A finite-difference based numerical method for the simulation of multiphase incompressible flows is presented. The proposed algorithm adapts the gradient-augmented level set method of Nave et al. [25] for use in multiphase fluid simulations by interpolating the fluid velocity and introducing a simple reinitialization procedure. The method is carried out only along a narrow band surrounding the interface, reducing computational effort while preserving the advantages of the gradient-augmented method. Numerical results show excellent agreement with an analytical solution and available experimental data. A new experimental benchmark is introduced, using data gathered in a wedge-driven wave tank. In addition to the optimally local advection scheme and sub-grid resolution afforded by the new method, numerical comparisons with the standard level set method reveal superior volume conservation.

© 2014 Elsevier Inc. All rights reserved.

1. Introduction

The gradient-augmented level set method developed in Nave et al. [25] improves the level set method by evolving both the level set function and its gradient values in a fully coupled fashion. The level set function is described by piecewise Hermite cubic polynomials on the cells of the computational domain, permitting the easy interpolation of derivative quantities at arbitrary points and allowing sub-grid resolution of the interface. By incorporating this method in incompressible flow simulations, our method inherits the following advantages: a cubically-defined level set function, reduced volume loss, richer sub-grid features, and an optimally local stencil (consisting of only a single grid cell). In the proposed method, the level set and gradient values are stored and advected only on a narrow band about the interface, reducing memory demands and computation time. Moreover, the level set function and its gradient are regularly reinitialized to a signed distance function, and the narrow band is rebuilt as the interface reaches its edge. The minimal volume loss exhibited by the method is easily corrected by a simple volume reinitialization procedure.

The level set method, originally introduced in [30], is a numerical technique for implicitly evolving an interface by advecting a scalar function that is one dimension higher than the interface of interest (see Fig. 1). By initializing the interface as the zero level set of this function (called the level set function), the interface propagates naturally as the zero level set evolves with the level set function. Some advantages of the level set method over other techniques include its ability to easily handle topology changes, the ease of implementation, and its relative simplicity. The level set method is an Eulerian-type method; therefore, it requires neither the tracking of an arbitrarily large number of marker particles nor any reconfiguration of the computational mesh to align with the interface; rather, values of the level set function are stored at cell centers and systematically updated at each time step.

This paper focuses primarily on the application of the level set method for modeling the free surface in multiphase flow. Specifically, our interest lies in the representation of the free surface between water and air for the purpose of modeling...
offshore ocean waves. Over the past several decades the level set method has been utilized for a variety of fluid flow applications, and continues to be a popular choice (see [24,25,38]).

One challenge of the level set method is ensuring that the level set function remains a signed distance function near the interface, thereby permitting accurate calculations of the normal vector, curvature, and Heaviside profile. Even when the level set function is initialized as a signed distance function, the evolution of the level set function does not guarantee that this quality is preserved, and a reinitialization procedure is required to regain this convenient quality. This task is typically accomplished via fast marching or fast sweeping methods, which have been developed to efficiently solve the Eikonal equation on the entire computational domain [35]. However, it has been shown that many simple reinitialization procedures result in non-physical motion of the interface (often in a direction opposed to the motion of the characteristics of the level set equation), resulting in the development of more sophisticated, high-order reinitialization procedures [9,22] to overcome this problem.

Conserving fluid mass is another challenge associated with level set implementations. Depending upon the demands of the application, it is common for simulations that include a level set formulation to exhibit non-physical losses of fluid mass as the computation progresses. For problems that demand a moderately large time scale, this volume loss can be detrimental to the quality of the simulation. A variety of methods exist to prevent either fluid phase from gaining or losing volume. One such method requires an additional reinitialization step that solves a perturbed Hamilton–Jacobi equation to steady state [8]. Another technique, known as the coupled level set volume-of-fluid (CLSVOF) method, borrows from the volume-of-fluid (VOF) method to conserve fluid mass during the advection of the level set [40]. A third approach, known as the particle level set method, borrows from Lagrangian methodologies that explicitly utilize characteristics to advect massless particles. Particles are seeded on either side of the zero contour of the level set and advected with the velocity field as the level set evolves. When particles cross the interface, representing a contradiction of the two methods, the level set function is altered to correct the change in mass represented by the crossing [29]. Another hybrid method utilizing both Lagrangian techniques and the level set method while achieving third order accuracy is presented in [28].

The standard level set technique and variations like the CLSVOF method are typically second-order accurate. They demand very large finite difference stencils for the calculation of derivative quantities [36] and cannot represent structures smaller than grid resolution. In Nave et al. [25], a globally third-order method is developed for advecting the level set function and its gradient information in a fully-coupled fashion, storing both values at the same location in each cell. The so-called gradient-augmented level set (GALS) method is optimally local in that only values on a single cell are required for the advection scheme, which is based on the methods of Courant et al. [13]. Sub-grid structure is represented to high accuracy since the level set function and its gradient are known at each node, permitting Hermite cubic interpolation over each rectangular grid cell (after the calculation of cross-derivatives). This enables the easy computation of derivative quantities like normal vectors and the curvature of the free surface. In addition to being higher order than standard level set methods, numerical experiments in [25] show that the GALS method also conserves volume much better than the standard level set method.

In this paper, we apply the gradient-augmented level set method to problems involving two-phase incompressible flow. Fluid velocity is interpolated in space and time, and a reinitialization procedure is introduced to maintain an approximate signed distance function. In the manner of Adalsteinsson [2], the method is implemented only along a narrow band near the interface, dramatically reducing computational cost. Numerical experiments reveal strong agreement with an analytical solution and experimental data, including comparison to a new experimental benchmark involving a wave tank. While the method is implemented and benchmarked in two spatial dimensions, it is intended to be easily extensible to three dimensions.

We note that the gradient augmented level set method, both in its original and narrow band implementation, exhibits third order global accuracy. In this paper it is paired with a lower order (first-order in time) Navier–Stokes solver that experiences no increase in order of accuracy by using a high order level set technique. There are several factors motivating this choice: First, the primary motivation of this paper is to demonstrate the feasibility of efficiently utilizing the semi-Lagrangian GALSM for multiphase Navier–Stokes flow; the results demonstrate that the GALSM is an excellent choice for this purpose and opens the door to future work with higher-order flow solvers. Second, the narrow-band gradient-augmented level set method proves to be a very efficient technique for evolving the interface (see the results in Section 6.8), representing less than 1% of the computation time when compared with solving for the fluid variables. Third, the GALSM provides a richer...
description of the interface as a cubic polynomial, enabling derivative information to be obtained with optimal locality at any point in the domain, an important property in the context of parallel computing. And finally, the GALSM method is superior to the classical level set method in preserving fluid volume (even though neither method is designed to be strictly conservative).

This paper is organized in the following way: A description of the fluid problem and our discretization scheme is given in Section 2. This is followed by a brief introduction to the standard level set method in Section 3 and a more detailed description of the gradient-augmented level set method in Section 4. Section 5 introduces the details of our implementation for multiphase flow applications. Section 6 shows the results of our numerical implementation, and concluding remarks are made in Section 7.

2. Fluid problem and discretization

We consider the evolution of a level set function $\phi(\vec{x})$ with a zero contour representing the free surface between two immiscible fluids. These fluids may have different densities and viscosities, but they are assumed to be incompressible. In the rectangular problem domain $B$, the motion of the two fluids is governed by the Navier–Stokes equations for incompressible flow, given by:

$$
\rho \left( \frac{\partial \vec{u}}{\partial t} + \vec{u} \cdot \nabla \vec{u} \right) = -\nabla p + \nabla \cdot 2\mu \vec{D} + \rho \vec{g}
$$

(1)

and

$$
\nabla \cdot \vec{u} = 0
$$

(2)

where $\rho$ represents the fluid density, $\mu$ represents the kinematic viscosity, $p$ is the fluid pressure, $\vec{u}$ represents the fluid velocity, $\vec{g}$ is the acceleration due to gravity, and the deformation matrix $\vec{D}$ is given by:

$$
\vec{D} = \frac{1}{2}(\nabla \vec{u} + \nabla^T \vec{u}).
$$

(3)

In the standard formulation of the level set method for multiphase flows [38], an additional term is added to (1) to account for the effects of surface tension, giving

$$
\rho \left( \frac{\partial \vec{u}}{\partial t} + \vec{u} \cdot \nabla \vec{u} \right) = -\nabla p + \nabla \cdot 2\mu \vec{D} + \rho \vec{g} + \sigma \kappa \delta(\phi) \vec{n}
$$

(4)

where $\sigma$ is the coefficient of surface tension, $\kappa$ is the curvature of the interface, $\delta$ is the Dirac delta function (ensuring that the surface tension is only applied at the location of the interface), and $\vec{n}$ is the surface normal vector. A Heaviside function is utilized to distinguish the fluid properties in each phase. For example, the fluid density and viscosity on the domain $B$ are given by

$$
\rho = \rho_1 H(\phi) + \rho_2 (1 - H(\phi))
$$

(5)

$$
\mu = \mu_1 H(\phi) + \mu_2 (1 - H(\phi))
$$

(6)

where the indices distinguish the properties of the two fluids. The general form of the Heaviside function is given by

$$
H(\phi) = \begin{cases} 
0 & \text{for } \phi < 0 \\
1 & \text{for } \phi \geq 0.
\end{cases}
$$

(7)

The rectangular computational domain is discretized as a staggered MAC grid, first introduced in [17]. As shown in Fig. 2, horizontal velocity components are stored on the left and right faces of each cell, vertical velocity components are stored on the top and bottom faces of each cell, and all scalar values (i.e. pressure, density, level set values, and Heaviside values) are stored in cell centers. Ghost values for these variables exist just outside the boundary, as shown in Fig. 2. The problem is discretized on a uniform grid with cells of dimension $h_x$ by $h_y$ and cell centers labeled with indices $i$ and $j$ in the $x$- and $y$-directions, respectively. Notice that cell centers are located at whole indices, whereas the faces of each cell have half indices. These index assignments are illustrated in Fig. 2.

The level set data (including the scalar level set values, the gradient values, and the cross-derivatives) are stored at cell centers, a minor departure from the original GALSM, which stored these values at cell vertices. This change is motivated by the convenience in our code of having all scalar values stored at cell centers. A more fundamental departure from the work of Nave et al. [25] is the need to interpolate the velocity field based upon the calculated velocity values at cell walls (see Fig. 2), as discussed in Section 5.1. In the original GALSM work, the velocity field was exactly prescribed everywhere in the computational domain.

