High Resolution Sharp Computational Methods for Elliptic and Parabolic Problems in Complex Geometries

Frédéric Gibou * Chohong Min † Ron Fedkiw ‡

November 2, 2012

In honor of Stan Osher's 70th birthday

Abstract

We present a review of some of the state-of-the-art numerical methods for solving the Stefan problem and the Poisson and the diffusion equations on irregular domains using (i) the level-set method for representing the (possibly moving) irregular domain's boundary, (ii) the ghost-fluid method for imposing the Dirichlet boundary condition at the irregular domain's boundary and (iii) a quadtree/octree node-based adaptive mesh refinement for capturing small length scales while significantly reducing the memory and CPU footprint. In addition, we highlight common misconceptions and describe how to properly implement these methods. Numerical experiments illustrate quantitative and qualitative results.

1 Introduction

We are considering three of the main equations in the class of elliptic and parabolic partial differential equations: the Poisson equation, the diffusion equation and the Stefan problem. The Poisson and the diffusion equations are two characteristic equations used in a plethora of scientific and engineering applications. They are important in their own right, for example in predicting the heat distribution in engines or the distribution of chemical species (see [48] and the references therein); they are also core building blocks in fields as diverse as fluid dynamics [73, 157, 56], finance (see [17]) and image processing (see e.g. [106, 102] and the references therein). The Stefan problem is a model often used to describe solidification processes, the method of choice for growing single crystals with applications in the aerospace industry (see e.g. [33] and the references therein). It is also used as a component for the study of vaporization processes [156, 35, 44, 145, 153, 146, 131, 132, 133, 134]. In addition, this model is applicable to a wide variety of other applications, including epitaxial growth [23, 116].

In the large majority of applications, the domains of integration for these equations have irregular shapes so that no closed-form solutions exist. Numerical methods are thus necessary and face three main challenges. First, the description of the physical domain must be versatile enough to account for the motion of free boundaries, which is the case of the Stefan problem. Second, boundary conditions must be imposed *at* the boundary of the irregular domain. We are focusing in this review on Dirichlet boundary conditions, i.e. the solution itself is given at the boundary. The case of imposing Neumann or Robin or jump boundary conditions is not the focus of this paper and we refer the interested reader to [32, 63, 11, 146, 75, 77, 104, 78, 114, 55, 50, 150, 83] and the references therein. Finally, typical scientific applications exhibit solutions with different length scales. In the context of electrostatics for example, the electric double layer is an extremely small region where the potential varies rapidly and must be captured by the numerical solution. From the numerical point of view, small length scales are related to very fine grids for which uniform grids are too inefficient to be practical.

*Mechanical Engineering Department & Computer Science Department, University of California, Santa Barbara, CA 93106.
†Mathematics Department, Ewha Womans University, Seoul, Korea 120-750.
‡Computer Science Department, Stanford University, Stanford CA 94305-9020.

In this paper, we review a successful approach for solving the Poisson and the diffusion equations and the Stefan problem using (i) a level-set approach to capture the geometry of the physical domain or the free boundary, (ii) a ghost-fluid method to impose Dirichlet boundary conditions at the irregular domain's boundary and (iii) a node-based adaptive mesh refinement framework based on quadtree/octree Cartesian grids to capture the small length scales of the problem while significantly reducing the CPU and memory requirements. The methods presented are numerically robust, second-order accurate in the L^∞-norm (and in some case third or fourth-order accurate) and applicable to arbitrary geometries in two and three spatial dimensions.

2 Equations and Free Boundary Representation

2.1 The Diffusion and the Poisson Equations

Consider a Cartesian computational domain, $\Omega \in \mathbb{R}^n$, with exterior boundary $\partial\Omega$ and a lower dimensional interface, Γ, that divides the computational domain into disjoint pieces, Ω^- and Ω^+ (see figure 1). The diffusion equation on Ω is given by:

$$\partial u/\partial t = \nabla \cdot (\beta \nabla u) + S, \tag{1}$$

where $u = u(\mathbf{x}, t)$ is the unknown, $\mathbf{x} = (x, y, z)$ is the space variable, $S(\mathbf{x})$ is the source term and $\beta(\mathbf{x})$ is the diffusion coefficient, i.e. a positive variable bounded from below by a strictly positive constant. Typically, the values for β are different constants in Ω^- and Ω^+. On $\partial\Omega$, either Dirichlet or Neumann boundary conditions are specified. A Dirichlet boundary condition of $u(\mathbf{x}) = u_\Gamma(\mathbf{x})$ is imposed on Γ. The initial condition for u is also given to close the system. The Poisson equation is the steady-state of the diffusion equation and therefore given by:

$$\nabla \cdot (\beta \nabla u) + S = 0. \tag{2}$$

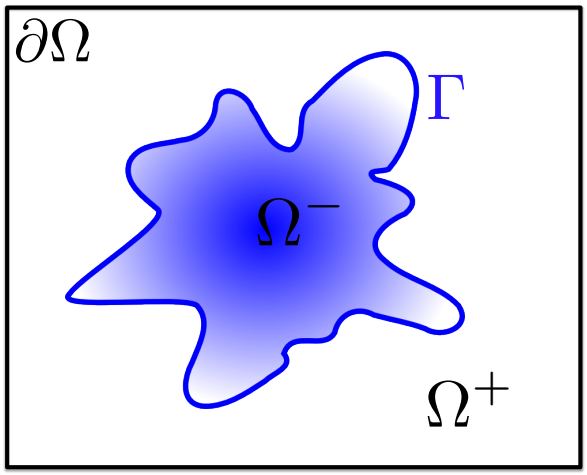


Figure 1: Schematic and notations of a typical computational domain.

2.2 The Stefan Problem

In the context of solidification phenomena, the Stefan problem describes the evolution of a scalar field, T (the temperature), equal to T_s in Ω^- and T_l in Ω^+, such that:

$$\begin{cases} \partial T_s/\partial t &= \nabla \cdot (D_s \nabla T_s) \qquad \text{in } \Omega^-, \\ \partial T_l/\partial t &= \nabla \cdot (D_l \nabla T_l) \qquad \text{in } \Omega^+, \end{cases} \tag{3}$$

where the subscripts s and l denote the solid and liquid phases, respectively. In general, the diffusion constants D_s and D_l are discontinuous across the solidification front Γ. The temperature at the solid-liquid interface is continuous, which is written as:

$$T_s = T_l = T_\Gamma \quad \text{on } \Gamma,$$

where T_Γ denotes the local interface temperature. The relation between the relevant physical quantities at the interface is given by Gibbs-Thompson boundary condition (see e.g. [6, 5]):

$$T_\Gamma = -\epsilon_c \kappa - \epsilon_v \mathbf{V} \cdot \mathbf{n}, \tag{4}$$

where $\mathbf{V}$ denotes the interface velocity field, $\mathbf{n}$ denotes the normal vector to the interface and κ denotes the interface's mean curvature. The parameters ϵ_c and ϵ_v control the strength of surface tension forces and molecular kinetics, respectively. Finally, the normal velocity at the interface is given by the jump in the temperature fluxes across the interface:

$$\mathbf{V} \cdot \mathbf{n} = -(D_l \nabla T_l - D_s \nabla T_s) \quad \text{on } \Gamma. \tag{5}$$

2.3 Domain Representation - The Level-Set Method

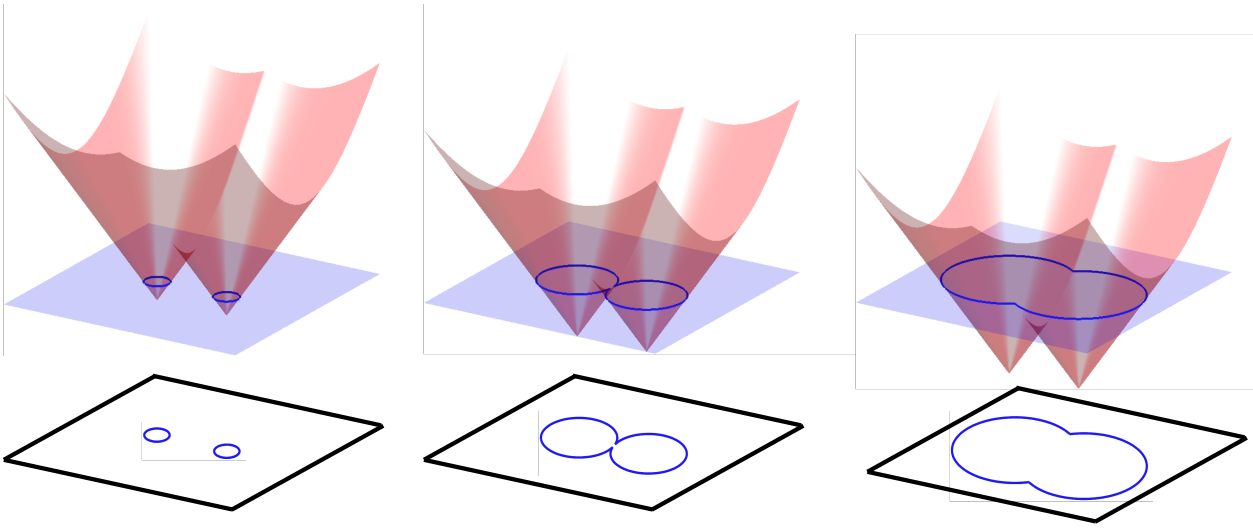


Figure 2: Level-set representation of a free boundary (blue solid line) in two spatial dimensions, moving in its normal direction, and subsequent changes in topology that are handled automatically. The level-set function is depicted in red. (Color online).

The irregular geometries and, in the case of the Stefan problem, the motion of the free boundary are described by the level-set method of Osher and Sethian [103]. This approach represents a curve in two spatial dimensions or a surface in three spatial dimensions by the zero-contour of a higher dimensional function, ϕ, called the level-set function, which is defined as the signed distance function to Γ:

$$\phi(\mathbf{x}) = \begin{cases} -\ d & \text{for } \mathbf{x} \in \Omega^-, \\ +\ d & \text{for } \mathbf{x} \in \Omega^+, \\ \quad 0 & \text{for } \mathbf{x} \in \Gamma, \end{cases}$$

where d is the Euclidian distance to Γ. Under a velocity field $\mathbf{V}$, the interface deforms according to the level-set equation:

$$\frac{\partial \phi}{\partial t} + \mathbf{V} \cdot \nabla \phi = 0. \tag{6}$$

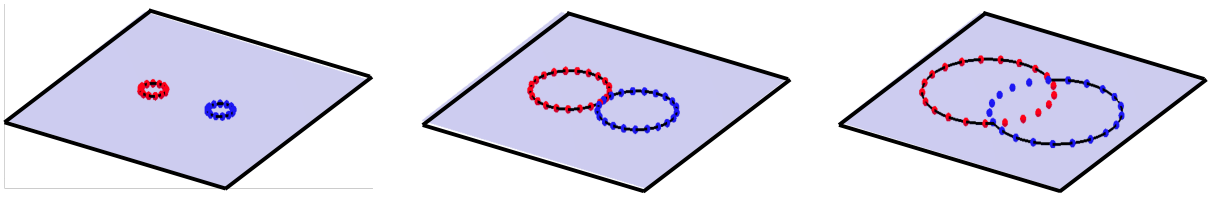


Figure 3: Front-tracking representation (dots) of a free boundary (black solid line) in two spatial dimensions, moving in its normal direction, and subsequent changes in topology that require surgical procedures and extra logics. The advantage of front-tracking methods is their accuracy. (Color online).

