3). This is sufficient for either the left- or the right-sided difference in (16) to satisfy

\[
|m_{ij} - g'(x_c)| \leq \left( \frac{26}{3} \kappa_{\text{max}} + C_S \right) h, \tag{18}
\]

where \( \kappa_{\text{max}} \) is defined in (4). This, in turn, is sufficient for the piecewise linear volume-of-fluid approximation \( \tilde{g}_{ij} = m_{ij} x + b_{ij} \) to still be second-order accurate

\[^8\text{In [24] the definition that the center column sum is exact to } O(h) \text{ was mistakenly defined as}
\]

\[
\left| S_i - \frac{1}{h^2} \int_{x_i}^{x_{i+1}} (g(x) - y_{j-1}h) \, dx \right| \leq C_S h,
\]

that is, \( y_{j-1}h \) appears in the integrand, rather than just \( y_{j-1} \). The correct definition appears here.
in the max norm:

\[ |g(x) - \tilde{g}_{ij}(x)| \leq \left( \frac{50}{3} \kappa_{\text{max}} + C_S \right) h^2 \quad \text{for all } x \in [x_i, x_{i+1}]. \quad (19) \]

See Section 4 of [24] for proofs of (18) and (19).

To summarize, once the $5 \times 5$ block of cells $\tilde{B}_{ij}$ centered on the cell $C_{ij}$ in which one wishes to reconstruct the interface has been rotated as described in Section 3, the interface reconstruction algorithm is based on choosing the slope $m_{ij}$ of the piecewise linear approximation $\tilde{g}_{ij} = m_{ij} x + b_{ij}$ to the interface $g(x)$ to be one of the three divided differences of the column sums $S_{i-1}$, $S_i$ and $S_{i+1}$ from the $3 \times 3$ subblock $B_{ij}$ centered on the cell $C_{ij}$ as shown in (16). The best choice is when both column sums are exact, which — provided that the condition in (5) is satisfied — is true in all but one case.

This one case is the one in which the interface $g$ satisfies (5) yet exits the $i$-th column $S_i$ across its top edge as shown in Figure 4. (In this particular case the interface is always monotonically increasing.) Example 2 of [25] demonstrates that this case can occur for any value of $h$, no matter how small. However, in [25] it is proven that when this case occurs, one of the two divided differences of column sums, $m^l_{ij}$ or $m^r_{ij}$, will still satisfy (18). Thus, choosing this quantity for the slope $m_{ij}$ in $\tilde{g}_{ij} = m_{ij} x + b_{ij}$ still yields a pointwise second-order accurate approximation to $g$; that is, the bound in (19) remains true. The following section contains a description of an algorithm for determining which of these cases is present, and hence which of the slopes in (16) — the first or the third — will yield a second-order accurate approximation to the interface in $C_{ij}$.

5. A description of the algorithm

Before proceeding one should note that there are a variety of ways to implement this algorithm. In particular, one can implement it so that it is not necessary rotate the $5 \times 5$ block $\tilde{B}_{ij}$. The description given here was chosen because it seems to be the easiest one to follow. Furthermore this is the way in which the algorithm was implemented in order to produce the computational results shown in Section 6.

There are two steps involved in computing the approximation $\tilde{g}_{ij}(x)$ to the interface $g(x)$ in a given cell $C_{ij}$.

I. Determine the slope $m_{ij}$ of the piecewise linear approximation

\[ \tilde{g}_{ij}(x) = m_{ij} x + b_{ij} \]

to the interface $g(x)$ in the cell $C_{ij}$.

II. Determine the y-intercept $b_{ij}$ of $\tilde{g}_{ij}(x)$.
Step I. Given that the $5 \times 5$ block $\tilde{B}_{ij}$ has been placed in the configuration described in Section 3 above, one need only consider the five cases listed below; namely Cases 1(a), 1(b) and Cases 2–4. From the discussion on column sums in Section 4 it is apparent that one needs two column sums that are either exact, or a left (resp., right) column sum that is exact and a center column sum that is exact to $O(h)$ as shown, for example, in Figure 4 (page 210).