The gradient-augmented level set method described in this paper is coupled with the stable, spatially second-order techniques utilized in Morgan [24] for discretizing and solving the two-phase incompressible Navier–Stokes equations. As
2.1. Chorin pressure projection scheme

The velocity and pressure are updated at each time step using the following scheme: Given the current velocity \( \vec{u}^n \) and pressure \( p^n \), we seek to advance the fluid velocity and pressure variables in time from \( t = n\Delta t \) to \( t = (n+1)\Delta t \), where the time step is given as \( \Delta t \). In the manner of the classic Chorin pressure projection scheme [11], an intermediate velocity is first calculated by rearranging the semi-discrete Navier–Stokes equation and neglecting the pressure term. The resulting intermediate velocity update is applied on the entire domain \( B \):

\[
\vec{u}^* = \vec{u}^n + \frac{\Delta t}{\rho} \left[ -\vec{u}^n \cdot \nabla \vec{u}^n + \nabla \cdot (2\mu D^n + \sigma K \delta(\phi) \vec{n}) + \rho \vec{g} \right].
\]

Next a pressure projection step is applied in a manner that ensures the updated velocity will be divergence-free. The updated pressure \( p^{n+1} \) is obtained by solving the following linear system:

\[
\frac{\Delta p^{n+1}}{\rho} = \nabla \cdot \vec{u}^*
\]

where \( \Delta \) on the left hand side represents the scalar Laplacian operator. Finally, the divergence-free, updated velocity is given by the update:

\[
\vec{u}^{n+1} = \vec{u}^* + \frac{\Delta t}{\rho} p^{n+1}.
\]

As noted in [24], this projection scheme is first-order accurate in time, yielding a first-order accurate solution to the fluid velocity and pressure in time.

2.2. Spatial discretization

The convective and viscous terms in Eq. (8) are discretized using a second-order upwinding scheme and a second-order central difference scheme, respectively. Written component-wise, the convective term has the form:

\[
\vec{u} \cdot \nabla \vec{u} = \left( u \frac{\partial u}{\partial x} + v \frac{\partial u}{\partial y} \right) \hat{x} + \left( u \frac{\partial v}{\partial x} + v \frac{\partial v}{\partial y} \right) \hat{y}.
\]
where upwinding is applied to each of the four terms in the same way. If we consider the cell face at \((m,n) = (i + \frac{1}{2}, j)\), the first term is discretized as
\[
\left[ u \frac{\partial u}{\partial x} \right]_{m,n}^{+} - u_{m,n}^{-} \frac{\partial u}{\partial x} \right]_{m,n}^{-} + u_{m,n}^{+} \left( \frac{\partial u}{\partial x} \right)_{m,n}^{+}
\]
where the velocity terms
\[
u_{m,n}^{+} = \frac{1}{2} (u_{m,n} + |u_{m,n}|) \]
\[
u_{m,n}^{-} = \frac{1}{2} (u_{m,n} - |u_{m,n}|)
\]
behave as a switch, choosing the appropriate upwinded derivative
\[
\left( \frac{\partial u}{\partial x} \right)_{m,n}^{+} = \frac{1}{2h_x} (3u_{m,n} - 4u_{m-1,n} + u_{m-2,n})
\]
\[
\left( \frac{\partial u}{\partial x} \right)_{m,n}^{-} = \frac{1}{2h_x} (-3u_{m,n} + 4u_{m+1,n} - u_{m+2,n}).
\]

The discretizations of the other three terms in Eq. (11) are analogous.

Written component-wise, the viscous term has the form:
\[
\nabla \cdot 2\mu \mathbf{D} = \left[ \frac{\partial}{\partial x} \left( 2\mu \frac{\partial u}{\partial x} \right) + \frac{\partial}{\partial y} \left( \mu \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \right) \right] \hat{x} + \left[ \frac{\partial}{\partial x} \left( \mu \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} \right) + \frac{\partial}{\partial y} \left( 2\mu \frac{\partial v}{\partial y} \right) \right] \hat{y}.
\]

This form of the viscous term is chosen in order to account for the variable viscosity in multiphase flow and the permissibility of flow that is not divergence-free in the algorithmic step prior to the pressure correction. If we again consider the cell face at \((m,n) = (i + \frac{1}{2}, j)\), the first term is discretized as
\[
\left[ \frac{\partial}{\partial x} \left( 2\mu \frac{\partial u}{\partial x} \right) \right]_{m,n}^{+} = \frac{1}{h_x} \left[ 2\mu_{R} \left( \frac{\partial u}{\partial x} \right)_{R} - 2\mu_{L} \left( \frac{\partial u}{\partial x} \right)_{L} \right]
\]
where the viscosity terms are those given at the cell centers
\[
\mu_{R} = \mu_{m+1/2,n} = \mu_{i+1,j} \]
\[
\mu_{L} = \mu_{m-1/2,n} = \mu_{i,j}
\]
and the partial derivatives are given by central differences
\[
\left( \frac{\partial u}{\partial x} \right)_{R} = \frac{1}{h_x} (u_{m+1,n} - u_{m,n})
\]
\[
\left( \frac{\partial u}{\partial x} \right)_{L} = \frac{1}{h_x} (u_{m,n} - u_{m-1,n}).
\]

The discretizations of the other terms in Eq. (17) are analogous.

The following expressions are valid for a level set function \(\phi\) that approximates a signed distance function from the interface (see Sections 3 and 4 for more on the level set function). The smeared Heaviside function (7) is
\[
H(\phi) = \begin{cases} 
0 & \text{for } \phi < -\epsilon \\
1 & \text{for } \phi > \epsilon \\
\frac{1}{2} (1 + \frac{\phi}{\epsilon} + \frac{1}{\epsilon^2} \sin \left( \frac{\pi \phi}{\epsilon} \right)) & \text{otherwise}
\end{cases}
\]
where \(\epsilon\) is typically set to \(1.5h\) and \(h = \max(h_x, h_y)\), though \(h_x\) and \(h_y\) are often identical in these approaches [29,38,40]. This guarantees that the interface width is smeared over approximately 3 cells. (Notice that the transitional region exists on a narrow band near the interface.) Similarly, the Dirac delta function is given by
\[
\delta(\phi) = \begin{cases} 
0 & \text{for } \phi < -\epsilon \\
0 & \text{for } \phi > \epsilon \\
\frac{1}{2} \left( \frac{1}{\epsilon} + \frac{1}{\epsilon} \cos \left( \frac{\pi \phi}{\epsilon} \right) \right) & \text{otherwise}
\end{cases}
\]
The normal vector to the interface is given by
\[ \vec{n} = \frac{\nabla \phi(x)}{|\nabla \phi(x)|} \]  
(25)

and the curvature on the interface is given by

\[ \kappa = \nabla \cdot \vec{n} = \frac{\phi_{xx} \phi_y^2 - 2 \phi_x \phi_y \phi_{xy} + \phi_{yy} \phi_x^2}{|\nabla \phi|^3}. \]  
(26)

As noted in [29], the use of smeared Heaviside and Delta functions renders the scheme to spatial first-order accuracy near the interface. In turn, the approximations to the density and viscosity are also first-order accurate near the interface. Accordingly, we expect the discretized viscous term \( \nabla \cdot 2 \mu \vec{D} \) to exhibit first-order accuracy in the vicinity of the interface instead of the second-order accuracy expected elsewhere in the domain.

2.3. Time step restrictions

The time step restrictions associated with this discretization involve a convective limit

\[ \Delta t_{\text{conv}} = \left( \frac{u_{\text{max}}}{h_x} + \frac{v_{\text{max}}}{h_y} \right)^{-1}, \]  
(27)

a viscous limit

\[ \Delta t_{\text{visc}} = \left[ \left( \frac{\mu}{\rho} \right)_{\text{max}} \left( \frac{1}{h_x^2} + \frac{1}{h_y^2} \right) \right]^{-1}, \]  
(28)

and a surface tension limit

\[ \Delta t_{\text{surf}} = \sqrt{\frac{1}{\rho_{\text{min}} \min(h_x, h_y)^3} \frac{3}{4 \sigma}}, \]  
(29)

where \( \mu \) and \( \rho \) in Eq. (28) are the viscosity and density of the same fluid phase, and \( u_{\text{max}} \) and \( v_{\text{max}} \) are the absolute values of the velocity components with the largest magnitudes on the computational domain. The time step is given by

\[ \Delta t = s \min(\Delta t_{\text{conv}}, \Delta t_{\text{visc}}, \Delta t_{\text{surf}}) \]  
(30)

where \( \Delta t_{\text{surf}} \) is only considered when \( \sigma \neq 0 \), and \( s \leq 1 \) is reduced as necessary to ensure stability in a given problem. Typically, we choose \( s \) between 0.5 and 0.8.

2.4. Solution summary

The following steps are conducted at every time step to advance the solution of the velocity field, pressure field, and level set function forward in time:

1. Use Eq. (8) to calculate the intermediate velocity field \( \vec{u}^* \) from \( \vec{u}^n \) and the discretizations of the convective and viscous terms found in Section 2.2.
2. Calculate the new pressure field \( p^{n+1} \) by solving the linear system (9) with \( \vec{u}^* \) on the right hand side.
3. Advance the velocity field from \( \vec{u}^* \) to \( \vec{u}^{n+1} \) using the update Eq. (10) and the new pressure field \( p^{n+1} \).
4. Evolve the level set function using the updated velocity \( \vec{u}^{n+1} \) (see Section 5.3 for details).
5. Update the Heaviside function using the new level set values. This automatically updates the fluid density and viscosity via Eqs. (5) and (6), respectively.
6. Return to Step 1.

3. The level set method

In the standard level set method, the location of an interface is represented by the zero contour of a scalar function \( \phi \) that is one spatial degree higher than the interface being represented. Values of this level set function are stored at nodes on the discretized domain, and the partitions of the domain interior and exterior to the physical surface have oppositely signed values of \( \phi \). By convention, the interior domain is located wherever \( \phi(x) > 0 \) and the region outside the surface has values \( \phi(x) < 0 \). Since only scalar values of the level set function are known in a given region, bi-linear interpolation is utilized in 2D (or tri-linear in 3D) to find the implicit zero contour, when necessary.

The level set function \( \phi \) is advected by the velocity field \( \vec{v}(x, t) \) according to the level set equation, given by

\[ \phi_t + \vec{v} \cdot \nabla \phi = 0, \]  
(31)
where \( \phi_t \) is the derivative of \( \phi \) with respect to time. A typical implementation of the standard level set method requires the calculation of derivative quantities via large upwind finite difference stencils. Essentially non-oscillatory (ENO) and WENO schemes are typically utilized for their high accuracy and reliability at shocks and discontinuities [36], but these methods require very large finite difference stencils and special treatment at boundaries. Runge–Kutta methods are typically utilized to advance the computation in time, producing globally second order methods [38].