The main advantage of an implicit representation of a moving front is its ability to naturally handle changes in topology, as illustrated in figure 2. This is in contrast to explicit methods, e.g. the front-tracking method of Tryggvason *et al.* [64, 65, 115, 146, 153] for which changes in topology require extra work (see figure 3). We note, however, that front-tracking methods have the advantage of accuracy (front-tracking preserve volumes better than level-set methods for the same grid resolution) and we refer the interested reader to the work of [18] for a front-tracking method that handle changes in topology. Volume of fluid methods also adopt an implicit formulation using the volume fraction of one phase in each computational cells (see e.g. [9, 13, 12, 34, 36, 54, 100, 117, 139, 154, 161, 163] and the references therein). These methods have the advantage of conserving the total volume by construction. They are however more complicated than level-set methods in three spatial dimensions and it is difficult to compute accurate smooth geometric properties such as curvatures from the volume fraction alone, although we refer the reader to the interesting work of Popinet on this issue [111]. Also, we note that phase-field models have been extensively used in the case of solidification processes [20, 37, 57, 68, 69, 67, 94, 57, 112, 113]. However, these models do not represent the interface in a sharp fashion, which in turn leads to a degradation of the accuracy where it matters most and impose sometimes stringent time step restrictions.

The level-set function can also be used to compute the normal to the interface $\mathbf{n}$ and the interface's mean curvature κ:

$$\mathbf{n} = \nabla\phi/|\nabla\phi| \qquad \text{and} \qquad \kappa = \nabla \cdot \mathbf{n}.$$

To keep the values of ϕ close to those of a signed distance function, i.e. $|\nabla\phi| = 1$, the reinitialization equation introduced in Sussman *et al.* [140]:

$$\frac{\partial\phi}{\partial\tau} + S(\phi_o)\left(|\nabla\phi| - 1\right) = 0, \tag{7}$$

is traditionally iterated for a few steps in fictitious time τ. Here $S(\phi_o)$ is a smoothed-out signum function and ϕ_0 is the value of the level-set function at the beginning of the reinitialization procedure.

3 The Ghost-Fluid Method for the Diffusion and the Poisson Equations

The Ghost-Fluid method (GFM), introduced in Fedkiw *et al.* [42] in the case of compressible gas dynamics, is a numerical technique designed to apply sharp boundary conditions at irregular domains and free boundaries. The basic idea is to consider two copies of the solution and, by defining ghost values that implicitly capture jump conditions, avoid numerically differentiating across discontinuities . This methodology has been applied to a wide range of applications including deflagration in Fedkiw *et al.* [43], compressible/incompressible fluids in Caiden *et al.* [24], flame propagation in Nguyen *et al.* [97], the Poisson equation with jump conditions in Liu *et al.* [78], free surface flows in Enright *et al.* [40], as well as in computer graphics [96, 39]. It was developed for the Poisson and the diffusion equations on irregular domains with Dirichlet boundary conditions and their applications in Gibou *et al.* [47, 45, 44, 46]. In what follows, we describe the algorithms, point out common misconceptions and describe how to properly implement those methods. We also note that several

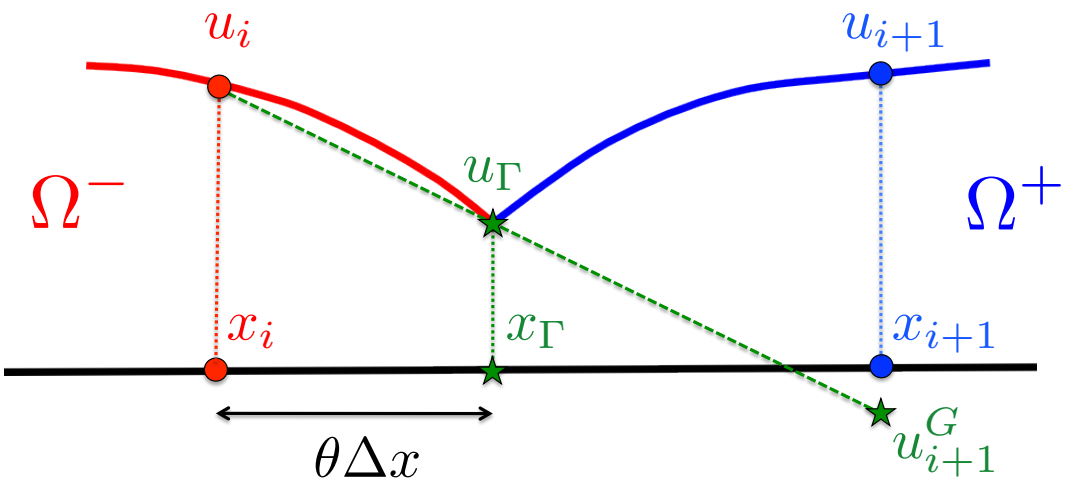


Figure 4: Definition of the ghost value u^G_{i+1} using a linear extrapolation. First, construct a linear interpolant $\tilde{u}(x) = ax + b$ of u such that $\tilde{u}(0) = u_i$ and $\tilde{u}(\theta\Delta x) = u_\Gamma$. Then define $u^G_{i+1} = \tilde{u}(\Delta x)$. (Color online).

authors have proposed both different and similar approaches to these types of problems, following the pioneer work of Shortley and Weller; see [84, 129, 75, 16, 152, 62, 61, 109, 107, 108, 168] and the references therein. Finally, we point out that the definition of the ghost nodes does not seek to impose conservative properties at the discrete level and therefore offer more flexibility is designing numerical schemes. It is important to note that, while conservation properties are necessary in the design of numerical methods for nonlinear hyperbolic conservation laws to guarantee the correct speed of propagation (Rankine-Hugoniot jump condition) where shocks are present, this is not the case for Elliptic and Parabolic equations. Therefore, even though the equations we seek to solved are based on conservation laws, approximating this condition (as opposed to enforcing it at the discrete level) is often sufficient and allows much flexibility to design accurate, simple and efficient schemes. We also refer the interested reader to a conservative Ghost-Fluid method for the study of detonation waves [98].

The diffusion equation (1) is discretized in time by the Crank-Nicolson scheme[1]:

$$u^{n+1} - \frac{1}{2}\Delta t \widetilde{\nabla\cdot(\beta\nabla u)}^{n+1} = u^n + \frac{1}{2}\Delta t \widetilde{\nabla\cdot(\beta\nabla u)}^{n} + \frac{1}{2}\Delta t\left(S^n + S^{n+1}\right),$$

where Δt is the time step and $\widetilde{\nabla\cdot(\beta\nabla u)}^{n}$ and $\widetilde{\nabla\cdot(\beta\nabla u)}^{n+1}$ are the spatial approximations of $\nabla\cdot(\beta\nabla u)$ at time t^n and t^{n+1}, respectively. The discretization of the spatial operator, including the special treatments needed at the interface, is performed in a dimension-by-dimension fashion. Therefore, without loss of generality, we only describe the discretization for the one-dimensional diffusion equation:

$$\frac{\partial u}{\partial t} = \frac{\partial}{\partial x}\left(\beta\frac{\partial u}{\partial x}\right) + S,$$

with a Dirichlet boundary condition of $u(x) = u_\Gamma(x)$ on the interface Γ.

The computational domain is discretized into cells of size Δx, with the grid nodes x_i located at the cells' center. The cell edges are referred to as faces, and the two faces bounding the grid node x_i are located at $x_{i\pm\frac{1}{2}}$. The numerical solution of the diffusion equation is computed at the grid nodes and is denoted by $u_i = u(x_i, t^n)$, where $t^n = n\Delta t$. Using second-order accurate central difference formulas for discretizing the

[1]For stiff problems, one may prefer the first-order accurate implicit Euler method.

spatial operator, the full discretization is written as:

$$u_i^{n+1} - \frac{1}{2}\Delta t \frac{\beta_{i+\frac{1}{2}}(\frac{u_{i+1}^{n+1}-u_i^{n+1}}{\Delta x}) - \beta_{i-\frac{1}{2}}(\frac{u_i^{n+1}-u_{i-1}^{n+1}}{\Delta x})}{\Delta x} = u_i^n + \frac{1}{2}\Delta t \frac{\beta_{i+\frac{1}{2}}(\frac{u_{i+1}^{n}-u_i^{n}}{\Delta x}) - \beta_{i-\frac{1}{2}}(\frac{u_i^{n}-u_{i-1}^{n}}{\Delta x})}{\Delta x} + \frac{1}{2}\Delta t(S_i^n + S_i^{n+1}). \quad (8)$$

In order to avoid differentiating the fluxes across the interface, where the solution presents a kink, a ghost value is used. Referring to figure 4, let x_Γ be an interface point between the grid points x_i and x_{i+1}, with a Dirichlet boundary condition of u_Γ^n at time t^n and of u_Γ^{n+1} at time t^{n+1}, applied at x_Γ. We define the ghost values $\left(u_{i+1}^n\right)^G$ and $\left(u_{i+1}^{n+1}\right)^G$ at x_{i+1} across the interface at time t^n and t^{n+1}, respectively, and rewrite equation (8) as:

$$u_i^{n+1} - \frac{1}{2}\Delta t \frac{\beta_{i+\frac{1}{2}}(\frac{\left(u_{i+1}^{n+1}\right)^G-u_i^{n+1}}{\Delta x}) - \beta_{i-\frac{1}{2}}(\frac{u_i^{n+1}-u_{i-1}^{n+1}}{\Delta x})}{\Delta x} = u_i^n + \frac{1}{2}\Delta t \frac{\beta_{i+\frac{1}{2}}(\frac{\left(u_{i+1}^{n}\right)^G-u_i^{n}}{\Delta x}) - \beta_{i-\frac{1}{2}}(\frac{u_i^{n}-u_{i-1}^{n}}{\Delta x})}{\Delta x} + \frac{1}{2}\Delta t(S_i^n + S_i^{n+1}). \quad (9)$$

The ghost values $\left(u_{i+1}^n\right)^G$ and $\left(u_{i+1}^{n+1}\right)^G$ are defined by first constructing an interpolant $\tilde{u}^n(x)$ of u^n at time t^n and another interpolant $\tilde{u}^{n+1}(x)$ of u^{n+1} at time t^{n+1} on the left of the interface, such that $\tilde{u}^n(0) = u_i^n$, $\tilde{u}^{n+1}(0) = u_i^{n+1}$, and then defining $\left(u_{i+1}^n\right)^G = \tilde{u}^n(\Delta x)$ and $\left(u_{i+1}^{n+1}\right)^G = \tilde{u}^{n+1}(\Delta x)$. Figure 4 illustrates the definition of the ghost cells in the case of a linear extrapolation. Linear, quadratic and cubic extrapolations are defined by[2]:

Linear Extrapolation: Take $\tilde{u}^{n+1}(x) = ax + b$ with:

- $\tilde{u}^{n+1}(0) = u_i^{n+1}$,
- $\tilde{u}^{n+1}(\theta\Delta x) = u_\Gamma^{n+1}$.

Quadratic Extrapolation: Take $\tilde{u}^{n+1}(x) = ax^2 + bx + c$ with:

- $\tilde{u}^{n+1}(-\Delta x) = u_{i-1}^{n+1}$,
- $\tilde{u}^{n+1}(0) = u_i^{n+1}$,
- $\tilde{u}^{n+1}(\theta\Delta x) = u_\Gamma^{n+1}$.

Cubic Extrapolation: Take $\tilde{u}^{n+1}(x) = ax^3 + bx^2 + cx + d$ with:

- $\tilde{u}^{n+1}(-2\Delta x) = u_{i-2}^{n+1}$,
- $\tilde{u}^{n+1}(-\Delta x) = u_{i-1}^{n+1}$,
- $\tilde{u}^{n+1}(0) = u_i^{n+1}$,
- $\tilde{u}^{n+1}(\theta\Delta x) = u_\Gamma^{n+1}$.