The constraint in (5) on the interface $z(s)$ ensures that, once the $5 \times 5$ block of cells $\tilde{B}_{ij}$ has been rotated as described above, only the following cases can occur:

(1) The center column sum $S_i$ is exact to $O(h)$, and hence satisfies (17), as shown, for example, in Figure 4, and one of the following two cases hold:

(a) The left column sum $S_{i-1}$ is exact, in which case one uses the left-sided divided difference:

$$m_{ij} = m_{ij}^l = S_i - S_{i-1}. \quad (20)$$

(b) The right column sum $S_{i-1}$ is exact, in which case one uses the right-sided divided difference:

$$m_{ij} = m_{ij}^r = S_{i+1} - S_i. \quad (21)$$

(2) All three column sums $S_{i-1}$, $S_i$ and $S_{i+1}$ are exact as shown, for example, in Figure 1. In this case one uses the centered difference of the left and right column sums:

$$m_{ij} = m_{ij}^c = \frac{1}{2} (S_{i+1} - S_{i-1}). \quad (22)$$

(3) The left column sum $S_{i-1}$ and center column sum $S_i$ are exact. In this case one uses the left-sided divided difference:

$$m_{ij} = m_{ij}^l = S_i - S_{i-1}. \quad (23)$$

(4) The center column $S_i$ and right column sum $S_{i+1}$ are exact. In this case one uses the right-sided divided difference:

$$m_{ij} = m_{ij}^r = S_{i+1} - S_i. \quad (24)$$

The various theorems and lemmas in [24; 25] prove that in each of the above cases the formulas in (20)–(24) result in a first-order accurate approximation $m_{ij}$ to the first derivative $g'(x_c)$ of the interface at the point $x_c = \frac{1}{2} (x_i + x_{i+1})$, as shown in Equation (18).

Next is a description of the algorithm that one uses to obtain the correct slope given only the volume fraction information in the $3 \times 5$ block of cells $\hat{B}_{ij}$.
The algorithm to choose the slopes. Once the $5 \times 5$ block $\tilde{B}_{ij}$ has been rotated as described in Section 3, one only uses the $3 \times 5$ portion $\hat{B}_{ij}$ of $\tilde{B}_{ij}$ to make the decision as to which of the cases listed above one uses for that particular cell $C_{ij}$. The algorithm for selecting the slope is as follows.

Case 1: (The center column sum $S_i$ is not exact, but is exact to $O(h)$.) First one checks the cell $C_{i,j+2}$. If $\Lambda_{i,j+2} > 0$, then the cell above the center column contains some material 1 and hence the center column sum $S_i$ is not exact. However, by Theorem 4 of [25] the condition on $h$ in (5) ensures that $S_i$ is exact to $O(h)$. Therefore, one next checks the left column sum $S_{i-1}$ and right column sum $S_{i+1}$ to determine which column sum is exact, and hence which difference one will use; that is, Case 1(a) or 1(b) from the list above. (The constraint in (5) will ensure that the column sums $S_{i-1}$ and $S_{i+1}$ are not both exact, but one of them will be.)

Case 1(a): If $\Lambda_{i-1,j+2} = 0$, the left column sum $S_{i-1}$ is exact, and hence one uses the left-sided difference:

$$m_{ij} = m^l_{ij} = S_i - S_{i-1}.$$  

Case 1(b): Otherwise, it must be the case that $\Lambda_{i+1,j+2} = 0$, and hence the right column sum is exact. Therefore, one uses the right-sided difference:

$$m_{ij} = m^r_{ij} = S_{i+1} - S_i.$$  

Case 2: (The center column sum must be exact.) Otherwise, $\Lambda_{i,j+2} = 0$, and hence the center column sum $S_i$ is exact. In this case one first checks to see which of the left and right column sums are exact. If both are exact, then one uses a centered difference. Otherwise one uses a one-sided difference with the center column and whichever of the left or right column sums is exact.