As previously mentioned, it is preferable to guarantee that the level set equation is a signed distance function at all times, thereby satisfying the Eikonal equation

\[
|\nabla \phi(\tilde{x})| = 1.
\]  

(32)

Since the level set function will not necessarily preserve this property as it is advected, it is necessary to reinitialize the level set function at regular time intervals. A variety of methods, including the fast marching methods and fast sweeping methods, have been developed to iteratively satisfy the reinitialization equation

\[
\phi_t = \text{sign}(\phi_0)(1 - |\nabla \phi(\tilde{x})|),
\]  

(33)

which ensures that Eq. (32) is satisfied. For the level set function at a given time step, Eq. (33) represents the evolution of the level set function towards a signed distance function in pseudo-time. Eq. (33) is a nonlinear hyperbolic equation and may be written as

\[
\phi_t + \vec{w} \cdot \nabla \phi = \text{sign}(\phi_0)
\]  

(34)

where \( \vec{w} = \text{sign}(\phi_0)(\nabla \phi / |\nabla \phi|) \) are the characteristics, which point outward from the zero contour of the level set function. Fast marching and sweeping methods evolve this hyperbolic equation to steady-state, at which point the level set function becomes a signed distance function while approximately preserving the zero contour [38].

4. The gradient-augmented level set method

4.1. Evolving the gradient

The GALS method improves upon the standard level set method by storing and evolving the gradient values of \( \phi \) at the same nodes where \( \phi \) is stored. Defining the gradient as \( \hat{\psi} = \nabla \phi \), its evolution equation can be derived by applying the gradient operator and the product rule to Eq. (31):

\[
\hat{\psi}_t + \vec{v} \cdot \nabla \hat{\psi} = -\nabla \vec{v} \cdot \hat{\psi},
\]  

(35)

where we may represent the velocity component-wise as \( \vec{v} = u\hat{x} + v\hat{y} \), and \( \nabla \vec{v} \) is the velocity deformation matrix defined by

\[
\nabla \vec{v} = \begin{pmatrix} u_x & v_x \\ u_y & v_y \end{pmatrix}.
\]  

(36)

Subscripts imply differentiation with respect to the given spatial dimension and \( \hat{x} \) and \( \hat{y} \) are unit vectors. The level set function and its gradient are evolved by solving Eqs. (31) and (35) simultaneously.

4.2. Hermite cubic interpolation

While Eqs. (31) and (35) evolve the level set function and its first derivatives, respectively, neither evolves the cross-derivative \( \phi_{xy} \), which is required for calculating the cubic interpolant of \( \phi \). While there are methods available that calculate this cross-derivative with optimal locality (i.e., using only the nodes of a single cell), a simpler choice is the second-order approximation given by:

\[
\phi_{xy} = \frac{\phi_{y}^{i+1,j} - \phi_{y}^{i-1,j}}{2h_x}.
\]  

(37)

Despite the relatively low order of this approximation, the global third-order accuracy of the method is not affected. Given that the Hermite cubic approximation to \( \phi \) is a fourth-order approximation, a second-order approximation of the second derivative \( \phi_{xy} \) will suffice. A thorough error analysis is given in [25]. In [34], strategies for evolving higher derivatives, including the cross-derivative, are explored.

With the values \( \phi, \phi_x, \phi_y, \) and \( \phi_{xy} \) known at every node on the domain, it is possible to perform Hermite cubic interpolation over every individual rectangular cell on the domain. The interpolant over a given cell is a linear combination of 16 basis functions, where each basis function is nontrivial for only one derivative on exactly one corner of the cell. On a two-dimensional cell with its bottom-left corner at the origin and side lengths \( h = h_x = h_y \), this polynomial is expressed by

\[
H(x) = \sum_{\bar{a} \in [0,1]^2} \phi_{\bar{a}} h^{(\alpha_1 + \alpha_2)} W_{\bar{a}} \left( \frac{x}{h} \right).
\]  

(38)
where $\vec{s} = (i, j)$ represents the local node indices and $\alpha$ represents the degree of each spatial derivative (e.g., $\alpha = (0, 1)$ implies differentiation with respect to $y$ only). $W(\vec{x})$ represents the aforementioned basis functions, which are given by

$$W_{\alpha}(\vec{x}) = w_{\alpha_1}(x_1)w_{\alpha_2}(x_2),$$

where

$$w_{\alpha}(x) = \begin{cases} f(x), & \text{if } s = 0, \alpha = 0 \\ f(1 - x), & \text{if } s = 1, \alpha = 0 \\ g(x), & \text{if } s = 0, \alpha = 1 \\ -g(1 - x), & \text{if } s = 1, \alpha = 1, \end{cases} \tag{40}$$

and

$$f(x) = 1 - 3x^2 + 2x^3 \quad \text{and} \quad g(x) = x(1 - x)^2. \tag{41}$$

This interpolant provides sub-grid resolution of the zero contour and much greater accuracy than the standard bi-linear (2D) of tri-linear (3D) interpolation obtained from the standard level set method. Differentiation of this interpolating polynomial is straightforward, allowing the simple computation of derivative quantities of $\phi$, such as curvature and normal vectors.

### 4.3. The generalized CIR scheme

In order to advect the level set function and its gradient, we consider the two level set equations together:

$$\phi_t + \vec{v} \cdot \nabla \phi = 0$$
$$\vec{\psi}_t + \vec{v} \cdot \nabla \vec{\psi} = -\nabla \vec{v} \cdot \vec{\psi}. \tag{39}$$

Observe that the left sides of each equation are the material derivatives, respectively, of the scalar field $\phi(\vec{x}, t)$ and the vector field $\vec{\psi}(\vec{x}, t)$. As such, they represent the derivatives of each quantity taken along a curve that moves with velocity $\vec{v}(\vec{x}, t)$. We refer to the left side of each equation as the advection portion and the right side as the source term. The equations may be reformulated in their characteristic forms along the characteristic curve $\frac{d\vec{x}}{dt} = \vec{v}(\vec{x}, t)$ as

$$\frac{d\phi}{dt} = 0 \tag{42}$$
$$\frac{d\vec{\psi}}{dt} = -\nabla \vec{v} \cdot \vec{\psi}. \tag{43}$$

The task of solving Eqs. (31) and (35) may then be divided into two separate steps. Since Eqs. (42) and (43) describe the change of the level set function $\phi$ and its gradient $\vec{\psi}$ along the curve $\frac{d\vec{x}}{dt} = \vec{v}(\vec{x}, t)$, the first task is to determine the displacement of $\vec{x}$ along this path. Next, updated nodal values of $\phi$ and $\vec{\psi}$ are found by solving the ODEs (42) and (43). This is accomplished using a generalization of the CIR (Courant, Isaacson, Reese) method developed in [13], as outlined here.

Given a particular node $\vec{x}_j$, we define $\phi^n$ and $\vec{\psi}^n$ as the values of these functions at time $t$ and $\phi^{n+1}$ and $\vec{\psi}^{n+1}$ as their values at time $t + \Delta t$. In addition, let $\vec{x}_j(\tau)$ represent the position traced backwards along the characteristic curve from time $t + \Delta t$ to time $t$ (see Fig. 3). The following numerical strategy must be executed for every node in the domain:

The first step locates the point $\vec{x}_j(\tau)$ by numerically solving the equation of the characteristic curve

$$\frac{d}{d\tau}\vec{x}(\tau) = \vec{v}(\vec{x}(\tau), \tau) \tag{44}$$

backwards from $\tau = t + \Delta t$ to $\tau = t$. 

---

**Fig. 3.** The path of the characteristic curve is traced backwards in time from the node $\vec{x}_j$ to an interior point $\vec{x}_j$. Level set values at $\vec{x}_j$ are found using the Hermite cubic polynomial.
Next, Eqs. (42) and (43) are satisfied in order to find the new values of \( \phi \) and \( \vec{\psi} \), respectively. This is simple for \( \phi \), since, by Eq. (42), the value of \( \phi \) does not change as it moves along the characteristic curve. Therefore, we simply assign \( \phi^{n+1}(\vec{x}_j) = \phi(\vec{x}_j) \) (using the Hermite cubic interpolant), and the level set equation is evolved.

The gradient is evolved by solving Eq. (43) numerically. We rewrite the equation as

\[
\frac{\partial}{\partial t} \vec{\psi}(\tau) = -\nabla \vec{v}(\vec{x}(\tau), \tau) \cdot \vec{\psi}(\tau), \quad \text{where} \quad \vec{\psi}(t) = \vec{\psi}(\vec{x}),
\]

and solve forwards from \( \tau = t \) to \( \tau = t + \Delta t \). The solution \( \vec{\psi}(t + \Delta t) \) is assigned to \( \vec{\psi}^{n+1}(\vec{x}_j) \), and the gradient of the level set is evolved.

4.4. The superconsistent Shu–Osher RK3 scheme

In [25], the authors recommend the following Runge–Kutta scheme for solving the advection problem posed in Section 4.3. The so-called superconsistent Shu–Osher RK3 scheme is given as:

\[
\begin{align*}
\bar{x}_1 &= \bar{x} - \Delta t \vec{v}(\bar{x}, t + \Delta t) \\
\nabla \bar{x}_1 &= \nabla \vec{v}(\bar{x}, t + \Delta t) \\
\bar{x}_2 &= \bar{x} - \Delta t \left( \frac{1}{4} \vec{v}(\bar{x}, t + \Delta t) + \frac{1}{4} \vec{v}(\bar{x}_1, t) \right) \\
\nabla \bar{x}_2 &= \nabla \vec{v}(\bar{x}_2, t + \Delta t) + \frac{1}{6} \vec{v}(\bar{x}_1, t) \cdot \nabla \vec{v}(\bar{x}_1, t) \\
\bar{x} &= \bar{x} - \Delta t \left( \frac{1}{6} \vec{v}(\bar{x}, t + \Delta t) + \frac{1}{6} \vec{v}(\bar{x}_1, t) + \frac{2}{3} \vec{v}(\bar{x}_2, t + \frac{1}{2} \Delta t) \right) \\
\nabla \bar{x} &= \nabla \vec{v}(\bar{x}, t + \Delta t) + \frac{1}{6} \nabla \vec{v}(\bar{x}_1, t) \cdot \nabla \vec{v}(\bar{x}_1, t) + \frac{2}{3} \nabla \bar{x}_2 \cdot \nabla \vec{v}(\bar{x}_2, t + \frac{1}{2} \Delta t) \\
\phi(\bar{x}, t + \Delta t) &= \phi(\bar{x}, t) \\
\vec{\psi}(\bar{x}, t + \Delta t) &= \nabla \phi(\bar{x}) \cdot \vec{\psi}(\bar{x}, t)
\end{align*}
\]

where \( \bar{x} \) is the position of a particular node and \( \bar{x} \) is the point inside the cell where interpolation occurs. Notice that the final two steps utilize the Hermite cubic polynomial defined in Section 4.2. It should be noted that the convective time step restriction in Section 2.3 ensures that \( \Delta t \) is small enough that the interpolation point \( \bar{x} \) lies within a cell that is adjacent to the node at \( \bar{x} \) (as in Fig. 3). This is convenient in the context of a narrow band implementation since it places an upper bound on the displacement of the zero level set in a single time step, but it should be noted that no inherent CFL condition is required for the GALS to be stable [34].