In these equations, $\theta \in [0, 1]$ refers to the cell fraction occupied by the subdomain Ω^-. The construction of $\tilde{u}^n$ is similar, with the solution u and the boundary condition u_Γ taken at time t^n instead of time t^{n+1}. Similar constructions define $\left(u_i^{n+1}\right)^G$ and $(u_i^n)^G$ using values to the right of x_{i+1}. Equation (9) gives a linear system for u^{n+1}. Likewise, the interface location (and therefore θ) is found by first constructing a linear or higher-order interpolant of the level-set function ϕ and then finding the zero of the interpolant. Note that the quadratic extrapolation is equivalent to the Shortley-Weller method [129].

Remarks:

- The approximation of the Poisson equation follows trivially from that of the diffusion equation.

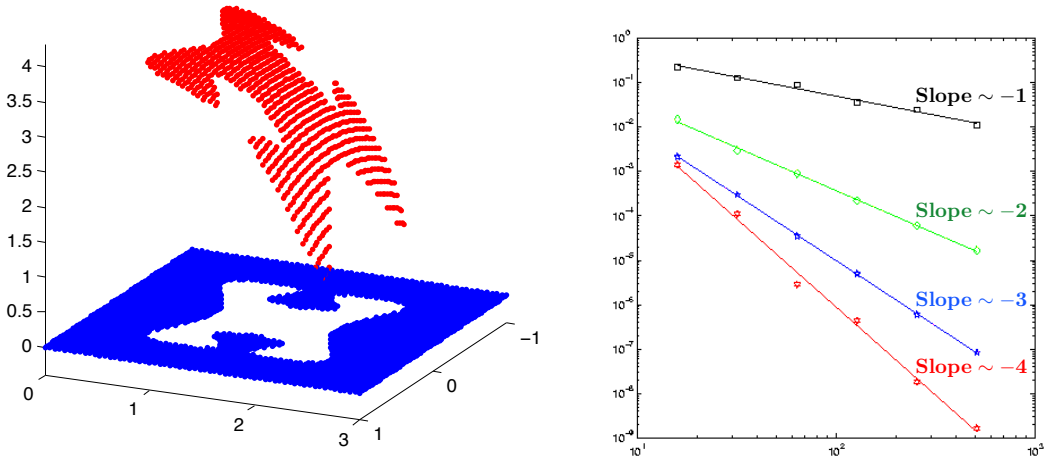
[2]One may prefer a Newton's form for constructing the interpolant $\tilde{u}(x)$.

- The interpolation formulas for the construction of the different extrapolations are not well-behaved if θ is too small. However, in this case, the interface Γ is close to a grid point, say $\mathbf{x}^*$, which in turn asserts that the solution is close to the known boundary condition $u_\Gamma(\mathbf{x}^*)$. Therefore, if the interface is too close to a grid point $\mathbf{x}^*$, then we simply define the solution at $\mathbf{x}^*$ as $u(\mathbf{x}^*) = u_\Gamma(\mathbf{x}^*)$. The heuristic we have used is to do so if $\theta < \Delta x$ in the case of linear extrapolations; if $\theta < \Delta x^2$ in the case of quadratic extrapolations; and if $\theta < \Delta x^3$ in the case of cubic extrapolations.
- In the case where not enough grid points are available to construct an interpolant, a lower degree interpolant is built. We refer the reader to the numerical tests sections for a discussion on the influence of lower extrapolations on the overall accuracy.
- In the case where the interface crosses to the left and right of a grid point, the interface boundary condition to the left and right are both used in the construction of the interpolant.
- In the case where third- or fourth-order accuracy is desired, the second-order central differencing used in equations (8) and (9) are replaced by the standard fourth-order accurate central differencing (see [45]).

3.1 Order of Accuracy and Common Misconceptions

We briefly present the typical accuracy that can be expected for the Poisson and the diffusion equations on irregular domains depending on the order of extrapolations. We use a conjugate gradient with incomplete Cholesky in the case where the linear system is symmetric and a BiCGSTAB with an ILU preconditioning in the case where the linear system is non-symmetric [49, 120]. We then turn our attention to common misconceptions and pitfalls in implementing this approach.

3.1.1 Typical Results for the Poisson Equation



(a) Solution profile. (b) Accuracy.

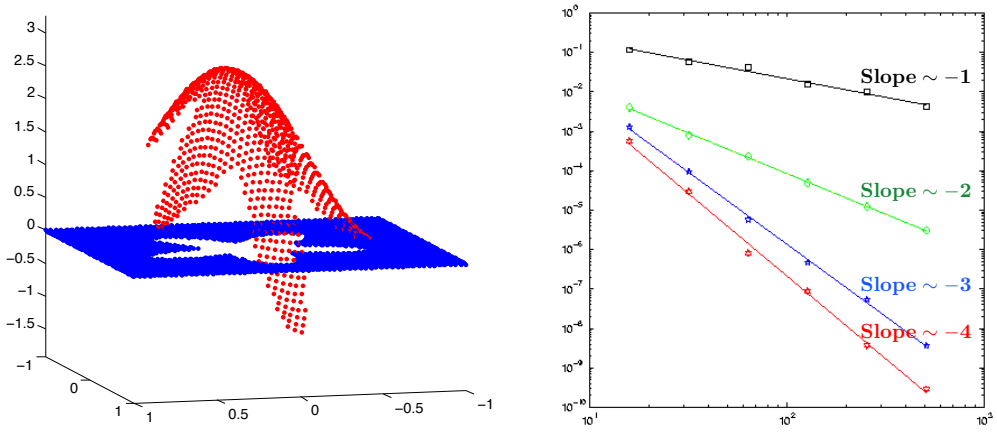
Figure 5: Typical results for the Poisson equations with Dirichlet boundary conditions. The results are for example 3.1.1. (a) The computed solution inside Ω^- (red) is decoupled from the computed solution outside (blue). (b) Loglog plot of the error in the L^∞-norm for constant (black), linear (green), quadratic (blue) and cubic (red) extrapolations. (Color online).

Consider the Poisson equation (2) on $\Omega = [-1, 1] \times [0, 3]$ with an exact solution of $u = 5 - \exp(.5(1 - t)(x^2 + y^2 - \frac{\pi^2}{25}))$. The interface is parametrized by:

$$\begin{cases} x(\alpha) &= .6\cos(\alpha) - .3\cos(3\alpha) \\ y(\alpha) &= 1.5 + .7\sin(\alpha) - .07\sin(3\alpha) + .2\sin(7\alpha) \end{cases},$$

where $\alpha \in [0, 2\pi]$. The numerical solution is illustrated in figure 5(a) and the accuracy using different extrapolations is depicted in figure 5(b). The order of accuracy is typically ~ 1 for constant extrapolations, ~ 2 for linear extrapolations, ~ 3 for quadratic extrapolations and ~ 4 for cubic extrapolations[3]. The reduction in accuracy for some resolutions is due to how many grid points are available to construct the interpolant and thus how the ghost values are defined.

3.1.2 Typical Results for the Diffusion Equation



(a) Solution profile. (b) Accuracy.

Figure 6: Typical results for the diffusion equations with Dirichlet boundary conditions. The results are for example 3.1.2. (a) The computed solution inside Ω^- (red) is decoupled from the computed solution outside (blue). (b) Loglog plot of the error in the L^∞-norm for constant (black), linear (green), quadratic (blue) and cubic (red) extrapolations. (Color online).

Consider the diffusion equation, equation (1), on $\Omega = [-1, 1] \times [-1, 1]$ with an exact solution of $u = \sin(\pi x) + \sin(\pi y) + \cos(\pi x) + \cos(\pi y) + x^6 + y^6$. The interface is star shaped, given by the set of points where $\phi = r - 0.5 - \frac{y^5 + 5x^4 y - 10x^2 y^3}{3r^5} = 0$, and $r = \sqrt{x^2 + y^2}$. The numerical solution is illustrated in figure 6(a), and the accuracy using different extrapolations is depicted in figure 6(b). The order of accuracy is typically ~ 1 for constant extrapolations, ~ 2 for linear extrapolations, ~ 3 for quadratic extrapolations and ~ 4 for cubic extrapolations.

3.1.3 Nature of Linear Systems and Accuracy on Gradients

In [45], it was shown that defining the ghost point $\left(u^{n+1}\right)^G$ by a linear extrapolation produces a symmetric linear system and that the linear system is non-symmetric for higher-order extrapolations. Also, the degree of the interpolation is important for the accuracy of the method. We refer the interested reader to Ng *et al.* [95], which concluded that a linear interpolation produces second-order accurate solutions and first-order accurate gradients, while a quadratic extrapolation produces second-order accurate solutions and second-order accurate gradients. This was first observed in [85]. We note that the location of the interface must also be found using a quadratic interpolation of the level-set function in the vicinity of the interface if second-order accurate gradients are to be calculated. Figure 7 demonstrates that the error of the gradient is largest close to the interface regardless of the order of interpolation for the interface location and extrapolation for the ghost values. This will be part of the reasons why adaptive grids where smaller cells are located near the interface are desirable (see section 5). Finally, the condition number of the linear system is affected by the choice of definition of the ghost values. Figure 8 depicts the typical trend. In this work we use a PCG (symmetric case) and a BiCGSTAB (non-symmetric case) solvers.

[3]We use a time step of $\Delta t = \Delta x^{3/2}$ and $\Delta t = \Delta x^2$ to emulate a third- and a fourth-order scheme in time.

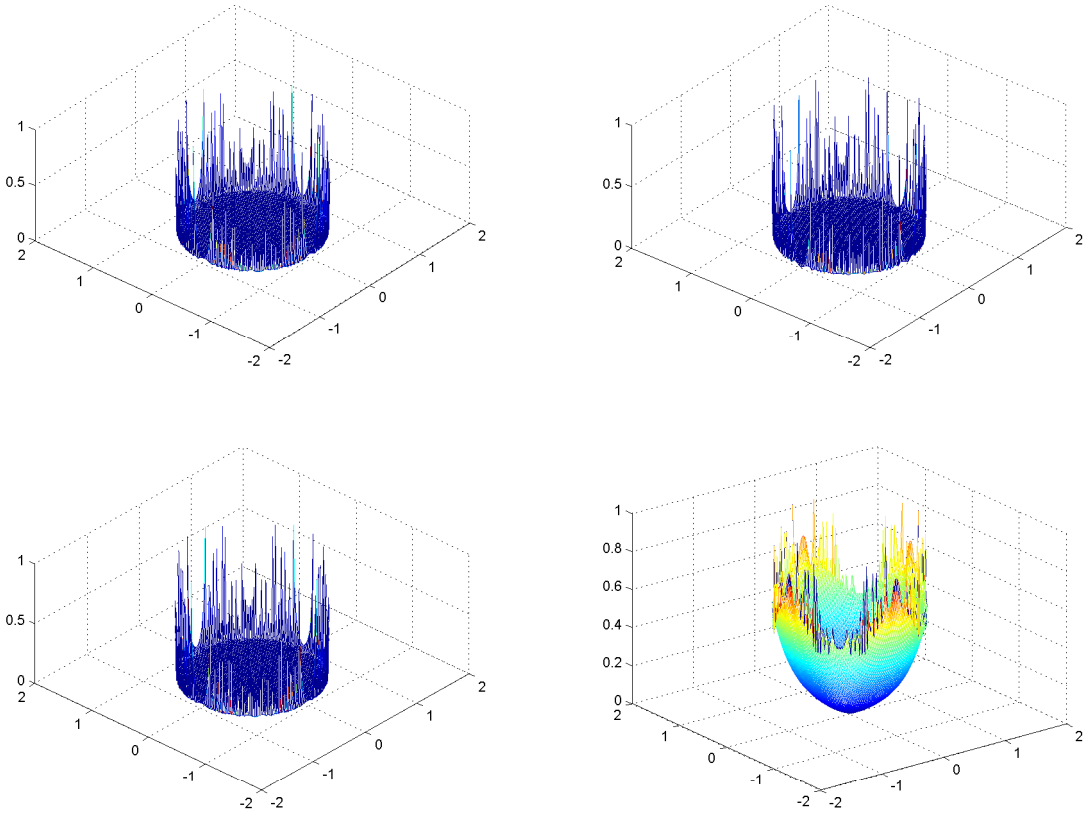


Figure 7: Typical error for the gradients of the solution in the L^∞ norm. The ghost cell values are defined by linear extrapolation of the solution in the top figures and by quadratic extrapolation of the solution in the bottom figures. The interface location is found by linear interpolation of ϕ in the left figures and by quadratic interpolation of ϕ in the right figures. Note that the errors are normalized.