Case 2(a): If $\Lambda_{i-1,j+2} = 0$ and $\Lambda_{i+1,j+2} = 0$, then both the left column sum $S_{i-1}$ and the right column sum $S_{i+1}$ are exact. Therefore, one uses a centered difference, since it is one order more accurate than a one sided difference:

$$m_{ij} = m^c_{ij} = \frac{1}{2}(S_{i+1} - S_{i-1}).$$  

Case 2(b): If only $\Lambda_{i-1,j+2} = 0$, and hence the right column sum $S_{i+1}$ is not exact, then one uses the left-sided difference:

$$m_{ij} = m^l_{ij} = S_i - S_{i-1}.$$  

Case 2(c): Otherwise, if only $\Lambda_{i+1,j+2} = 0$, and hence the left column sum $S_{i-1}$ is not exact, then one uses the right-sided difference:

$$m_{ij} = m^r_{ij} = S_{i+1} - S_i.$$
**Step II.** Once the slope $m_{ij}$ has been found the constraint

$$\tilde{\Lambda}_{ij}(\tilde{g}) = \Lambda_{ij}(g),$$

where $\tilde{\Lambda}_{ij}(\tilde{g})$ denotes the fraction of material 1 in the $ij$-th cell due to $\tilde{g}$, immediately determines $b_{ij}$. In other words, once the $5 \times 5$ block of cells $\tilde{B}_{ij}$ has been appropriately rotated, $b_{ij}$ is a single valued function of $\tilde{\Lambda}_{ij}(\tilde{g})$ and $m_{ij}$:

$$b_{ij} = b_{ij}(\tilde{\Lambda}_{ij}(\tilde{g}), m_{ij}).$$

There are a variety of formulas one can employ to determine $b_{ij}$ given $m_{ij}$. For example, there is an approach that is based on representing the boundary of the portion of the cell $C_{ij}$ that contains material 1 by directed line segments and using the divergence theorem to compute the volume fraction $\tilde{\lambda}_{ij}$ developed by S. G. Roberts and used in [26]. There is the approach developed by J. S. Saltzman and used in [23] that is based on employing a coordinate system in which the approximate interface $\tilde{g}_{ij}(x)$ is given by

$$n_{ij}^x x + n_{ij}^y y = \sigma,$$

where $n_{ij} = (n_{ij}^x, n_{ij}^y)$ is the unit normal to $\tilde{g}_{ij}(x)$ that points away from the material 1 and $\sigma$ is the distance from $\tilde{g}_{ij}$ to $(x_i, y_j)$, the lower left hand corner of the cell $C_{ij}$. In work with Kothe et al. [8; 10; 32; 11], M. W. Williams developed algorithms for working on three-dimensional hexahedral and other unstructured meshes. Details of this work may also be found in Williams’ Ph.D. thesis [33]. Finally, an article by Scardovelli and Zaleski [29] describes a collection of formulas one may use on two and three dimensional rectangular grids.

The algorithm that was used to compute the computational example shown in Section 6 of this article is based on determining which of the three polygons the approximate interface $\tilde{g}_{ij}(x)$ forms when it passes through the cell $C_{ij}$:

1. a triangle,
2. the complement of a triangle in a square, and
3. a trapezoid.

In other words, material 1 is contained in a region that has the shape of one of the three polygons listed above.

Given the volume fraction $\Lambda_{ij}$ — and hence the volume $V_{ij}$ of material 1 in the $ij$-th cell — and the slope $m_{ij}$, one can write down algebraic formulas for each of these polygons. In this way the polygon with the correct volume is readily identified, and with it the point of intersection of the approximate interface $\tilde{g}_{ij}(x)$ with two of

---

As previously noted, strictly speaking one is given the area of material 1 in the $ij$-th cell. By convention this area is referred to as the volume of material 1 in the $ij$-th cell.
the four grid lines: \( x = x_i, x = x_{i+1}, y = y_j \) and \( y = y_{j+1} \) that form the edges of the cell \( C_{ij} \). Given this information, the y-intercept \( b_{ij} \) is easily found.