This Runge–Kutta scheme is fourth-order accurate in time locally and third-order accurate globally [25].

5. Adapting the GALS method for flow problems

This section contains our modifications to the gradient-augmented level set (GALS) method that make it suitable for use in two-phase incompressible flow models.

5.1. Velocity interpolation

The GALS method as developed in Nave et al. [25] takes the value of the velocity field and the velocity deformation matrix for granted on the entire domain. Notice that the Runge–Kutta scheme presented in Section 4.4 requires the velocity to be known at arbitrary points in space and at half time steps. In computational fluid models in general, the velocity field is known only at a finite number of points on the mesh, and the derivatives of the velocity field components must be computed from these values.

Using simple Lagrange polynomial interpolation on each cell (or over multiple cells), the values of the velocity field and its derivatives may be approximated on non-nodal points. Recall that the velocity field \( \vec{v} \) is known by its x-component \( u \) and y-component \( v \) on the staggered nodes of the domain described in Section 2. If we consider only one velocity component (say \( u \)) and some rectangular subdomain of the mesh shown in Fig. 2 having \( n \) nodes at each corner, this subdomain will contain \( n \) nodes in the x-direction and \( m \) nodes in the y-direction. The value of \( u \) at any arbitrary point in this subdomain may be interpolated using the known values of \( u \) at the nodes via:
\[ u(x, y) = \sum_{i=1}^{n} \sum_{j=1}^{m} u(x_i, y) \tilde{l}_i(x) \tilde{l}_j(y) \]  
(46)

where
\[ l_i(x) = \prod_{j \neq i}^{n} \frac{x - x_j}{x_i - x_j} \quad \text{and} \quad \tilde{l}_i(y) = \prod_{j \neq i}^{m} \frac{y - y_j}{y_i - y_j}. \]  
(47)

The four derivative quantities in Eq. (36) are found over the subdomain by differentiating Eq. (46) with respect to the appropriate variable. For \( u \), these derivatives take the form
\[ \frac{\partial u}{\partial x} = \sum_{i=1}^{n} \sum_{j=1}^{m} u(x_i, y) \tilde{l}_i'(x) \tilde{l}_j(y) \]  
(48)

and
\[ \frac{\partial u}{\partial y} = \sum_{i=1}^{n} \sum_{j=1}^{m} u(x_i, y) \tilde{l}_i(x) \tilde{l}_j'(y) \]  
(49)

where the product rule gives
\[ \tilde{l}_i'(x) = \left( \prod_{j \neq i}^{n} \frac{1}{x_i - x_j} \right) \sum_{k \neq i} \left( \prod_{j \neq k}^{n} x - x_j \right) \]  
(50)

and
\[ \tilde{l}_j'(y) = \left( \prod_{j \neq i}^{m} \frac{1}{y_i - y_j} \right) \sum_{k \neq i} \left( \prod_{j \neq k}^{m} y - y_j \right). \]  
(51)

We emphasize that only nodes containing values of the \( x \)-velocity component \( u \) are used to interpolate \( u \), and only nodes containing values of the \( y \)-velocity component \( v \) are used to interpolate \( v \). The interpolant of \( v \) has exactly the same form as that presented for \( u \).

One risk in applying this method is the possibility of introducing numerical oscillations; however, this risk is greatest when using interpolating polynomials of a high degree. In our trials, using only 4 velocity nodes for a given velocity component was sufficient and resulted in no instability. The chosen points form the smallest possible rectangle containing the point of interest. This results in a bilinear interpolant of the velocity component \( (m = 2 \text{ and } n = 2) \). Moreover, the locality of the scheme is preserved by only using four velocity nodes for the interpolation. If desired, one may instead use a larger \((3 \times 3)\) rectangle that encloses the node, giving a biquadratic interpolant \((m = 3 \text{ and } n = 3)\). The half time velocities in the Runge–Kutta scheme of Section 4.4 are found by averaging the velocity at the current and previous time steps.

5.2. Reinitialization on a narrow band

We began this study by incorporating the gradient-augmented level set method of Nave et al. [25] into two-phase flow simulations in its original form (without reinitialization). The resulting solutions became very unstable due to strong distortion of the level set function in flow simulations. In order to prevent this phenomenon, we incorporate a reinitialization procedure that reconstructs the level set function as a signed distance function while simultaneously assigning appropriate gradient values.

Since advection and reinitialization of the level set is generally a computationally expensive procedure, we limit the domain of interest to a narrow band of nodes near the zero contour, in the manner of Adalsteinsson [2]. Since only a very small subset of nodes requires reinitialization, we explicitly calculate the distance of each node from the interface using the orthogonal distance fitting procedures of Ahn et al. [3]. This procedure involves the iterative solution of the following nonlinear system of equations, where \( \vec{x} \) represents the interface location nearest to the node at location \( \vec{x}_i \):

\[ \phi(\vec{x}) = 0 \]
\[ \nabla \phi(\vec{x}) \times (\vec{x} - \vec{x}_i) = 0. \]  
(52)

For a sufficiently close initial guess, this equation is satisfied to a very small tolerance within only a few Newton–Raphson iterations for each node. The Hermite cubic interpolant of the level set function is used at each iteration to update the level set function and gradient values on left side of the system.
The system (52) has been used extensively for the calculation of the signed distance from an implicit zero contour (see [3,10,37]). It is well known that the shortest distance between two points is a line. Moreover, the line connecting a point in space to the nearest point on a smooth surface is parallel to the normal vector on the surface at that point. System (52) takes advantage of these two facts to solve for the nearest point \( \vec{x} \) on a surface from a given point \( \vec{x}_i \). The length of the line segment connecting these 2 points is taken to be the (signed) distance. Since the direction of steepest ascent (or descent) of the signed distance function from \( x_i \) is in the direction of \( x \) (and is parallel to the normal vector at \( x \) ), this direction (or the opposite direction, depending on the sign) is chosen as the gradient of the signed distance function. This assigned gradient value is normalized via Eq. (32), since it is a property of signed distance functions that the magnitude of the gradient be unity. After this procedure is completed on all nodes, the original interface (the zero contour) is retained, all level set values represent the distance to the interface, and the Eikonal equation (32) is satisfied. See Section 5.2.2 for a thorough description of this procedure.

We note that a similar reinitialization procedure for the gradient-augmented level set method is implemented in [5]. Both procedures solve system (52) on a node-by-node basis, and both procedures utilize the Newton iteration scheme in [10]. However, the reinitialization procedure in [5] is a “hybrid” method that also utilizes a marching procedure to define the level set function on the entire domain rather than on a narrow band alone.

5.2.1. Building the narrow band

The narrow band consists of individual bands of nodes and is built from the interface outward. As an example, we define three bands (i.e. sets of nodes) \( N_1, N_2, N_3 \), which correspond to the first, second, and third bands, respectively. We add nodes to each set in the following order:

1. Nodes \( \vec{x}_i \) bordering a cell that contains the interface are added to the set \( N_1 \).
2. Nodes \( \vec{x}_i \not\in N_1 \) bordering a cell that contains a node \( \vec{x}_j \in N_1 \) are added to the set \( N_2 \).
3. Nodes \( \vec{x}_i \not\in N_1 \cup N_2 \) bordering a cell that contains a node \( \vec{x}_j \in N_2 \) are added to the set \( N_3 \).

In this manner, the narrow band is built in a layer-by-layer fashion, starting at nodes that surround the interface. See Fig. 4 for an illustration of the classification scheme.

In this work, for simplicity we employ a naive \( O(n^2) \) algorithm for building the narrow band (where \( n = 1/h \), assuming \( h = h_x = h_y \) ); we sweep over the entire domain once for each band. A more sophisticated approach might consist of an initial \( O(n^2) \) sweep to find the first band \( N_1 \), followed by an efficient procedure for adding the additional bands. For example, each band could be added by searching only the neighbors of the outermost band (an \( O(n) \) procedure) using a queue structure for each band. Another alternative is to leverage the heap sort technology commonly used for fast marching methods; see [10] and [29] for a discussion of these methods.

We remark that it is impossible to reinitialize the interface on the nodes in \( N_1 \) in a manner that guarantees preservation of the interface position due to the limited number of shape functions (16 shape functions in 2D) representing the level set function on each cell. As a result, the interface is artificially perturbed every time the reinitialization procedure is
conducted. Moreover, we note that it can be difficult to explicitly detect subgrid structure between nodes when determining which nodes should be placed in the set \( N_1 \). However, if reinitialization is conducted frequently, then the level set function will not stray grossly from a signed distance function; as a result, we choose nodes \( \tilde{x}_i \) with the property |\( \phi(\tilde{x}_i) \)| ≤ \( \sqrt{h_x^2 + h_y^2} \) to be in \( N_1 \) since this value implies that the interface is inside a cell neighboring the node \( \tilde{x}_i \).

We also note that the number of bands \( w \) is arbitrary. A smaller number of bands will require more frequent reinitialization (and a resulting decline in accuracy) since the moving interface will reach the edge of the narrow band more quickly (see Section 5.3.1), but a larger number of bands is more computationally demanding. We have found that setting \( w = 3 \) gives stable results, but the interface position is subtly perturbed at almost every time step. Using \( w < 3 \) is not recommended, since the smeared Heaviside function is spread over approximately 3 cells about the interface, and its definition requires a smoothly varying \( \phi \) value in this region. Setting \( w = 6 \), for example, provides a more accurate result (due to less frequent reinitialization) while still limiting the size of the computational domain.