3.1.4 Importance of Time Dependent Boundary Conditions

It is important to note that the boundary condition at the interface must be imposed at the appropriate time. I.e. we set $u_\Gamma^{n+1} = u_\Gamma(\mathbf{x}, t^{n+1})$ when building the linear system and set $u_\Gamma^n = u_\Gamma(\mathbf{x}, t^n)$ when evaluating the right-hand-side of equation (9). Setting the boundary condition as $u_\Gamma^{n+1} = u_\Gamma(\mathbf{x}, t^n)$ in the linear system introduces a lagging in time (i.e. a first order perturbation) and thus a drop in the accuracy from second-order to first-order. We propose here an example and refer the interested reader to [45] for a discussion on the influence of perturbations in the location of the boundary condition on the accuracy of the method.

Consider an irregular domain, Ω^-, described in polar coordinates as:

$$\begin{cases} x(\theta) &= 0.02\sqrt{5} + (0.5 + 0.2\sin(5\theta))\cos(\theta) \\ y(\theta) &= 0.02\sqrt{5} + (0.5 + 0.2\sin(5\theta))\sin(\theta) \end{cases},$$

where $\theta \in [0, 2\pi]$, and an exact solution of $u = \exp(-t + x + y)$ in Ω^- and $u = 0$ in Ω^+. The right-hand-side S in equation (1) is defined accordingly. We solve the diffusion equation to a final time of $t = .1$, in this case defining the ghost cell by linear extrapolation. Table 1(a) shows that this treatment produces second-order accurate solution in the L^∞-norm. In contrast, if the boundary condition is imposed as $u_\Gamma = u_\Gamma(t^n, \mathbf{x})$, table 1(b) shows that the solution process drops from second-order accuracy to first-order accuracy.

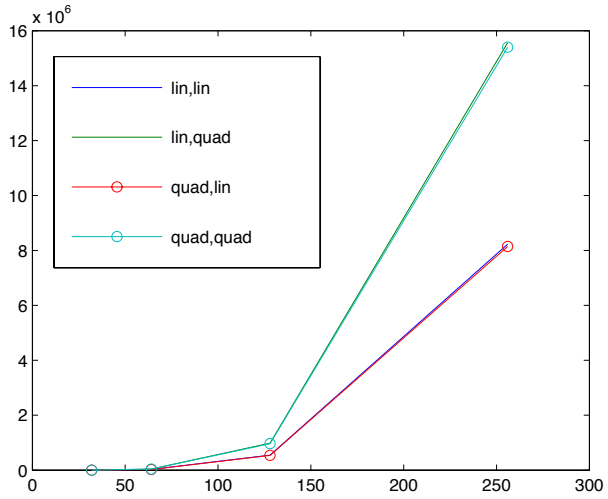


Figure 8: Condition number versus the grid size for a typical two-dimensional Poisson solver in irregular domains. The four curves illustrate the impact of the extrapolation used to define the ghost values (first parameter in the legend's caption) and the order of the interpolation for finding the interface location (second parameter). The two (superimposed) curves with the smallest condition numbers are associated with the linear extrapolation for defining the ghost cells. (Color online).

(a) Correctly imposing the boundary condition at t^{n+1}.

Grid	$\|\|u-u_h\|\|_1$	Order	$\|\|u-u_h\|\|_\infty$	Order
32^2	0.0001061	–	0.0002057	–
64^2	4.039×10^{-5}	1.39	7.949×10^{-5}	1.37
128^2	8.959×10^{-6}	2.17	1.955×10^{-5}	2.02
256^2	2.344×10^{-6}	1.93	4.766×10^{-6}	2.04

(b) Incorrectly imposing the boundary condition at t^n.

Grid	$\|\|u-u_h\|\|_1$	Order	$\|\|u-u_h\|\|_\infty$	Order
32^2	0.01892	–	0.04412	–
64^2	0.01048	0.852	0.02450	0.848
128^2	0.005483	0.934	0.01289	0.926
256^2	0.002823	0.958	0.006569	0.973

Table 1: (a) Linear extrapolation definition of the ghost cell producing second-order accuracy in the L^∞-norm. (b) The boundary condition is incorrectly imposed at time t^n instead of t^{n+1} leading to first-order accuracy in the L^∞-norm.

3.1.5 The Dimension-by-Dimension Framework

One of the advantages of the Ghost-Fluid Method is the ability to define the ghost values in a dimension-by-dimension framework. This process is illustrated in two spatial dimensions in figure 9, where two ghost values G_x and G_y need to be defined in the x- and y- directions, respectively. A misconception set forth in [165] is that the values G_x and G_y need to be the same. Imposing this assumption, the authors conclude that multidimensional extrapolations are necessary, which in turn reduces the computational efficiency of the method. This assumption, however, is incorrect. The two values G_x and G_y may be different, can be computed *independently* in a dimension-by-dimension framework and only require one-dimensional extrapolation procedures.

3.1.6 Influence of High Frequency Modes

In [165], Zhang and Liu pose a proof that incorrectly claims that the methods in Gibou *et al.* [47] would give lower order accuracy on certain types of problems and set forth an example problem where [47] should then obtain lower order accuracy. Although their conclusions are incorrect, they nonetheless point out interesting facts about the behavior of the ghost-fluid method for parabolic and elliptic problems.

According to [165], the drop in the order of accuracy in the method of Gibou *et al.* [47, 45] can be observed by considering numerical examples where the solution contains high frequencies. However, in what

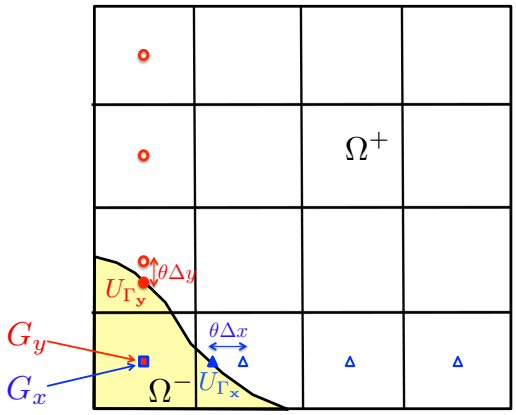


Figure 9: Procedure to define the ghost cells in two spatial dimensions, for each spatial direction *independently*: First construct a *one-dimensional* interpolant $\tilde{u}(\xi)$ using the boundary condition at the interface and the value of u at as many other grid nodes as it is necessary to achieve the desired accuracy. Then, define $u^G_{i+1} = \tilde{u}(\Delta\xi)$, where $\Delta\xi$ is the distance between two adjacent grid nodes in the spatial direction considered. In the schematic above, the ghost value G_x, used in the discretizations in the x-direction, is constructed using the interface value U_{Γ_x} and a subset of the values of u at the blue triangles' locations; whereas the ghost value G_y, used in the discretizations in the y-direction is, constructed using the interface value U_{Γ_y} and a subset of the values of u at the red circles' locations. (Color online).

follows, we present numerical evidence that the order of accuracy of the methods proposed in [47, 45] are consistent with the conclusions of the authors in the case of the numerical tests of [165].

Consider an irregular domain defined by a disk centered at the origin with radius $r = \pi/5$ and an exact solution defined as:

$$u = \begin{cases} \exp\left(a(1+ct)(\mathbf{x}\cdot\mathbf{x} - r^2)\right) - 1 & \mathbf{x}\cdot\mathbf{x} > r^2 \\ 0 & \mathbf{x}\cdot\mathbf{x} \leq r^2 \end{cases}, \tag{10}$$

with $c = .01$ and $a = .5$. The source term S in equation (1) is derived accordingly. The diffusion coefficient β is taken to be constant and equal to $\beta = 10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}$ and 10^{-6}. Two types of extrapolations for defining the ghost values are considered: linear and quadratic. The initial and final times are taken to be $t_{\text{initial}} = 20$ and $t_{\text{final}} = 21$, respectively. Tables 2 through 4 give the results obtained in the case of linear extrapolations, while tables 5 through 7 give the results in the case of quadratic extrapolations. It is clear that the method with linear extrapolation is second-order accurate in the L^∞-norm, while the method with quadratic extrapolation is third-order accurate in the L^∞-norm, as stated in [47, 45]. We draw attention to the fact that accuracy analyses based on a Taylor-type expansion, as in [165], can be misleading. Indeed, Taylor-type analyses can indicate the minimum order of accuracy for a method, but cannot be used to conclude the highest achievable order of accuracy, as pointed out in [82, 71, 45].

(a) Linear extrapolation - $\beta = 10^{-1}$

Grid	$\|\|u - u_h\|\|_1$	Order	$\|\|u - u_h\|\|_\infty$	Order
32^2	0.0002446	–	0.0004627	–
64^2	4.137×10^{-5}	2.56	0.0001310	1.82
128^2	1.484×10^{-5}	1.48	3.739×10^{-5}	1.81
256^2	3.893×10^{-6}	1.93	9.198×10^{-6}	2.02
512^2	6.805×10^{-7}	2.52	2.177×10^{-6}	2.08

(b) Linear extrapolation - $\beta = 10^{-2}$

Grid	$\|\|u - u_h\|\|_1$	Order	$\|\|u - u_h\|\|_\infty$	Order
32^2	0.0001050	–	0.0004459	–
64^2	1.868×10^{-5}	2.49	0.0001290	1.79
128^2	6.016×10^{-6}	1.63	3.692×10^{-5}	1.81
256^2	1.556×10^{-6}	1.95	9.133×10^{-6}	2.02
512^2	2.741×10^{-7}	2.51	2.172×10^{-6}	2.07

Table 2: Error norms for the example of section 3.1.6.

(a) Linear extrapolation - $\beta = 10^{-3}$

Grid	$\|\|u-u_h\|\|_1$	Order	$\|\|u-u_h\|\|_\infty$	Order
32^2	4.110×10^{-5}	–	0.0003302	–
64^2	8.355×10^{-6}	2.30	0.0001186	1.48
128^2	2.138×10^{-6}	1.97	3.491×10^{-5}	1.76
256^2	5.327×10^{-7}	2.00	8.887×10^{-6}	1.97
512^2	9.763×10^{-8}	2.45	2.160×10^{-6}	2.04

(b) Linear extrapolation - $\beta = 10^{-4}$

Grid	$\|\|u-u_h\|\|_1$	Order	$\|\|u-u_h\|\|_\infty$	Order
128^2	7.84×10^{-7}	–	2.48×10^{-5}	–
256^2	1.81×10^{-7}	2.11	7.87×10^{-6}	1.66
512^2	3.59×10^{-8}	2.34	2.10×10^{-6}	1.90
1024^2	9.29×10^{-9}	1.95	5.63×10^{-7}	1.90
2048^2	2.243×10^{-9}	2.05	1.397×10^{-7}	2.01

Table 3: Error norms for the example of section 3.1.6.