Note that several of the other algorithms for representing the approximate interface listed above allow one to design the method so that it is not necessary to rotate the \( 5 \times 5 \) block of cells \( \tilde{B}_{ij} \). However, the implementation of such an algorithm may be more complex than the one described here.

### 6. A computational example

Table 1 contains the max norm error from a computation in which the interface reconstruction algorithm described in this paper is used to approximate \( \cos x \) for \( 0 \leq x \leq \pi \) on square grids with cell sizes varying from \( 32^{-1} \) to \( 4096^{-1} \). The error reported in the table is computed according to the formula

\[
L^\infty \text{error} = \max \left\{ \max_{x \in [x_i, x_{i+1}]} | g(x) - \tilde{g}_{ij}(x) | \right\},
\]

where \( \max_{x \in [x_i, x_{i+1}]} | g(x) - \tilde{g}_{ij}(x) | \) is computed at 1000 points between the endpoints \( x_l \) (resp., \( x_r \)) at which the curve \( g(x) \) enters (resp., exits) the cell \( C_{ij} \) and the outer maximum in (25) is taken over all cells \( C_{ij} \) that satisfy \( 0 < \Delta x_{ij} < 1 \).

The third column contains the error (25) for the cell size reported in the second column. The fourth column contains the theoretical error bound from [25], which is quoted in Equation (2) (and also in Equation (26) below). Note that for all values of \( \Delta x \) the actual error is two orders of magnitude less than the theoretical error bound.

The last column of Table 1 contains the convergence rate. For a particular value of \( \Delta x = 2^{-k} \), the convergence rate is defined to be the rate at which the error would have to decrease in going from \( \Delta x = 2^{-(k-1)} \) to \( \Delta x = 2^{-k} \) in order to achieve the

<table>
<thead>
<tr>
<th>( k )</th>
<th>cell size ( \Delta x = 2^{-k} )</th>
<th>( L^\infty ) error</th>
<th>theoretical error bound</th>
<th>convergence rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>05</td>
<td>32(^{-1})</td>
<td>1.07( \cdot 10^{-4} )</td>
<td>8.43( \cdot 10^{-2} )</td>
<td>2.19</td>
</tr>
<tr>
<td>06</td>
<td>64(^{-1})</td>
<td>2.67( \cdot 10^{-5} )</td>
<td>2.11( \cdot 10^{-2} )</td>
<td>2.00</td>
</tr>
<tr>
<td>07</td>
<td>128(^{-1})</td>
<td>7.26( \cdot 10^{-6} )</td>
<td>5.27( \cdot 10^{-3} )</td>
<td>1.88</td>
</tr>
<tr>
<td>08</td>
<td>256(^{-1})</td>
<td>1.80( \cdot 10^{-6} )</td>
<td>1.32( \cdot 10^{-3} )</td>
<td>2.02</td>
</tr>
<tr>
<td>09</td>
<td>512(^{-1})</td>
<td>4.45( \cdot 10^{-7} )</td>
<td>3.29( \cdot 10^{-4} )</td>
<td>2.01</td>
</tr>
<tr>
<td>10</td>
<td>1024(^{-1})</td>
<td>1.12( \cdot 10^{-7} )</td>
<td>5.95( \cdot 10^{-5} )</td>
<td>1.99</td>
</tr>
<tr>
<td>11</td>
<td>2048(^{-1})</td>
<td>2.83( \cdot 10^{-8} )</td>
<td>2.06( \cdot 10^{-5} )</td>
<td>1.99</td>
</tr>
<tr>
<td>12</td>
<td>4096(^{-1})</td>
<td>7.13( \cdot 10^{-9} )</td>
<td>5.14( \cdot 10^{-6} )</td>
<td>1.99</td>
</tr>
</tbody>
</table>