### 5.2.2. Node-wise reinitialization procedure

Having outlined the steps required to build the narrow band, we now describe the implementation of the reinitialization procedure. Let \( \phi_0 \) and \( \psi_0 \) be the Hermite cubic function and its gradient specified by the nodal values prior to reinitialization, and let \( \phi_R \) and \( \psi_R \) represent these values after reinitialization. Given a node \( \tilde{x}_i \), we seek to solve the system (52), which will provide the nearest point on the interface to \( \tilde{x}_i \).

This procedure is first conducted on the band \( N_1 \) and then conducted on all other nodes in the narrow band. For nodes in the band \( N_1 \), the initial guess is given by

\[
\tilde{x}_i^0 = \tilde{x}_i - \alpha \frac{\psi_0(\tilde{x}_i)}{|\psi_0(\tilde{x}_i)|}
\]  

(53)

which represents a linear approximation of the interface’s position relative to node \( \tilde{x}_i \), where \( \alpha \) is an arbitrary parameter (0.9 in our case). For each node \( \tilde{x}_i \) in the band \( N_w \) where \( w \neq 1 \), we sweep over all nodes in the square of node-dimension \((2w + 1)^2\) centered at node \( x_i \). Of the nodes \( y_j \) in this square that are in \( N_1 \), we calculate the values

\[
\tilde{y}_j^0 = \tilde{y}_j - \alpha \frac{\psi_R(\tilde{y}_j)}{|\psi_R(\tilde{y}_j)|}
\]  

(54)

and choose the initial guess \( \tilde{x}_i^0 = \tilde{y}_j^0 \) for the value of \( \tilde{y}_j^0 \) which minimizes \( |\tilde{x}_i - \tilde{y}_j^0| \). This represents a linear approximation of the closest point on the interface to the node \( \tilde{x}_i \), based on the reinitialized level set values in the band \( N_1 \).

Given the initial value at a node, we seek to iterate towards a solution \( \tilde{x} \) to the system (52). One Newton iteration strategy involves repeatedly applying the step:

\[
\Delta \tilde{x} = -J^{-1}(\tilde{x}) \tilde{f}(\tilde{x})
\]  

(55)

\[
\tilde{x}^{k+1} = \tilde{x}^k + \alpha \Delta \tilde{x}
\]  

(56)

where

\[
J(\tilde{x}) = \begin{bmatrix}
\phi_x(\tilde{x}) & \phi_y(\tilde{x}) \\
\phi_{xx}(\tilde{x})(y - y_i) - \phi_{xy}(x - x_i) & \phi_{yy}(y - y_i) - \phi_{yx}(x - x_i) + \phi_x
\end{bmatrix}
\]  

(57)

and

\[
\tilde{f}(\tilde{x}) = \begin{bmatrix}
\phi(\tilde{x}) \\
\nabla \phi(\tilde{x}) \times (\tilde{x} - \tilde{x}_i)
\end{bmatrix}
\]  

(58)

until the value of \(|\Delta \tilde{x}|\) falls below a certain tolerance \((10^{-3}h_xh_y)\) is an appropriate choice. Note that all level set values and derivatives are obtained using the Hermite cubic polynomial representation of the level set function. This method is based on the Taylor expansion of \( f(\tilde{x}) \) and works well in this application. Another method, proposed by Chopp [10] involves 2 separate steps:

\[
\Delta \tilde{x}_1 = -\phi(\tilde{x}_1) \frac{\nabla \phi(\tilde{x}_1)}{|\nabla \phi(\tilde{x}_1)|^2}
\]  

(59)

\[
\Delta \tilde{x}_2 = (\tilde{x}_i - \tilde{x}_1) - \nabla \phi(\tilde{x}_1) \frac{(\tilde{x}_i - \tilde{x}_1) \cdot \nabla \phi(\tilde{x}_1)}{|\nabla \phi(\tilde{x}_1)|^2}
\]  

(60)

\[
\tilde{x}^{k+1} = \tilde{x}_1 + \alpha \Delta \tilde{x}_1 + \alpha \Delta \tilde{x}_2
\]  

(61)

where iteration is complete when \( \sqrt{\Delta \tilde{x}_1^2 + \Delta \tilde{x}_2^2} \) falls below a certain tolerance (as before). In either case, once the method has converged to a solution \( \tilde{x}_i \), the reinitialized nodal level set values are set to
\[ \phi_R(\tilde{x}) = \text{sign}(\phi_0(\tilde{x}))|\tilde{x} - \tilde{x}| \]  
\[ \psi_R(\tilde{x}) = \psi_0(\tilde{x}) / |\psi_0(\tilde{x})| \]  

In practice, both approaches work well, but the latter typically converges in fewer iterations and has the advantage of not requiring second-order derivative quantities. Since neither of these methods is guaranteed to converge, in the rare case that convergence fails, a secondary method is applied to find the interface: The iteration scheme is simply repeated using only the first step \( \Delta \tilde{x}_1 \) (and not \( \Delta \tilde{x}_2 \)) at every iteration, and the initial guess is the point \( \tilde{x}^0 \) from the first Newton scheme that provided the “closest” approximation by giving the smallest value of \( |f(\tilde{x}^0)| \). Finally, if the second iteration scheme fails, the last resort is choosing the “closest” approximation from both schemes (i.e. the point \( \tilde{x}^0 \) giving the smallest value of \( |f(\tilde{x}^0)| \)).

In cases where very small features develop over a node (such as a tiny droplet encompassing a single node), it is impossible to reinitialize the level set value and its gradient in a way that accurately depicts a signed distance function. This is due to the discontinuity in the gradient near the node and the requirement that the level set function be represented by piecewise cubic polynomials. As a result, a modification to the procedure above is required near very small features, since the usual reinitialization procedure can result in significant distortion of the interface near such nodes. First, note that such small features will occur only at nodes in the band \( N_1 \), and that any distortion to the interface is limited to nodes on this band. One solution is to not reinitialize at such nodes. This is accomplished by first applying the reinitialization procedure as described above and then reverting back to the original level set values \( \phi_0 \) and \( \psi_0 \) on nodes near very small features. In our implementation, these nodes \( \tilde{x}_i \) are identified simply by detecting whether the sign of the level set function differs from the sign of \( \phi(\tilde{x}_i) \) at both nodes on the immediate left and right (or top and bottom) of the node \( \tilde{x}_i \).

Other procedures, such as using central differences to calculate gradient values at these nodes, have been attempted by the authors. But these procedures are imperfect, because they do not allow the level set function to represent a signed distance function near such nodes, and they still result in some perturbation of the interface. This is the problem with representing an inherently non-smooth function (the signed distance function) with a smooth cubic function. A more thorough discussion of this topic is presented in [6], where a “gradient limiting” procedure is introduced to limit the distortion of the interface near such small features. This study shows good preliminary results using this procedure but has the disadvantage of requiring multiple level set values to be stored at some nodes and introducing discontinuities in the level set function. This is clearly an area where additional research is required, and we plan to address it in more detail in future work.

Another problem worth noting is the case in which multiple solutions to (52) exist for a single node. For example, take any node situated between 2 circular interfaces on the line through their centers. By solving (52) on the nodes in \( N_1 \), first, the issue of multiple solutions is almost completely alleviated for all other nodes in the narrow band, since excellent initial guesses are provided by the nodes in \( N_1 \). While reinitializing nodes in the band \( N_1 \), the initial guess is generally good enough to ensure that the correct solution is picked. This issue is complicated by sub-grid features, but we have encountered no problems using the procedures outlined above.

5.3. The narrow-band GALS procedure

**Algorithm 1** is conducted at every time step, advancing the level set function and its gradient forward in time from \( \phi^n \) to \( \phi^{n+1} \) and from \( \psi^n \) to \( \psi^{n+1} \), respectively. This procedure is Step 4 of the full numerical algorithm outlined in Section 2.3. As necessary, **Algorithm 2** is carried out to reinitialize the narrow band, as described in the preceding subsections.

5.3.1. Treatment at boundaries

A number of considerations must be made at the boundaries of both the narrow band and the computational domain. First we describe the criteria for triggering reinitialization.

Reinitialization is triggered whenever the interface is “near” the edge of the narrow band, since we wish to prevent the interface from leaving the narrow band (where it cannot to be represented). This is best accomplished by triggering reinitialization whenever the interface is inside a cell that shares a node in \( N_{w-1} \). The motivation for this reinitialization “trigger” is evident in Fig. 5. Suppose the figure represents the location of the interface and narrow band on a subset of the domain at time \( t \). The velocity field characteristics A and B represent the semi-Lagrangian paths of the characteristics from time \( t \) to time \( t + \Delta t \), terminating at nodes \( x_A \) and \( x_B \), respectively. At time \( t + \Delta t \), the characteristic B will carry a positive level set value to the node \( x_B \), meaning that the interface will move into the cell shared by \( x_A \) and \( x_B \). However, the characteristic A will be unable to carry useful information to the node \( x_A \), since it originates outside the narrow band. To prevent this situation from arising, the narrow band should be rebuilt and the level set function should be reinitialized at time \( t \).

We have described the spatial trigger for reinitialization in the previous paragraph. The reinitialization procedure should also be called after a certain amount of time (say \( w \Delta t \)) has elapsed since the last reinitialization. This prevents the interface from remaining within the narrow band while also undergoing significant distortion and drifting far from a signed distance function.