(a) Linear extrapolation - $\beta = 10^{-5}$

Grid	$\|\|u-u_h\|\|_1$	Order	$\|\|u-u_h\|\|_\infty$	Order
32^2	1.01×10^{-6}	–	1.16×10^{-5}	–
64^2	3.66×10^{-7}	1.47	8.72×10^{-6}	0.420
128^2	1.89×10^{-7}	0.956	8.68×10^{-6}	0.006
256^2	5.86×10^{-8}	1.69	4.77×10^{-6}	0.86
512^2	1.28×10^{-8}	2.19	1.69×10^{-6}	1.49
1024^2	3.16×10^{-9}	2.02	5.18×10^{-7}	1.71
2048^2	7.77×10^{-10}	2.02	1.31×10^{-7}	1.98

(b) Linear extrapolation - $\beta = 10^{-6}$

Grid	$\|\|u-u_h\|\|_1$	Order	$\|\|u-u_h\|\|_\infty$	Order
32^2	2.99×10^{-6}	–	2.17×10^{-5}	–
64^2	1.21×10^{-6}	1.30	1.56×10^{-5}	0.469
128^2	5.67×10^{-7}	1.10	1.28×10^{-5}	0.288
256^2	2.17×10^{-7}	1.38	5.58×10^{-6}	1.20
512^2	8.47×10^{-8}	1.36	1.83×10^{-6}	1.60
1024^2	2.451×10^{-8}	1.79	5.23×10^{-7}	1.81
2048^2	5.918×10^{-9}	2.05	1.30×10^{-7}	2.01

Table 4: Error norms for the example of section 3.1.6.

Remarks: Some trends, pointed out in [165], are interesting. What can be observed from tables 2-7 is that the smaller the diffusion coefficient β, the finer resolution is needed to reach the asymptotic regime. For example, table 2(a) indicates that the asymptotic regime is reached for grids 256^2 and finer in the case where $\beta = 10^{-1}$, while table 7(b) shows that a much finer grid of 2048^2 is needed in the case of $\beta = 10^{-6}$. However, this trend is natural. A small diffusion coefficient, β, in this problem means that the effect of the source term S dominates and, in order to see the effects of diffusion, one needs more accuracy, more precision and thus smaller grid sizes. This is quite similar to turbulence modeling where, to accurately model small viscosity, one needs incredibly fine grids, which are beyond current computational resources. In fact, researchers in turbulence do not claim to, or even try to, accurately simulate such a small viscosity. They instead model it by either adding special tensors or changing the way the convection term (related to the source term S in the present paper) is treated (see the LES discussion in [158]).

In [165], the authors conclude that a ghost-fluid approach should be avoided for simulating the Navier-Stokes equations for very small viscosity, because the asymptotic regime requires computationally intractable fine grids. However, in our view, their conclusions are misleading. First, one should note that even on very coarse grids, for which the asymptotic regime is not reached at all, the maximum error coming from the diffusion part is quite small. For example, table 4(b) gives a maximum error on the order of 10^{-5} on a 64×64 grid in the case of a linear extrapolation, while table 7(b) indicates a maximum error of the order of 10^{-8} in the case of a quadratic extrapolation. Second, in the case of the Navier-Stokes equations, the numerical errors induced by the approximations of the momentum term and the treatment of the incompressibility

(a) Quadratic extrapolation - $\beta = 10^{-1}$

Grid	$\|\|u-u_h\|\|_1$	Order	$\|\|u-u_h\|\|_\infty$	Order
32^2	2.233×10^{-6}	–	1.638×10^{-5}	–
64^2	6.252×10^{-7}	1.84	3.519×10^{-6}	2.22
128^2	3.369×10^{-8}	4.21	3.640×10^{-7}	3.27
256^2	6.004×10^{-9}	2.49	4.478×10^{-8}	3.02
512^2	5.898×10^{-10}	3.35	5.187×10^{-9}	3.11

(b) Quadratic extrapolation - $\beta = 10^{-2}$

Grid	$\|\|u-u_h\|\|_1$	Order	$\|\|u-u_h\|\|_\infty$	Order
32^2	2.030×10^{-6}	–	1.632×10^{-5}	–
64^2	3.094×10^{-7}	2.71	3.434×10^{-6}	2.25
128^2	1.922×10^{-8}	4.01	3.628×10^{-7}	3.24
256^2	2.606×10^{-9}	2.88	4.468×10^{-8}	3.02
512^2	2.679×10^{-10}	3.28	5.068×10^{-9}	3.14

Table 5: Error norms for the example of section 3.1.6.

(a) Quadratic extrapolation - $\beta = 10^{-3}$

Grid	$\|\|u - u_h\|\|_1$	Order	$\|\|u - u_h\|\|_\infty$	Order
32^2	1.495×10^{-6}	–	1.427×10^{-5}	–
64^2	1.663×10^{-7}	3.17	2.734×10^{-6}	2.38
128^2	1.328×10^{-8}	3.65	3.503×10^{-7}	2.96
256^2	1.154×10^{-9}	3.53	4.405×10^{-8}	2.99
512^2	1.300×10^{-10}	3.15	5.622×10^{-9}	2.97

(b) Quadratic extrapolation - $\beta = 10^{-4}$

Grid	$\|\|u - u_h\|\|_1$	Order	$\|\|u - u_h\|\|_\infty$	Order
128^2	7.612×10^{-9}	–	2.707×10^{-7}	–
256^2	6.385×10^{-10}	3.58	4.052×10^{-8}	2.74
512^2	6.649×10^{-11}	3.26	8.556×10^{-9}	2.24
1024^2	5.140×10^{-12}	3.69	7.496×10^{-10}	3.51
2048^2	5.555×10^{-13}	3.21	9.177×10^{-11}	3.03

Table 6: Error norms for the example of section 3.1.6.

(a) Quadratic extrapolation - $\beta = 10^{-5}$

Grid	$\|\|u - u_h\|\|_1$	Order	$\|\|u - u_h\|\|_\infty$	Order
32^2	4.95×10^{-8}	–	5.22×10^{-7}	–
64^2	1.01×10^{-8}	2.29	1.99×10^{-7}	1.39
128^2	2.30×10^{-9}	2.13	8.76×10^{-8}	1.19
256^2	2.98×10^{-10}	2.95	2.43×10^{-8}	1.85
512^2	3.18×10^{-11}	3.23	4.89×10^{-9}	2.31
1024^2	2.98×10^{-12}	3.42	6.63×10^{-10}	2.88
2048^2	3.22×10^{-13}	3.21	8.46×10^{-11}	2.97

(b) Quadratic extrapolation - $\beta = 10^{-6}$

Grid	$\|\|u - u_h\|\|_1$	Order	$\|\|u - u_h\|\|_\infty$	Order
32^2	5.04×10^{-9}	–	5.36×10^{-8}	–
64^2	1.06×10^{-9}	2.25	2.16×10^{-8}	1.31
128^2	3.10×10^{-10}	1.77	1.38×10^{-8}	0.65
256^2	6.43×10^{-11}	2.27	5.79×10^{-9}	1.25
512^2	1.07×10^{-11}	2.59	1.81×10^{-9}	1.68
1024^2	1.51×10^{-12}	2.82	4.65×10^{-10}	1.96
2048^2	1.98×10^{-13}	2.93	6.27×10^{-11}	2.89

Table 7: Error norms for the example of section 3.1.6.

condition are likely to dwarf the error produced by the viscous term. In addition, a viscosity of the order of 10^{-6} corresponds to highly turbulent flows, for which extremely fine grids are required to capture the small length scale of the problem. In fact, in those regimes, a turbulence model would use coarser grids, as discussed above, and the error produced by the model itself would dominate the treatment of the viscous term.

4 A Level-Set Approach to the Stefan Problem

In [46], Gibou *et al.* presented a methodology based on the level-set method and the ghost-fluid method to solve the Stefan problem. In [29], Chen *et al.* had earlier proposed a similar methodology, except that the treatment of the boundary conditions was different. Kim *et al.* applied that framework to the simulation of the solidification processes in [70]. We note that the first level-set approach to solve the Stefan problem was given in Sethian and Strain [126]. In that work, the diffusion equation was solved using a boundary integral approach. Other authors have proposed successful approaches to both the Stefan problem and its extension to the solidification of binary alloys [160, 162, 151, 167, 142, 164, 141, 7, 52]. In the case of the Stefan problem, the two main ingredients are:

1. Solving the equations in (3) for the temperature field T on both side of the free boundary, while imposing at the front the Dirichlet boundary condition given by the Gibbs-Tompson condition (4).

2. Capturing the interface motion using the level-set equation (6) with a given velocity field defined by equation (5).

However, one additionally needs to ensure that valid values of the solutions on each side of the interface are defined in the appropriate domains. This is done with extrapolation procedures, following Aslam [8] and described in 4.2. Also, in the case of the design of high-order accurate schemes, it is necessary to guarantee that time evolution procedures are adequate; this will be described in section 4.3. Finally, reinitialization schemes are needed in the framework of the level-set method, and care must be taken to guarantee their proper behavior. We point out common ill-treatments and their fixes in section 4.4.

4.1 Algorithm to Solve the Stefan Problem

The methodology introduced in section 3 to solve the diffusion equation can be applied independently to each of the subdomains, Ω^- and Ω^+, since their respective solutions can be decoupled using the ghost-fluid approach and the Dirichlet boundary condition (4). Therefore two copies of the temperature, T_s^n and T_l^n, are defined on every grid node of the computational domain Ω. They represent the temperature at time step t^n in the solid region, Ω^-, and in the liquid region, Ω^+, respectively. Then the diffusion equations in (3) are solved in both Ω^- and Ω^+ with the new interface location given by the zero-contour of ϕ at time t^{n+1}. Dirichlet boundary conditions are imposed on the interface using the Gibbs-Thomson relation in equation (4). When computing the Gibbs-Thomson relation, we use the value of the normal velocity $\mathbf{V} \cdot \mathbf{n}$ at time t^n, but the interface curvature κ is computed at time t^{n+1} to reflect the updated morphology of the front. On the boundary of the computational domain, $\partial\Omega$, either Dirichlet or Neumann boundary conditions can be imposed.

As detailed in [46, 45] and illustrated in figure 10, the interface may sweep some grid nodes from time t^n to t^{n+1}, so the temperature at these nodes needs to be extrapolated to define a valid right-hand-side in the Crank-Nicholson formula (9). Also, as noted in [1, 2], the interface's velocity, given by equation (5), is only valid exactly *at* the interface. However, in the discretization of the level-set equation, equation (6), a valid velocity field is required at the nodes in a small band near the interface. Therefore, the velocity field must be extended to the nodes in a small band on each side of the interface by constant extrapolation in the normal direction. The rationale for extrapolations in the normal direction is based on the fact that the interface propagates only in its normal direction[4]. The extrapolation procedures we use are those of [8], detailed in section 4.2. The procedure for solving the Stefan problem follows the algorithm given in algorithm 1.