**Table 1.** The max norm of the error from the computation of \( \cos x \) for \( 0 \leq x \leq \pi \).
error that is shown for \( \Delta x = 2^{-k} \). In other words,

\[
\text{convergence rate in the row with } \Delta x = 2^{-k} := \log_2 \left( \frac{\text{error}(2^{-(k-1)})}{\text{error}(2^{-k})} \right),
\]

where \( \text{error}(2^{-k}) \) denotes the error in the max norm when \( \Delta x = 2^{-k} \). It is apparent that the error in the max norm decreases at a rate commensurate with a method that is second-order accurate in the max norm as claimed.

### 7. Conclusions

The main result of [24; 25] is a proof that

\[
|g(x) - \tilde{g}_{ij}(x)| \leq \left( \frac{50}{3} \kappa_{\text{max}} + C_S \right) h^2 \quad \text{for all } x \in [x_i, x_{i+1}],
\]

where \( \tilde{g}_{ij}(x) \) is the volume-of-fluid approximation to the interface \( g(x) \) in the cell \( C_{ij} \) that is described in this paper, \( \kappa_{\text{max}} \) is the maximum curvature of the interface as defined in (4), \( x_i \) and \( x_{i+1} \) denote the left and right edges respectively of the cell \( C_{ij} \), \( h \) is the length of each side of the square grid cell \( C_{ij} \) and

\[
C_S = \frac{\sqrt{3}}{2} \left\{ (2\sqrt{2} - 1)4\sqrt{\kappa_{\text{max}}} + \left( 1 - \frac{7(1 + \sqrt{2})}{20} \right) 32 \frac{1}{3} (\sqrt{\kappa_{\text{max}}})^3 \right\}^2.
\]

The bound in (26) holds whenever the grid size \( h \) and the maximum value

\[
\kappa_{\text{max}} = \max_s |\kappa(s)|
\]

of the curvature \( \kappa(s) \) of the interface \( z(s) \) in the 3 \( \times \) 3 block of cells \( B_{ij} \) satisfies

\[
h \leq \frac{C_h}{\kappa_{\text{max}}}, \quad \text{where } C_h = \frac{1}{25}. \tag{27}
\]

However, in [24; 25] there are no examples of algorithms for finding the volume-of-fluid approximation \( \tilde{g}_{ij}(x) \) in each cell \( C_{ij} \) which contains a portion of the interface. This article contains a description of one such algorithm. This algorithm is new and has not appeared previously in the scientific literature. As shown in Table 1 the computations to approximate \( \cos x \) on the interval \([0, \pi]\) shown in Section 6 are consistent with the theoretical error bounds in [24; 25]. In other words, the computational approximation of \( \cos x \) made with this new algorithm is consistent with the claim that it is second order accurate in the max norm provided that the interface \( g \in C^2 \) and (27) is satisfied.

It may be possible for the interface to pass through the center cell, then exit the 3 \( \times \) 3 block \( B_{ij} \), wander around the computational domain, and reenter the 3 \( \times \) 3 block \( B_{ij} \) and the center cell \( C_{ij} \) again. For the purposes of this paper it is assumed that this does not happen. When implementing the algorithm described in this paper, one can design the code to automatically check for such cases and flag the 3 \( \times \) 3
block $B_{ij}$ for grid refinement, so that no cell contains two instances of the interface in a configuration such as the one just described.

In [21] J. E. Pilliod and Puckett described two volume-of-fluid interface reconstruction algorithms they had developed, and which they named the Least Squares Volume-of-Fluid Interface Reconstruction Algorithm (LVIRA) and the Efficient Least Squares Volume-of-Fluid Interface Reconstruction Algorithm (ELVIRA). They then presented computations with these algorithms on both $C^2$ and $C^0$ interfaces. When the underlying exact solution was a circle, the LVIRA and ELVIRA algorithms were shown to be second-order accurate in the max norm.