Following the advection of the level set function on nodes in \( \{N_1, \ldots, N_{w-1}\} \) in **Algorithm 1**, the values of \( \phi, \phi_x, \phi_y, \) and \( \phi_{xy} \) must be extrapolated to the band \( N_w \). Adalsteinsson [2] first developed the level set method for use on a narrow band and provides a variety of techniques for treating the narrow band boundary. In this work, we use linear extrapolation from...
Algorithm 1 Advection on the narrow band.

for all nodes $\vec{x}_i \in \{N_1, \ldots, N_w-1\}$ do
  Solve the characteristic equation (44) using the RK3 scheme of Section 4.4 to find $\vec{\phi}^n$ and $\vec{\psi}^n$. Use Hermite cubic interpolation in the cell containing $\vec{x}_i$ to update the level set and its gradient:
  $\phi^{n+1}(\vec{x}_i) = \phi^n(\vec{x}_i)$
  $\vec{\psi}^{n+1}(\vec{x}_i) = \nabla \phi^n(\vec{x}_i)$
end for

for all nodes $\vec{x}_i \in N_w$ do
  Extrapolate the level set function and gradient values to the boundary of the narrow band using nodes interior to the narrow band. Also extrapolate values to the domain boundary, as appropriate (see Section 5.3.1).
end for

for all nodes $\vec{x}_i \in \{N_1, \ldots, N_w-1\}$ do
  Calculate the cross-derivative $\phi_{xy}(\vec{x}_i)$ via Eq. (37) or another suitable finite difference equation. Copy these values to the boundary level set nodes.
end for

if the interface is near the edge of the narrow band (see Section 5.3.1) then
  Conduct the reinitialization procedure.
end if

Algorithm 2 Reinitialization on the narrow band.

for all nodes $\vec{x}_i$ do
  Redefine the bands $N_i$ based upon the current interface position.
end for

for all nodes $\vec{x}_i \in \{N_1, \ldots, N_w\}$ do
  Find the point on the interface nearest to node $\vec{x}_i$ by solving
  $\phi(\vec{x}) = 0$
  $\vec{\psi}(\vec{x}) \times (\vec{x} - \vec{x}_i) = 0$
  and reinitialize the value of the level set function and its gradient, as described in Section 5.2.
end for

for all nodes $\vec{x}_i \in \{N_1, \ldots, N_{w-1}\}$ do
  Calculate the cross-derivative $\phi_{xy}(\vec{x}_i)$ via Eq. (37) or another suitable finite difference equation.
end for

---

Fig. 5. Interface approaching the narrow band boundary. The black curve represents the zero contour of the level set function. Open circles represent nodes in the $N_w$ band, squares represent nodes in the narrow band but not in $N_w$, and dots represent nodes outside the narrow band. Cells inside the narrow band (where the level set is defined) are shaded in grey, and the band of cells bounded by nodes in $N_{w-1}$ and $N_w$ only is drawn with a crosshatch pattern. The arrows (labeled A and B) represent velocity field characteristics that terminate at the nodes $x_A$ and $x_B$, respectively. Note that $x_A \in N_w$ and $x_B \in N_{w-1}$.

the interior of the narrow band to build an approximate solution on the edge of the narrow band. Specifically, for each node $x_i \in N_w$, all 8 of its neighboring nodes are inspected. Neighboring nodes that are in the set $N_{w-1}$ (along with one of their neighbors) are used to linearly extrapolate data to the edge. For example, for the node $x_C$ in Fig. 5, only one of its neighbors ($x_D$) is in the set $N_{w-1}$. As a result, its level set values are determined by the expressions:

$$\phi(x_C) = 2\phi(x_D) - \phi(x_E)$$

$$\phi_x(x_C) = 2\phi_x(x_D) - \phi_x(x_E)$$
\[
\phi_y(x_C) = 2\phi_y(x_D) - \phi_y(x_E) 
\]

In this case, only one eligible neighbor existed. For nodes with numerous eligible neighbors, the linear extrapolation is conducted along a straight line for each neighbor, and the values are averaged. In practice, this has provided a stable method for extrapolating the level set function. And since the level set values are extrapolated to the narrow band boundary with relatively low accuracy (stability is the chief concern on the boundary), the values of \( \phi_{xy} \) are set to zero at each step. Nave et al. [25] describe setting \( \phi_{xy} = 0 \) as a stable (though low accuracy) approach to assigning the cross-derivative. Since the level set is reinitialized at time intervals of about \( w/\Delta t \), there is little chance for values from the boundary to pollute the level set function at the interface.

The last boundary issue concerns the extrapolation of the level set values to the domain boundary. This proves to be a simple matter in our implementation, since the boundary level set nodes are actually ghost nodes, existing beyond the computational domain, as shown in Fig. 2, where the level set values are stored at the black dots and the grey area is the physical problem domain. The level set values and derivatives (along with their classification in \( N_i \)) are copied from the nearest level set node in the physical domain, except that \( \phi_x \) is set to zero for ghost nodes along a vertical boundary, and \( \phi_y \) is set to zero for ghost nodes along a horizontal wall. The values of \( \phi_{xy} \) on the boundary are copied from the interior after \( \phi_{xy} \) has been calculated on the interior.

5.4. Mass conservation

As demonstrated in Nave et al. [25], the GALS method conserves volume much more robustly than the standard level set method. Consistent with this finding, most of the numerical experiments in Section 6 demonstrate excellent mass conservation, even for turbulent flow (see Section 6.2). However, for simulations on very large time scales, such as the wave tank benchmark of Section 6.4, very gradual decreases in the volume of the denser fluid begin to dominate the simulation. As a result, for long time-scale problems, a simple mass conservation routine is added after advection and prior to reinitialization of the level set function:

\[
\phi \leftarrow \phi + \beta(V_0 - V)
\]

where \( V_0 \) is the initial volume of the denser fluid, \( V \) is the volume of the denser fluid prior to the update, and \( \beta \) is a small constant inversely proportional to the perimeter of the interface. This is performed on all nodes where level set values are stored; note that no change to the gradient quantities is necessary. This intuitive update is similar to the mass conservation routine utilized in Sussman and Uto [39].

The frequency of this update (if applied) may be tailored to the individual experiment at hand. For less volatile simulations (like the solitary wave benchmark of Section 6), the change in volume per iteration is extremely small, so such an update would rarely be needed. In general, one may periodically check the volume and begin adjusting the volume when a specific fluctuation threshold is met.

5.5. Initialization

For many problems, we seek to initialize the position of the zero contour according to a particular curve that is given as a function \( y = f(x) \) (see, for example, the position of the solitary wave free surface (69) in Section 6.3). While the appropriate initial values for \( \phi \) and \( \tilde{\psi} \) on the computational domain may not be obvious, we introduce an easy procedure for initialization using the narrow-band GALS method:

1. For all nodes on the computational domain, assign the following level set values:

\[
\begin{align*}
\phi(x) &= f(x_i) - y_j \\
\phi_x(x) &= f'(x_i) \\
\phi_y(x) &= -1 \\
\phi_{xy}(x) &= 0
\end{align*}
\]

where the node position is given by \( x = (x_i, y_j) \) on the MAC grid.

2. Build the narrow band, as in Section 5.2.1.

3. Reinitialize all nodes on the narrow band by solving Eq. (52), as in Section 5.2.1.

Since subroutines are already in place for these purposes, the initialization of the level set function and its derivatives proves to be very simple. The result is a signed distance function with a zero level set contour satisfying \( y = f(x) \).
6. Numerical results

6.1. Sloshing benchmark

The first test examines the low-amplitude oscillations of the free surface between two viscous fluids in the rectangular domain shown in Fig. 6(a). The initial position of the free surface is given by the expression:

$$\eta(x) = d + a_0 \sin(\pi (0.5 - x))$$

(68)

where the initial amplitude is $a_0 = 0.01$ m, the characteristic dimension of the tank in Fig. 6(a) is $d = 1$ m, and the origin is positioned at the lower left corner of the domain. As the simulation progresses, the free surface maintains the shape of Eq. (68) but with varying amplitude. The physical parameters for the simulation are $Re = 100$ and $\nu_2/\nu_1 = \rho_2/\rho_1 = 100$, and the simulation is conducted on a uniform mesh of dimension $100 \times 130$, where the vertical dimension of the tank is $1.3d$. Free-slip boundary conditions are applied on all walls of the computational domain.

The results of the time evolution of the free surface are compared to the analytical solution derived by Prosperetti [32] for the linearized Navier–Stokes equations. The numerical and analytical results for the evolution of the free surface amplitude are compared in Fig. 6(b) in non-dimensional time ($t \sqrt{g/d}$). The comparison shows excellent agreement between the numerical GALS result and the analytical solution. Due to the very small motions of the interface, the simulation experienced a mass variation of less than 0.04% (without the aid of the mass conservation routine of Section 5.4).

6.2. Dambreak benchmark

The dambreak benchmark simulates the behavior of a rectangular water column after the failure of a dam supporting it. The initial configuration of the fluid is shown in Fig. 7(a), where $L_s$ represents the position of the surge front along the bottom of the domain as the simulation progresses.

Inspired by the experiments of Martin and Moyce [23], we choose the parameter $a = 0.05715$ m and a domain of dimension $5a \times 1.25a$ with the physical parameters of the two fluids corresponding to realistic values for water and air at room temperature with a density ratio of 816 and a viscosity ratio of 63.4. Our numerical experiment shows a solution for the surge front velocity (Fig. 7(b)) that is faster than the results found experimentally by Martin and Moyce [23]. This disparity is common for most interface-tracking methods [1,15,19,31], prompting Hu and Sueyoshi [19] to repeat the experiment 58 years later. The results of these two laboratory experiments are compared to the solution obtained using the GALS method in Fig. 7(b) in non-dimensional time ($t \sqrt{g/d}$). The GALS solution is found to lie between these two sets of data.

Fig. 8 shows the time evolution of the dambreak solution in five frames at a resolution of $320 \times 80$. Despite the turbulence of the flow, the volume of the denser fluid varied by less than 0.5% of the initial volume for resolutions as coarse as $80 \times 20$ (without the aid of the mass conservation routine of Section 5.4).
As noted in Nichols [26] and Sussman [41], applying a standard no-slip boundary condition on the bottom wall is incompatible with a moving contact line on the same wall, since the prescribed no-slip velocity along the wall is zero. This results in a flow that is slightly retarded by the inability of the contact line to move at the appropriate speed. As a result, numerous strategies have been implemented to advance a moving contact line on a wall boundary, but they frequently rely on heuristic methods that utilize free parameters, since the motion of such a contact line is not well understood. A more popular and elegant choice is to utilize free-slip boundary conditions on all walls of the domain, thereby permitting nonzero tangential fluid velocity at the physical walls [12,14,21,27]. This is an appropriate choice for the dam break benchmark since the low viscosity and high velocity of the flow result in a boundary layer that is far smaller than the width of a single grid cell.

Overall, the results for the finest resolutions fall within the range of experimental results and appear to exhibit convergent behavior.

6.3. Solitary wave benchmark

We examine the run-up of a solitary wave on a vertical wall, as depicted in Fig. 9(a). At $t = 0$, the level set function and the fluid velocity are prescribed, giving a solitary waveform of amplitude $a_0$ moving from left to right in the center of the computational domain. The run-up amplitude $a_{\text{run-up}}$ is the highest amplitude reached by the wave as it ascends the right wall.