Algorithm 1 : Procedure to Solve the Stefan Problem

1. Initialize ϕ as a signed distance function,
2. Initialize T_s^0 in Ω^-, and T_l^0 in Ω^+ at t^0,
3. **while** (the final time is not reached)
4. $\quad t^n := t^{n+1}$,
5. $\quad$ Quadratically extrapolate, in the normal direction, T_s^n from Ω^- to Ω^+ and T_l^n from Ω^+ to Ω^-,
6. $\quad$ Calculate the velocity V at grid nodes and constantly extrapolate it in a band around Γ,
7. $\quad$ Evolve the interface by solving equation (6) for ϕ^{n+1}, and reinitialize using (7),
8. $\quad$ Solve the diffusion equations in Ω^- and Ω^+ for T_s^{n+1} and T_l^{n+1}, using the Gibbs-Thomson relation (4) as the Dirichlet boundary condition on Γ,

end while

4.2 High-Order Extrapolation - Aslam's Technique

As mentioned in section 4.1, it is necessary to extrapolate scalar quantities across an interface in the normal direction. In the case of defining the velocity field in a band around the interface, a constant extrapolation procedure is sufficient. However, in the case of defining a valid right-hand-side for equation (9), high-order extrapolations are necessary. Such high-order extrapolations in the normal direction are performed in a series of steps, as proposed in Aslam [8]. For example, suppose that one needs to generate a cubic extrapolation of a scalar quantity Q from the region where $\phi \leq 0$ to the region where $\phi > 0$. The procedure is to first compute $Q_{nnn} = \nabla \left(\nabla \left(\nabla u \cdot \mathbf{n} \right) \cdot \mathbf{n} \right) \cdot \mathbf{n}$ in the region $\phi \leq 0$ and then extrapolate it across the interface in a constant fashion by solving the following partial differential equation:

$$\frac{\partial Q_{nnn}}{\partial \tau} + H(\phi + \texttt{offset}) \nabla Q_{nnn} \cdot \mathbf{n} = 0,$$

where H is the Heaviside function and `offset` accounts for the fact that Q_{nnn} is not numerically well-defined in the region where $\phi \geq$ `offset`. Typically, in the case where Q_{nnn} is computed by central differencing, we take $\texttt{offset} = 2\sqrt{\Delta x^2 + \Delta y^2}$.

[4] The tangential component of a velocity field changes a curve's parameterization (if any), not its location.

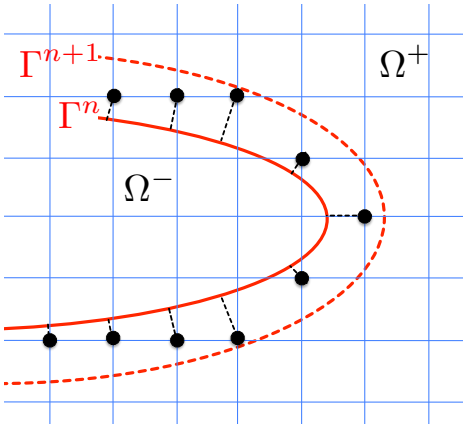


Figure 10: Interface at time t^n (red solid line) and t^{n+1} (red dashed line). The black solid disks represent grid nodes that are swept over by the interface between the two consecutive time steps and where valid values of T^n are needed in order to evaluate the right-hand-side in equation (9). Those values are obtained by extrapolating T^n from $\phi^n \leq 0$ to $\phi^n > 0$ in the direction normal to the interface (black dotted lines). (Color online).

The value of Q across the interface is then found by solving the following three partial differential equations. First solve

$$\frac{\partial Q_{nn}}{\partial \tau} + H(\phi)\left(\nabla Q_{nn} - Q_{nnn}\right) = 0,$$

defining Q_{nn} in such a way that its normal derivative is equal to Q_{nnn}. Then solve

$$\frac{\partial Q_{n}}{\partial \tau} + H(\phi)\left(\nabla Q_{n} - Q_{nn}\right) = 0,$$

defining Q_{n} in such a way that its normal derivative is equal to Q_{nn}. Finally solve

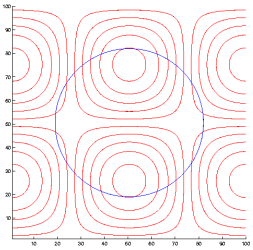
$$\frac{\partial Q}{\partial \tau} + H(\phi)\left(\nabla Q - Q_{n}\right) = 0,$$

defining Q in such a way that its normal derivative is equal to Q_n. These equations are solved using a fifth-order accurate WENO scheme [59, 60, 79] in space and a third-order accurate TVD scheme [130] in fictitious time τ. This step is computationally expensive and it is therefore important to localize this process as much as possible: We use a few iterations in fictitious time (typically 15), since one usually only seeks to extrapolate the values of Q in a narrow band of a few grid cells around the interface. The operations can also be performed in a small band near the interface to improve the efficiency of this step. In addition, one may use a 'local' approach to store and compute the desired quantities. We refer the interested reader to the work of Brun *et al.* [21], who have introduce a truly local level-set method using hash-table constructs. In particular, their approach allows for the storage of only a band of grid points around the free boundary, while accessing the data with a $O(1)$ complexity. Their method thus combine efficiency in CPU as well as in memory requirement for local level-set methods. We will also discuss an efficient approach based on Quadtree/Octree data structure in section 5. Figure 11 illustrates the constant, linear, quadratic and cubic extrapolation results obtained with this technique.

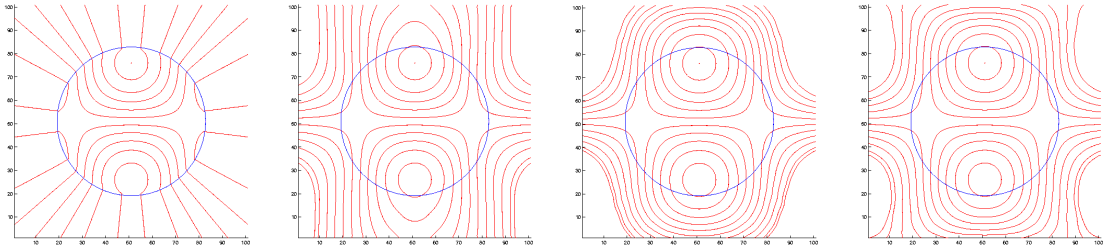
Remark: In the illustrative example above, we presented a third-order accurate extrapolation. We note that a third-degree extrapolation will be needed only in the case where an overall fourth-order solution is computed. Therefore, since Q is fourth-order accurate in that case, its third derivative is convergent.

4.3 Time Discretization

In [45], Gibou and Fedkiw pointed out that special care is needed when defining the interface's normal velocity and evolving the level-set equation in time. They considered the Frank-Sphere solution in one



(a) Exact extrapolation.



(b) Constant. (c) Linear. (d) Quadratic. (e) Cubic.

Figure 11: Extrapolation results using the methodology of Aslam [8] with different degree of extrapolations. The red lines represent the iso-contour of the solution, which is defined analytically inside the circular domain (blue line) and extrapolated outside. (Color online).

spatial dimension on a domain $\Omega = [-1, 1]$ with Dirichlet boundary conditions at the domain boundaries. The Frank sphere solution in one spatial dimension describes a slab of radius $R(t) = S_0\sqrt{t}$, for which the exact solution takes the form:

$$T = \begin{cases} 0 & s \leq S_0, \\ T_\infty \left(1 - \frac{F(s)}{F(S_0)}\right) & s > S_0, \end{cases}$$

where $s = |x|/\sqrt{t}$. In one spatial dimension $F(s) = \text{erfc}(s/2)$, with $\text{erfc}(z) = 2\int_z^\infty e^{-t^2} dt/\sqrt{\pi}$.

Choosing the initial time to be $t_{\text{initial}} = 1$ and $T_\infty = -.5$, the initial radius is defined through the definition of the normal velocity, $\mathbf{V_n} = -D\left[\nabla T\right]_{|\Gamma} \cdot \mathbf{n}$, as $S_0 \approx .86$. The initial interface is defined using $\phi = |x| - S_0$ and the solution is computed until $t_{\text{final}} = 1.5$. The Crank-Nicholson scheme in time is used with a time step restriction of $\triangle t \approx \triangle x^{3/2}$ to emulate a third-order accurate scheme in time[5]. Also, a cubic extrapolation is used to define the ghost values. Howevever, this method produces results that are only second-order accurate, as shown in figure 12(a).

This lower accuracy originates from the lack of consistency in the definition of $\mathbf{V}_\mathbf{n}^{n+1}$. For example to approximate the one-dimensional equation:

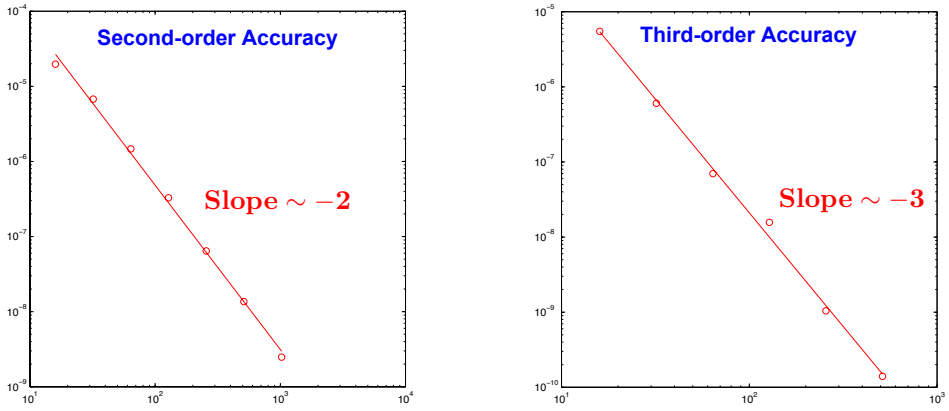
$$\frac{d\phi}{dt} = \mathbf{V_n}(\phi)|\nabla\phi|,$$

with the Crank-Nicholson scheme, evolving ϕ from time t^n to time t^{n+1}, the following three steps are performed:

1. Use $\mathbf{V}_\mathbf{n}^n(\phi^n)$ to evolve ϕ^n to ϕ_{temp}^{n+1} with an Euler step.
2. Use $\mathbf{V}_\mathbf{n}^{n+1}(\phi_{\text{temp}}^{n+1})$ to evolve ϕ_{temp}^{n+1} to ϕ^{n+2} with an Euler step.
3. Define $\phi^{n+1} = (\phi^n + \phi^{n+2})/2$.

[5]In practice a third-order accurate scheme in time should be chosen.

In the case of the Stefan problem, the normal velocity at time t^{n+1} needs to satisfy the relation $\mathbf{V_n^{n+1}} = \mathbf{V_n^{n+1}}(\phi^{n+1})$. Therefore the $\mathbf{V_n^{n+1}}$ from step 2 above needs to be consistent with the ϕ^{n+1} computed in step 3, which may not be the case. To solve this problem steps 2 and 3 are iterated until the normal velocity at time t^{n+1} satisfies the relation $\mathbf{V_n^{n+1}} = \mathbf{V_n^{n+1}}(\phi^{n+1}) = \mathbf{V_n}\left(\left(\phi^n + \phi^{n+2}\right)/2\right)$ to some tolerance. In practice, the tolerance is taken to be 10^{-8}, and typically 3 or 4 iterations are needed. Figure 12(b) demonstrates that such a time discretization produces a third-order accurate solution.



(a) Naive time evolution. (b) Corrected time evolution.

Figure 12: Error analysis in the L^∞-norm for the one-dimensional Frank sphere solution of section 4.3. The symbols represent the errors of the numerical solution on a log-log scale, and the solid lines depict the least square fit with slope -2.18 (a) and -3.02 (b).