In all of the other computations they computed the errors in the averaged $l^1$ norm; that is, the $l^1$ norm averaged over 1000 random perturbations of the problem. For example, the $l^1$ error reported in approximating a circle was an average of the errors obtained when approximating 1000 different unit circles in which the center of the circle was chosen at random. They then compared the errors with errors obtained when they used several other widely used volume-of-fluid interface reconstruction algorithms such as SLIC [19] and the method developed by Parker and Youngs [20]. The only other algorithm that was close to being second-order accurate in the averaged $l^1$ norm consisted of taking the centered difference for the slope — that is, $m_{ij} = m_{ij}^c$, where $m_{ij}^c$ is defined in (16).

It is not clear whether the proofs in [24; 25] apply to the LVIRA and ELVIRA algorithms. Hence, it is not clear whether LVIRA and ELVIRA are second-order accurate in the max norm, or in the $l^1$ and $l^2$ norms when the errors are not averaged over many computations. Future work should include a study of this issue and a direct comparison between the algorithm presented here and the LVIRA and ELVIRA algorithms.

One should also note that both the LVIRA and ELVIRA algorithms, as well as the algorithm presented here, reconstruct lines exactly. It is an open problem to prove whether or not this is a sufficient condition for the algorithm to be second-order accurate when the underlying interface is $C^2$.

Corollary 22 in [24] states that the algorithm presented in this paper with slope $m_{ij} = 0$ (i.e., the piecewise constant or “stair-step” volume-of-fluid interface reconstruction algorithm) will be first-order accurate whenever the interface is $C^1$ rather than $C^2$. (See footnote 10 in [24] regarding how smooth the interface must be in order to prove Corollary 22.) Future work should include an exploration of how well the algorithm presented here approximates interfaces that are less than $C^2$.

Finally, when the interface reconstruction algorithm is coupled to an adaptive mesh refinement algorithm, the parameter

$$H_{\text{max}} = C_h (\kappa_{\text{max}})^{-1},$$

where $\kappa_{\text{max}}$ is the maximum curvature of the interface over the $3 \times 3$ block of cells.
$B_{ij}$ centered on a given cell $C_{ij}$, can be used to develop a criterion for determining when to increase the resolution of the grid. Namely, the computation of the interface in $C_{ij}$ is under-resolved whenever

$$h > H_{\text{max}},$$

and hence the grid needs to be refined in a neighborhood of the block $B_{ij}$.

References


Received February 23, 2010.

**Elbridge Gerry Puckett:** egpuckett@ucdavis.edu

*University of California, Davis, Department of Mathematics, One Shields Avenue, Davis, CA 95616, United States*

http://www.math.ucdavis.edu/
<table>
<thead>
<tr>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>On the accuracy of finite-volume schemes for fluctuating hydrodynamics</td>
<td>149</td>
</tr>
<tr>
<td>Aleksandar Donev, Eric Vanden-Eijnden, Alejandro Garcia and John Bell</td>
<td></td>
</tr>
<tr>
<td>A volume-of-fluid interface reconstruction algorithm that is second-order accurate in the max norm</td>
<td>199</td>
</tr>
<tr>
<td>Elbridge Gerry Puckett</td>
<td></td>
</tr>
<tr>
<td>Implicit particle filters for data assimilation</td>
<td>221</td>
</tr>
<tr>
<td>Alexandre Chorin, Matthias Morzfeld and Xuemin Tu</td>
<td></td>
</tr>
<tr>
<td>Parallel in time algorithms with reduction methods for solving chemical kinetics</td>
<td>241</td>
</tr>
<tr>
<td>Adel Blouza, Laurent Boudin and Sidi Mahmoud Kaber</td>
<td></td>
</tr>
<tr>
<td>A hybrid parareal spectral deferred corrections method</td>
<td>265</td>
</tr>
<tr>
<td>Michael L. Minion</td>
<td></td>
</tr>
</tbody>
</table>