The initial free surface location and velocity components are given by the third-order solitary wave solution of Grimshaw [16], repeated here:

$$
\eta(x) = h + \varepsilon s^2 - \frac{3}{4} \varepsilon^2 s^2 t^2 + \varepsilon^3 \left( \frac{5}{8} s^2 t^2 - \frac{101}{80} s^4 t^2 \right)
$$

$$
\frac{u(x, y)}{\sqrt{gh}} = \varepsilon s^2 - \varepsilon^2 \left[ \frac{1}{4} s^2 + s^4 + y^2 \left( \frac{3}{2} s^2 - \frac{9}{4} s^4 \right) \right] - \varepsilon^3 \left[ \frac{19}{40} s^2 + \frac{1}{5} s^4 - \frac{5}{5} s^6 \right] + y^2 \left( \frac{3}{2} s^2 - \frac{15}{4} s^4 + \frac{15}{2} s^6 \right) + y^4 \left( \frac{3}{8} s^2 + \frac{45}{16} s^4 - \frac{45}{16} s^6 \right)
$$
Fig. 8. Dambreak simulation. The mesh is uniform and of dimension 320 × 80. From top to bottom, the samples shown are approximately at times $t = 0.00, 0.06, 0.12, 0.18$, and $0.24$ s.

$$\frac{v(x,y)}{\sqrt{gh}} = -\sqrt{3} \epsilon y^* t \left\{ -\epsilon s^2 + \epsilon^2 \left[ \frac{3}{8} s^2 + 2 s^4 + y^*^2 \left( \frac{1}{2} s^2 - \frac{3}{2} s^4 \right) \right] + \epsilon^3 \left[ \frac{49}{640} s^2 - \frac{17}{20} s^4 - \frac{18}{5} s^6 \right] 
+ y^*^2 \left( -\frac{13}{16} s^2 - \frac{25}{16} s^4 + \frac{15}{2} s^6 \right) + y^*^4 \left( -\frac{3}{40} s^2 + \frac{9}{8} s^4 - \frac{27}{16} s^6 \right) \right\}$$

(71)

where $\epsilon$ is the ratio $a_0/h$, $y^* = (y - h)/h$, $s = \text{sech}(\alpha x)$, $t = \tanh(\alpha x)$, and $\alpha = \sqrt{3\epsilon/4(1 - 5\epsilon/8 + 71\epsilon^2/128)}$. This approximation provides a stable and symmetric solitary waveform for values of $a_0/h$ up to about 0.4. This numerical experiment is conducted with $h = 1$ and for various initial amplitudes up to 0.4. The density of the denser fluid is set to 1000 kg/m$^3$ while the density of the lighter fluid is set to approximately zero (0.0001 kg/m$^3$). Viscosity is zero in both fluid phases and the grid resolution is 200 × 60 for all trials with free slip boundary conditions on the solid wall boundaries. The results show
Fig. 9. Soliton evolution. (a) The initial waveform and peak run-up waveform. (b) The wave profile is shown at equally-spaced time intervals starting at the initial position and ending at the highest point \(a_0/h = 0.325\). The overlaid frame shows the superimposition of the first five profiles.

A long-term goal for this research is the incorporation of fluid–solid interaction in the context of modeling wave energy converters. Since experimental validation will be an invaluable tool in developing these numerical methods, we present a comparison of the narrow-band GALS method to experimental data obtained in a wave tank at Duke University.

The wave tank, which is narrow enough to be approximated by a 2D simulation, measures 9.1 meters in length and uses a driven, oscillating wedge to generate low amplitude waves. A diagram of the wave tank is given in Fig. 11(a). The wedge moves only in the vertical direction, and its tip has an initial position of \((x_0, y_0) = (0.114, 0.095)\), where the origin is located at the bottom left corner of the wave tank and the units are meters. Its motion is sinusoidal having an amplitude of 8.1 cm and an arbitrary frequency with the angle of the wedge given by 26.7 degrees. In our numerical experiment, the method of Sanders et al. [33] is used to advect the fluid with the arbitrary motion of the solid wedge. The foam beach (illustrated in Fig. 11(a)) damps the wave motion at the far end of the tank and is modeled using a body force on the fluid that is proportional to the fluid velocity. In these trials, a proportionality constant of 30.0 is used. The physical parameters of the two fluids correspond to realistic values for water and air at room temperature with a density ratio of 816 and a viscosity ratio of 63.4, and the grid resolution is \(300 \times 25\) for all trials.

The experiment is conducted by allowing the wedge to oscillate continuously until a relatively steady state is achieved, at which point the mean wave amplitude is measured over a large time span. Due to the long time scale of this experiment,
the mass conservation procedure of Section 5.4 was necessary to prevent very small incremental losses in the water volume from accumulating. As a result, the water volume is held approximately constant throughout the course of the experiment.

The results of the numerical and experimental trials are shown in Fig. 11(b) using a water height of 45 cm. In general, the trends of the numerical and experimental amplitudes are in good agreement. We suspect that the lowest-frequency
peaks in amplitude are a result of the characteristic frequency of the wave tank and the inability of the beach to completely dampen waves of all frequencies.

6.5. Rising bubble

In order to test the relative order of convergence of the method, a rising bubble benchmark is conducted utilizing the domain in Fig. 12(a) in the manner of [4] and [20]. A bubble of lower density and viscosity than the surrounding fluid is allowed to rise due to the force of gravity. Parameters for this test are taken from [4] and [20]: \( \rho_1/\rho_2 = \mu_1/\mu_2 = 10, \ g = 0.98 \text{ m/s}^2, \ \sigma = 24.5 \text{ N/m}, \) and \( Re = 35. \) Fig. 12(b) shows the bubble free surface at equally-spaced time intervals during the course of the simulation.

One advantage of the gradient-augmented level set method is the ability to use the Hermite cubic polynomial to determine the derivatives needed for the curvature (26) at a given point without resorting to large finite difference stencils. That is the strategy explored here and in the next test. However, it was found that despite the second-order accuracy of the curvature approximation provided by the GALSM (see Section 6.6), this surface tension dominated flow was slightly unstable near “pinched” points (points of high curvature on the interface) when differentiating the local Hermite cubic function to find the second-order derivatives \( \phi_{xx} \) and \( \phi_{yy} \) in (26). As a result, second-order central difference approximations of gradient values were used to determine these 2 quantities. For example, the value of \( \phi_{xx}^{i,j} \) at a node is found via

\[
\phi_{xx}^{i,j} = \frac{(\phi_{xx}^{i+1,j} - \phi_{xx}^{i-1,j})}{(2h_x)}
\]

and the value of \( \phi_{yy}^{i,j} \) at a node is found via

\[
\phi_{yy}^{i,j} = \frac{(\phi_{yy}^{i,j+1} - \phi_{yy}^{i,j-1})}{(2h_y)}
\]

At cell walls, similar central difference approximations are made to determine the second-order derivative terms for calculating curvature. These values produced a much more stable simulation than the second-order derivatives calculated from differentiating the piecewise Hermite cubic polynomials, which produced non-physical stretching and jittering of the interface near pinched points. The cause of this phenomenon is likely the distortion caused by reinitialization near very small features (see Section 5.2.2). As previously noted, this is a topic of continuing research that will be discussed further in future work. At present, the finite difference stencil used for \( \phi_{xx} \) and \( \phi_{yy} \) is comparable in size to the finite difference stencil used at each step for calculating \( \phi_{xy} \); therefore, the calculation of these derivatives requires no additional inconvenience.

The relative order of convergence is found by recording the mean rise velocity inside the bubble for simulations of varying coarseness. The error norms are calculated over the first 5 s of the simulation utilizing the norms defined in [4]. The highest-resolution simulation (\( h = 1/200 \)) is used as the true solution, and the experiment simultaneously tests convergence in time and space by refining both for each trial.

It has been noted that the gradient-augmented level set method exhibits third order global convergence in the presence of a prescribed velocity field. However, the spatial discretizations used in the fluid solver exhibit second-order spatial accuracy away from the interface and first-order accuracy near the interface (due to the smeared Heaviside profile). Moreover, the pressure projection method is first-order accurate in time, and the GALSM utilizes a locally fourth-order accurate time integrator to trace the semi-Lagrangian characteristics. As a result, the method will exhibit at least first-order accuracy in space and time. The results of this experiment depict slightly better than first-order convergence, as indicated in Table 1. We speculate that the observed increase in accuracy results from a smaller contribution of the first-order spatial terms as the smeared Heaviside profile shrinks and the second-order spatial terms dominate.

To test mass conservation, this experiment is also conducted using both the gradient-augmented level set method and the standard level set method. Using no mass conservation techniques (such as the one described in Section 5.4) and a 40 \( \times \) 80 cell grid, the fluid volume varies by only 1.5% during the first 5 s of the simulation for the GALS implementation.
Table 1
Relative order of convergence.

<table>
<thead>
<tr>
<th>h</th>
<th>|e|_1</th>
<th>ROC1</th>
<th>|e|_2</th>
<th>ROC2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1/10</td>
<td>5.515E-2</td>
<td>–</td>
<td>6.458E-2</td>
<td>–</td>
</tr>
<tr>
<td>1/20</td>
<td>2.350E-2</td>
<td>1.231</td>
<td>2.869E-2</td>
<td>1.170</td>
</tr>
<tr>
<td>1/40</td>
<td>8.385E-3</td>
<td>1.487</td>
<td>1.012E-2</td>
<td>1.504</td>
</tr>
<tr>
<td>1/60</td>
<td>4.563E-3</td>
<td>1.501</td>
<td>5.570E-3</td>
<td>1.472</td>
</tr>
<tr>
<td>1/80</td>
<td>2.951E-3</td>
<td>1.514</td>
<td>1.068E-2</td>
<td>1.064</td>
</tr>
<tr>
<td>1/100</td>
<td>4.563E-3</td>
<td>1.501</td>
<td>5.570E-3</td>
<td>1.472</td>
</tr>
<tr>
<td>1/120</td>
<td>8.385E-3</td>
<td>1.487</td>
<td>1.068E-2</td>
<td>1.064</td>
</tr>
<tr>
<td>1/140</td>
<td>2.350E-2</td>
<td>1.231</td>
<td>2.869E-2</td>
<td>1.170</td>
</tr>
<tr>
<td>1/160</td>
<td>5.515E-2</td>
<td>–</td>
<td>6.458E-2</td>
<td>–</td>
</tr>
</tbody>
</table>

Fig. 13. Reinitialization benchmark.

Table 2
Reinitialization convergence of \( \phi \).

<table>
<thead>
<tr>
<th>h</th>
<th>|e|_1</th>
<th>OC1</th>
<th>|e|_2</th>
<th>OC2</th>
<th>|e|_\infty</th>
<th>OC_\infty</th>
</tr>
</thead>
<tbody>
<tr>
<td>1/64</td>
<td>5.687E-9</td>
<td>–</td>
<td>6.413E-9</td>
<td>–</td>
<td>1.134E-8</td>
<td>–</td>
</tr>
<tr>
<td>1/128</td>
<td>4.047E-10</td>
<td>3.813</td>
<td>4.500E-10</td>
<td>3.833</td>
<td>7.332E-10</td>
<td>3.951</td>
</tr>
<tr>
<td>1/512</td>
<td>1.503E-12</td>
<td>4.010</td>
<td>1.651E-12</td>
<td>4.033</td>
<td>2.895E-12</td>
<td>3.972</td>
</tr>
</tbody>
</table>

Using the standard level set method, the volume varies by 62.3%. Using a \( 80 \times 160 \) cell grid results in volume variation of 0.3% and 23.5%, respectively.