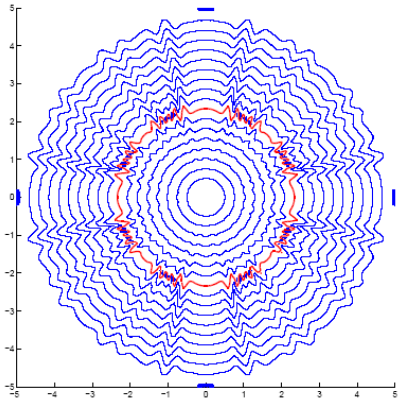
4.4 Level-Set Evolution and Reinitialization

The level-set advection equation (6) and the reinitialization equation (7) are discretized with a HJ-WENO scheme in space [59, 60, 79] and a TVD-RK3 in time [130]. Gibou and Fedkiw noted in [45] that the solution of the reinitialization equation is only second-order accurate in the L^∞-norm, despite the fact that the numerical approximations used are fifth-order accurate in space. Russo and Smereka also pointed out that the original interface location is not preserved during the course of solving the reinitialization equation. They then corrected this anomaly by imposing explicitly in the numerical method the correct initial location of the rarefaction wave solution [118]. Later, Du Chene *et al.* extended this method to fourth-order accuracy in the L^∞-norm and showed that curvature computations are second-order accurate in the L^∞-norm [30]. Figure 13 illustrates the difference in the computation of the interface's mean curvature between the traditional HJ-WENO scheme of [59] and the modified HJ-WENO scheme of [30]. Min and Gibou also used the idea of Russo and Smereka with slight modifications in the context of adaptive mesh refinement [90], and Min pointed out that it is advantageous in terms of speed and memory to replace the traditional Runge-Kutta scheme in time with a Gauss-Seidel iteration of the forward Euler scheme [88]. Finally, we mention that other techniques can be used to reinitialize ϕ as a distance function [125, 124, 149, 166, 148, 31, 147, 53], each with their pros and cons. We refer the interested readers to the book by Osher and Fedkiw [101] as well to the book by Sethian [127] for more details on the level-set method.

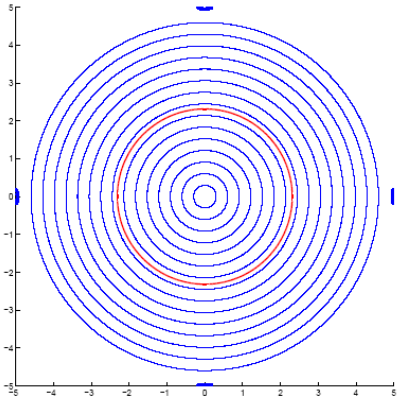
4.5 Accuracy of the Stefan Problem

Consider the Stefan problem in a domain $[-1,1]\times[-1,1]$ with Dirichlet boundary conditions at the domain's boundary. In two spatial dimensions, the Frank sphere solution describes a disk of radius $R(t) = S_0\sqrt{t}$ parameterized by S_0. The exact solution takes the form:

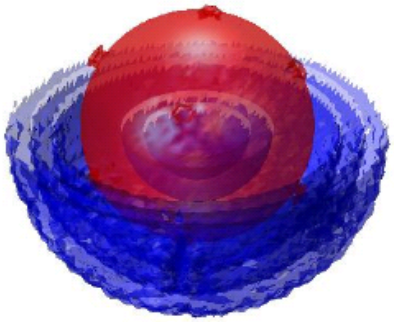
$$T = \begin{cases} 0 & s \le S_0, \\ T_\infty \left(1 - \frac{F(s)}{F(S_0)}\right) & s > S_0, \end{cases}$$



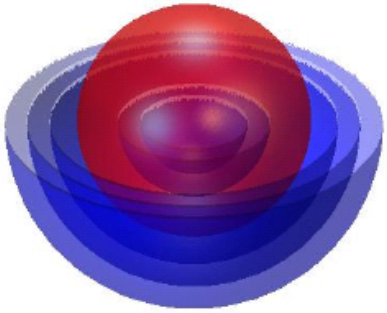
(a) Standard HJ-WENO in 2D [59].



(b) Modified HJ-WENO in 2D [30].



(c) Standard HJ-WENO in 3D [59].



(d) Modified HJ-WENO in 3D [30].

Figure 13: Comparison of the isocontour of the mean curvature for circular (2D) and spherical (3D) interfaces using the HJ-WENO scheme from Jiang and Peng [59] and the modified HJ-WENO scheme of Du Chéné *et al.* [30]. (Color online).

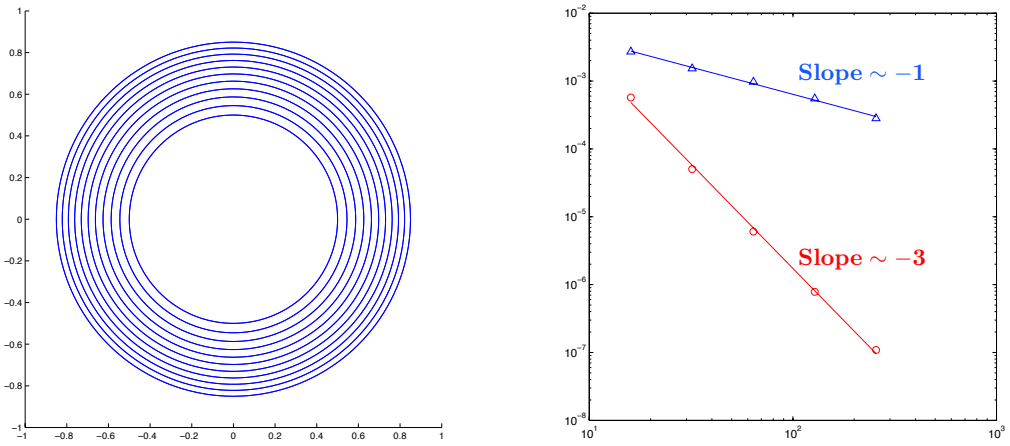
where $s = |x|/\sqrt{t}$, and with T_∞ and S_0 related by the jump condition $V_n = -D\left[\nabla T\right]_{|\Gamma} \cdot \mathbf{n}$. In two spatial dimensions $F(s) = \mathrm{E}_1(s^2/4)$, with $\mathrm{E}_1(z) = \int_z^\infty (e^{-t}/t)\, dt$. We choose the initial time to be $t_{\text{initial}} = 1$ and the initial radius to be $S_0 = .5$, hence defining $T_\infty \approx -.15$. Figure 14(a) depicts snapshots of the interface evolution and figure 14 (b) presents the accuracy results for the first-order accurate and the third-order accurate schemes of [47] and [45], respectively.

Qualitative Behavior: In [46], Gibou *et al.* described the effects of surface tension, anisotropy, diffusion parameters and compared non-trivial crystal growth to solvability theory. For the sake of presenting similar results once only, we will show the typical results in the case of adaptive grids in section 8, noting that the results on adaptive grids are identical to those on uniform grids, with an obvious gain in efficiency.

5 Adaptive Mesh Refinement - Node-Based Approach on Quadtrees/Octrees

5.1 Introduction

The problems considered so far were discretized on uniform grids. Elliptic and parabolic problems produce solutions that are smooth except near boundaries, where a combination of Dirichlet boundary conditions and diffusion coefficients may introduce jumps in the solution gradients (and sometimes the solution itself). We also showed in section 3.1.3 that the accuracy of the numerical solution may deteriorates near the irregular domain's boundary. In addition, in the case where the solution varies rapidly in narrow regions, it is very



(a) Evolving front from $t = 1s$ to $t = 2.89s$.

(b) Accuracy in the L^∞-norm.

Figure 14: Two-dimensional Frank sphere solution of section 4.5. (a) Interface evolution at different times and (b) accuracy in the L^∞-norm for the method of [47] and [45] versus the number of grid points in a log-log scale. The open symbols are the numerical errors and the solid lines are the least-square fits with slope -0.80 for the method of [47] and -3.07 for the method of [45], respectively.

desirable to refine the grid in that region only while keeping a coarser grid structure in the parts of the domain where the solution is known to be smooth. Finally, in the large majority of applications modeling diffusion dominated phenomena, the region where the solution varies rapidly is only located near the boundary of the irregular domain. For these reasons, it is desirable to design adaptive meshing strategies that enable the ability to refine the grid near the interface while coarsening the grid away from it.

Several strategies for solving partial differential equations on adaptive meshes have been introduced in the past several decades. Unstructured meshes used in the finite element method are extremely successful in structural mechanics where deformations are small. However, in the case of free boundary problems, the high cost of regularly reconstructing a boundary fitted mesh is computationally inefficient. Nevertheless, authors have successfully analyzed Stefan-type problems for simulating dendritic growth; see e.g [52, 167] and the references therein.

In the case of Cartesian grids, the first work to consider adaptive mesh refinement was that of Berger and Oliger [15]. In this work, a coarse uniform grid discretizes the computational domain and blocks of uniform grids are then recursively added as needed. Numerical methods for a large class of partial differential equations have been introduced using this framework; see e.g. [14, 138, 85] and the references therein. More recently, quadtree and octree data structures have been preferred [3], since they allow the grid to be continuously refined without being bound by blocks of uniform grids. In the case where the equations considered are those of fluid dynamics, for which finite volume approaches are the state-of-the-art, a cell-centered approach is preferred. This is due mainly to the fact that the numerical approximations of the gradient and the divergence operators conserve their analytical 'minus transpose' property, which in turn guarantees stability properties. Several works have used this cell-centered approach for the simulation of fluids; see e.g. [110, 81, 80] and the references therein. We note that block structured AMR solvers, aided by efficient multigrid solvers (see [32] and the references therein), have advantages in that the entire grid structure may be stored efficiently, which may speed up the execution time. However, they do not have the flexibility of Octrees and require more grid points and therefore computational time. We also refer the interesting work of [22] that discusses high performance computing using octrees and the work of [144] on an efficient multigrid method on Octree grids.

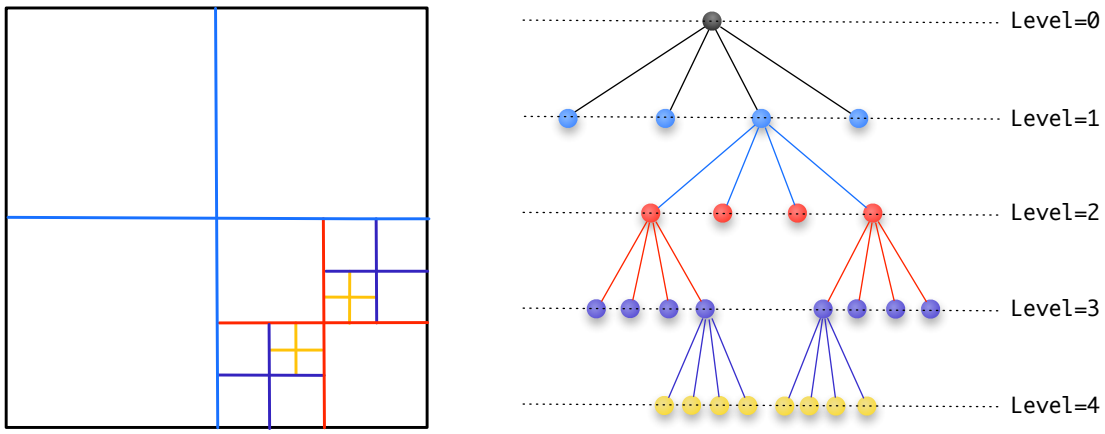


Figure 15: Discretization of a two-dimensional domain (left) and its quadtree representation (right). The entire domain corresponds to the root of the tree (level 0). Each cell can then be recursively subdivided further into four children. In this example, the tree is non-graded, since the difference of level between some adjacent cells exceeds one. (Color online).

Finite difference approaches do not have the 'minus transpose' properties on adaptive meshes, and special projection schemes must be used to ensure numerical stability [89]. However, in the case of elliptic and parabolic problems, finite difference schemes can be highly efficient. In particular, Min, Gibou and co-workers introduced a simple framework for discretizing standard operators on quadtree/octree [91, 89, 27]. In this framework, the data is sampled at the cells' vertices, and finite difference schemes can be developed to obtain second-order accurate solution in the L^∞-norm while considering arbitrary quadtree/octree grids. In addition, this approach has the advantage of producing second-order accurate gradients in the L^∞-norm. This property is especially beneficial in the case of diffusion-dominated phenomena like the Stefan problem, since the solution's *gradients* eventually determine the accuracy of the method (through the definition of the interface velocity (5)).