6.6. Reinitialization

In order to test the order of convergence of the reinitialization procedure, we initialize the level set function and its gradient on the unit square as

\[
\phi(x, y) = e^{-(x-0.5)^2-(y-0.5)^2} \quad e^{0.15^2}
\]

\[
\phi_x(x, y) = -2(x-0.5)e^{-(x-0.5)^2-(y-0.5)^2}
\]

\[
\phi_y(x, y) = -2(x-0.5)e^{-(x-0.5)^2-(y-0.5)^2}
\]

which generates a zero level set representing a circle of radius 0.15 in the center of the domain. Note that the level set function itself is far from a signed distance function and the cross derivative is not prescribed. The reinitialization procedure is executed one time to generate a signed distance function. Figs. 13(a) and 13(b) show the level set function before and after reinitialization, and Table 2 shows fourth order convergence in all three error norms, which are calculated on a band of width 0.2 centered over the zero contour. Table 3 shows that the components of the gradient exhibit third order convergence (the results are identical for each component), and Table 4 shows second order convergence of the curvature.

The Hermite cubic polynomial on each cell represents a fourth order approximation to the original function \( \phi(x, y) \), since the values of \( \phi \) and \( \nabla \phi \) are prescribed at the nodes and \( \phi_{xy} \) is calculated to second order. Because the reinitialization procedure finds the exact location of the interface that satisfies (52) (within a tolerance), the reinitialization procedure is expected to produce a fourth order approximation of the exact solution, as has been demonstrated. Additionally, we expect to achieve third order convergence in first derivatives and second order convergence in second derivatives of \( \phi(x, y) \). Since second order derivatives exist in all terms of the curvature expression (26), second order convergence of curvature is expected and is achieved. Note that the second-order derivatives required in Eq. (26) are calculated with both the Hermite cubic polynomial (Table 4) and with the second-order central differences given in the previous section (Table 5).
6.7. Breaking wave

The domain of the solitary wave benchmark is modified by including a rectangular reef in the path of the wave, as shown in Fig. 14, for the purpose of illustrating a breaking wave. The ratio defining the reef height is \( h_r/h = 0.85 \) and the third order solitary wave solution of Grimshaw is again utilized to initialize a moving solitary wave of amplitude \( a_0/h = 0.4 \). The breaking wave front is shown in Fig. 15. Despite the coarseness of the mesh (600 × 60), the detail of the front is captured well by the cubic shape functions. The behavior of the breaking wavefront is qualitatively comparable to those of Helluy et al. [18].

6.8. Computation time

This section investigates the amount of computation time required to complete the fluid solve and the level set procedures. Specifically, the first 0.01 s of the bubble experiment of Section 6.5 is conducted, and the amount of time spent calculating (a) the fluid velocity and pressure (using an SSOR PCG solve), (b) the advection steps of the GALS procedure (including extrapolation to the edges of the narrow band), and (c) the reinitialization procedure introduced here is recorded at each time step. Reinitialization is conducted at every time step, and the average value of each task is shown in the tables below. Table 6 shows the results of using only a narrow band for the level set, and Table 7 shows the results of defining the level set on the entire domain. The final column in each table shows the percentage of calculation time spent on the GALS procedures (advection and reinitialization). All tests were conducted on a 2.4 GHz Intel Core i5 chip as serial processes.

Defining \( n = 1/h \) where \( h = h_x = h_y \) in this experiment, we anticipate that \( O(n) \) operations are needed to advect and reinitialize the level set on a narrow band, since the band is a constant number of nodes in width and its length increases
Fig. 15. Wave breaking over a submerged reef. The mesh is uniform and of dimension $600 \times 60$. From top to bottom, the samples shown are at times $t = 2.40, 3.00, 3.84, 4.44, \text{ and } 6.00 \text{ s}$.

Table 6
Average computation time per time step: Advection and reinitialization on a narrow band only.

<table>
<thead>
<tr>
<th>$h$</th>
<th>Fluid solve</th>
<th>Order</th>
<th>Advection</th>
<th>Order</th>
<th>Reinitialization</th>
<th>Order</th>
<th>GALSM percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>1/20</td>
<td>5.628E−3</td>
<td>–</td>
<td>1.914E−4</td>
<td>–</td>
<td>1.254E−3</td>
<td>–</td>
<td>20.43%</td>
</tr>
<tr>
<td>1/40</td>
<td>4.805E−2</td>
<td>3.094</td>
<td>6.100E−4</td>
<td>1.672</td>
<td>2.344E−3</td>
<td>0.903</td>
<td>5.79%</td>
</tr>
<tr>
<td>1/80</td>
<td>3.485E−1</td>
<td>2.895</td>
<td>1.811E−3</td>
<td>1.570</td>
<td>5.311E−3</td>
<td>1.180</td>
<td>2.00%</td>
</tr>
<tr>
<td>1/120</td>
<td>1.103E0</td>
<td>2.842</td>
<td>2.449E−3</td>
<td>0.745</td>
<td>7.975E−2</td>
<td>1.003</td>
<td>0.94%</td>
</tr>
<tr>
<td>1/160</td>
<td>2.200E0</td>
<td>2.400</td>
<td>3.461E−3</td>
<td>1.202</td>
<td>9.342E−2</td>
<td>0.550</td>
<td>0.58%</td>
</tr>
</tbody>
</table>

Table 7
Average computation time per time step: Advection and reinitialization on the entire domain.

<table>
<thead>
<tr>
<th>$h$</th>
<th>Fluid solve</th>
<th>Order</th>
<th>Advection</th>
<th>Order</th>
<th>Reinitialization</th>
<th>Order</th>
<th>GALSM percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>1/20</td>
<td>5.861E−3</td>
<td>–</td>
<td>5.225E−4</td>
<td>–</td>
<td>1.416E−2</td>
<td>–</td>
<td>28.52%</td>
</tr>
<tr>
<td>1/40</td>
<td>4.685E−2</td>
<td>2.999</td>
<td>3.208E−3</td>
<td>2.618</td>
<td>1.085E−1</td>
<td>2.937</td>
<td>70.45%</td>
</tr>
<tr>
<td>1/80</td>
<td>3.258E−1</td>
<td>2.798</td>
<td>1.480E−2</td>
<td>2.206</td>
<td>6.315E−1</td>
<td>2.541</td>
<td>66.48%</td>
</tr>
<tr>
<td>1/120</td>
<td>1.015E0</td>
<td>2.803</td>
<td>3.474E−2</td>
<td>2.104</td>
<td>1.895E0</td>
<td>2.710</td>
<td>65.53%</td>
</tr>
<tr>
<td>1/160</td>
<td>2.267E0</td>
<td>2.793</td>
<td>6.182E−2</td>
<td>2.003</td>
<td>4.422E0</td>
<td>2.946</td>
<td>66.42%</td>
</tr>
</tbody>
</table>

with $n$. As a result, the number of nodes in the narrow band increases with $O(n)$, and advection and reinitialization require $O(1)$ operations on each node. Table 6 shows the expected result to be true, and we discover that the time spent advecting and reinitializing the level set becomes extremely small relative to the computation time of the fluid solver as $n$ increases.

When the level set is defined on the entire domain (rather than the narrow band alone), we expect the computation to be $O(n^2)$, since the number of nodes corresponds to $O(n^2)$ and the level set operations are still $O(1)$ on each node. However, Table 7 shows that the increase in computation time is consistently larger than $O(n^2)$ and increases at nearly the same rate as the fluid solver. As a result, the level set computations consistently exceed the time required for the fluid solver. We attribute this increase in computation time to the slower convergence of the reinitialization procedure on nodes very far away from the interface.
This test demonstrates that the narrow band gradient-augmented level set procedure described in this paper is very computationally efficient relative to a two-phase Navier–Stokes solver, making the GALSM a very small portion of the overall computation. Moreover, the reinitialization method introduced here is prohibitively expensive when applied on the entire domain, unless a more sophisticated Newton iteration procedure is conducted for solving the reinitialization equation at each node. However, there is no need to reinitialize the level set function on the entire domain when level set values are only of interest on a tiny band of nodes near the interface. We note that more efficient procedures for reinitializing the level set function on the entire domain exist. Fast marching and fast sweeping methods utilize advanced data structures and characteristic information to minimize the number of floating point operations required for reinitialization on the entire domain, but they are typically second order accurate (versus the fourth order accuracy evidenced in Section 6.6). The reinitialization procedure developed here is intended for implementation on a narrow band only, and as a result, its application to the entire domain is excessively inefficient. See [5] for a description of a hybrid reinitialization procedure that is similar to the one presented here near the interface but also includes a fast marching procedure for nodes far away from the interface. We reemphasize that there is no need for level set values far away from the interface in the current flow application, since the CFL conditions limit motion of the interface to one cell per time step.

7. Conclusion

The presented method utilizes a multi-phase Navier–Stokes flow solver and third-order narrow-band gradient-augmented level set method for the representation of the free surface between fluid phases. As demonstrated in Section 6.5, the method produces solutions for two-phase incompressible flow that are at least first-order accurate. The other numerical benchmark tests reveal the method produces solutions that agree well both with analytical results and experimental data. In particular, it has been demonstrated that the method is well-suited to problems involving ocean waves (including breaking waves), a key area of research interest. Moreover, the use of in-house wave tank facilities provides a straightforward method of validating the method and will be utilized again in subsequent studies. The superiority of the GALSM over the standard level set method is demonstrated by a more than tenfold improvement in volume conservation over the standard method.

Further research will involve the implementation of the method in three spatial dimensions. The optimally local advection scheme of the GALM method is very well suited for parallelization since advection occurs on a single cell and doesn’t require a large difference stencil like traditional level set approaches. (It should be noted that the computation time of the level set routines is also very small compared to the fluid solver.) Of great interest is the incorporation of the GALS method into a third-order (or higher) difference scheme or finite volume Navier–Stokes solver, making the GALSM a very small portion of the overall computation.

Acknowledgements

Curtis Lee’s effort on this work was supported by a Computational Science Graduate Fellowship (CSGF) from the US Department of Energy, and Peter Mucha’s effort was supported by NSF DMS-0645369. This support is gratefully acknowledged.

References