5.2 Spatial Discretization and Refinement Criterion

Quadtrees used in two spatial dimensions and octrees used in three spatial dimensions are standard data structures described in detail in Samet [121, 122]; herein, we present only the basics. Referring to figure 15, a single quadtree cell covers the entire two-dimensional domain and is associated to the root of the tree. Subsequently, cells are recursively split into four children until the desired size of the smallest cells is achieved. The process is identical in three spatial dimensions, except that cells are split into eight children. By definition, the level of the root cell is zero and is incremented by one for each new generation of children. Finally, a tree is said to be non-graded if the size difference between adjacent cells is not constrained; this impacts the ease of mesh generation and, to some extent, the computational efficiency [93, 155].

A meshing procedure that seeks to place the smallest cells near the boundary of the irregular domain and to coarsen the grid away from it is straightforward in cases where the domain is described implicitly. In [135], Strain proposed a criteria based on the Whitney decomposition. For a general function $\phi : \mathbb{R}^n \to \mathbb{R}$ with Lipschitz constant $\text{Lip}(\phi)$, the Whitney decomposition was later extended by Min in [87] to the following. Starting from the root cell, split any cell C for which:

$$\min_{v \in \text{vertices}(C)} |\phi(v)| \leq \text{Lip}(\phi) \cdot \text{diag-size}(C), \tag{11}$$

where diag-size(C) refers to the length of the diagonal of the current cell C and v refers to a vertex (node) of the current cell. In the case of a free boundary problem, the grid G^n at time t^n for which the smallest cells are on the interface Γ^n, must be adapted to a new grid G^{n+1} at time t^{n+1} to follow the evolution

of the interface, i.e. one must impose that the smallest cells are on Γ^{n+1}. Algorithm 2 gives the details of how adaptive grids are generated. In this algorithm, $\tilde{\phi}^{n+1} : \mathbb{R}^n \to \mathbb{R}$ represents the level-set function ϕ^{n+1} that has been reinitialized as a signed distance function. This process is simple and extremely efficient computationally since grid cells far away from the interface are few, resembling a local level-set approach. Note also that the solution of the reinitialization equation does not require that the pseudo time step τ used in equation (7) be taken uniformly for all cells, since only the steady-state solution matters. In turn, the time step taken for cells far away from the interface is large and compensate for the larger distance the rarefaction solution to the Eikonal equation needs to propagate to. In fact, [21] showed that the level-set method on Quadtree grids of [90] is on a par with a truly local level-set approach using hash-table structures.

Remarks:

- In the case where the refinement is performed near the interface in a quadtree/octree framework, the number of grid points is proportional to the surface of the irregular domain rather than its volume. Since, for elliptic problems, the main factor determining the execution time and memory consumption is the size of the resulting linear system, these discretizations are highly efficient.
- In the case where ϕ is a signed distance function, $\mathrm{Lip}(\phi) = 1$. In practice, $\mathrm{Lip}(\phi)$ in equation (11) plays the role of a parameter controlling the degree of "gradedness" of the grid. We have taken $\mathrm{Lip}(\phi) \approx 1.1$ to generate grids that are close to being graded and $\mathrm{Lip}(\phi) \approx 1/2$ to generate highly non-graded grids.

Algorithm 2 : Grid Generation

Input : G^n and $\tilde{\phi}^{n+1} : \mathbb{R}^d \to \mathbb{R}$

```
1.  G^{n+1} = G^n
2.  C = the root cell of G^{n+1}
3.  if the Lipschitz condition for φ̃^{n+1} is satisfied at C
4.      if C is a leaf cell
5.          split C
6.      end if
7.      for each child cell C' of C
8.          go to 3 with C = C'
9.      end for
10. else
11.     merge C
12. end if
```

Output : G^{n+1}

5.3 Finite Difference Discretizations

In the case of nonregular Cartesian grids, the main difficulty is to derive discretizations at T-junction nodes, i.e. nodes for which there is a missing neighboring node in one of the Cartesian directions. For example, figure 16(b) depicts a T-junction node, v_0, with three neighboring nodes v_1, v_2 and v_3 aligned in three Cartesian directions and one ghost neighboring node, v_g, replacing the missing grid node in the remaining Cartesian direction. The value of the node-sampled function $u : \{v_i\} \to \mathbb{R}$ at the ghost node v_g could, for example, be defined by linear interpolation:

$$u_g^G = \frac{u_3 s_4 + u_4 s_3}{s_3 + s_4}. \tag{12}$$

However, instead of using this second-order accurate interpolation, one can instead use the following third-order accurate interpolation. First, note that a simple Taylor expansion demonstrates that the interpolation error in equation (12) is given by:

$$u_g^G = \frac{u_3 s_4 + u_4 s_3}{s_3 + s_4} = u(v_g) + \frac{s_3 s_4}{2} u_{yy}(v_0) + O(\triangle x_{\mathrm{s}})^3, \tag{13}$$

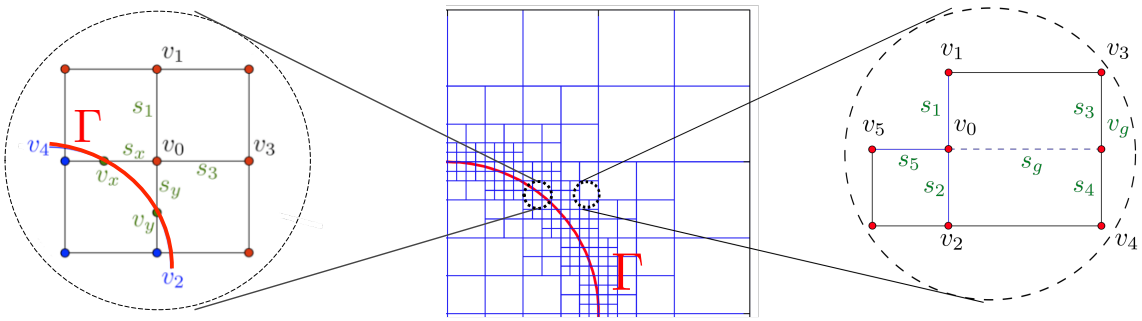
where $\triangle x_s$ is the size of the smallest grid cell with vertex v_0. The term $u_{yy}(v_0)$ can be approximated using the standard central differencing discretization:

$$\frac{2}{s_1+s_2}\left(\frac{u_1-u_0}{s_1}+\frac{u_2-u_0}{s_2}\right),$$

and used in equation (13) to define a third-order interpolation for u_g^G:

$$u_g^G = \frac{u_3 s_4 + u_4 s_3}{s_3+s_4} - \frac{s_3 s_4}{s_1+s_2}\left(\frac{u_1-u_0}{s_1}+\frac{u_2-u_0}{s_2}\right). \tag{14}$$

Similar techniques can be used to define third-order accurate ghost values in three spatial dimensions; we refer the interested reader to [90] for the formulas. We also point out that such definitions of ghost values only use the node values of the cells adjacent to v_0, which is beneficial since, accessing cells not immediately adjacent to the current cell is more difficult and could increase CPU and/or memory requirements.



(a) Locally uniform grid near Γ.

(b) T-junction.

Figure 16: Local grid configuration near a node v_0. The schematic on the right describes a T-junction where a node is missing in the x-direction. In contrast, the grid near the interface Γ is locally uniform (left).

The third-order interpolations defined above allow us to treat T-junction nodes in a same fashion as a regular node, up to third-order accuracy. Here, we refer to a regular node as a node for which all the neighboring nodes in the Cartesian directions exist. Therefore, we can define finite difference formulas for the first- and second-order derivatives at every node using standard formulas in a dimension-by-dimension framework. For example, referring to figure 17, we use the central difference formulas for u_x and u_{xx}:

$$D_x^0 u_0 = \frac{u_2-u_0}{s_2}\cdot\frac{s_1}{s_1+s_2}+\frac{u_0-u_1}{s_1}\cdot\frac{s_2}{s_1+s_2}, \tag{15}$$

$$D_{xx}^0 u_0 = \frac{u_2-u_0}{s_2}\cdot\frac{2}{s_1+s_2}-\frac{u_0-u_1}{s_1}\cdot\frac{2}{s_1+s_2}, \tag{16}$$

the forward and backward first-order accurate approximations of the first-order derivatives:

$$\begin{aligned} D_x^+ u_0 &= \frac{u_2-u_0}{s_2}, \\ D_x^- u_0 &= \frac{u_0-u_1}{s_1}, \end{aligned} \tag{17}$$

and the second-order accurate approximations of the first-order derivatives:

$$\begin{aligned} D_x^+ u_0 &= \frac{u_2-u_0}{s_2} - \frac{s_2}{2}\text{minmod}\left(D_{xx}^0 u_0, D_{xx}^0 u_2\right), \\ D_x^- u_0 &= \frac{u_0-u_1}{s_1} + \frac{s_1}{2}\text{minmod}\left(D_{xx}^0 u_0, D_{xx}^0 u_1\right), \end{aligned} \tag{18}$$

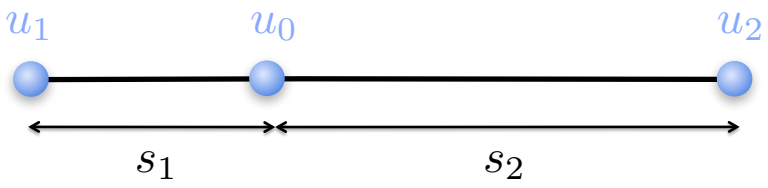


Figure 17: A one-dimensional adaptive grid.

where the minmod slope limiter [130, 79], defined as:

$$\text{minmod}(x,y) = \begin{cases} x & \text{if } |x| > |y|, \\ y & \text{otherwise,} \end{cases}$$

is used to avoid differencing across regions where gradients are large (i.e. near kinks). Similarly, approximations for first-order and second-order derivatives are obtained in the y- and z- directions.

5.4 Interpolation Procedures

Interpolation procedures are necessary to define data anywhere in a cell, for example, in order to use semi-Lagrangian methods (see section 7.1) or to interpolate a velocity field defined on uniform grids onto an adaptive level-set. In [136], Strain pointed out that piecewise bilinear (resp. trilinear) interpolations are natural choices in quadtree (resp. octree) data structures, since they involve data sampled at the cell's vertices only. However, these interpolations lead to low-order accurate schemes and induce a large amount of numerical dissipation, which in turn leads to a loss of mass in the level-set evolution.

In [90], Min and Gibou proposed the following quadratic interpolation scheme that automatically avoids nearby discontinuities in their constructions. Considering a cell C with dimensions $[0,1]^2$, the interpolated value of a scalar function u at (x,y) is:

$$\begin{aligned} u(x,y) = {} & u(0,0)(1-x)(1-y) \\ & + u(0,1)(1-x)(\quad y) \\ & + u(1,0)(\quad x)(1-y) \\ & + u(1,1)(\quad x)(\quad y) \end{aligned} - u_{xx}\frac{x(1-x)}{2} - u_{yy}\frac{y(1-y)}{2}, \tag{19}$$

where the second-order derivatives u_{xx} and u_{yy} are defined as:

$$u_{xx} = \text{minmod}_{v\in\text{vertices}(C)}(D^0_{xx}u(v)) \qquad \text{and} \qquad u_{yy} = \text{minmod}_{v\in\text{vertices}(C)}(D^0_{yy}u(v)).$$

5.5 Computing Second-Order Accurate Gradients

Calculating gradients with accuracy can be of significant importance for applications in which the flux at the interface defines the interface's velocity, for example, in the case of the Stefan problem. When this is the case, it is a strong advantage for a numerical method to produce second-order accurate gradients, which is a distinguishing feature of the method of Chen *et al.* [27]. In two spatial dimension, the components of the gradient are computed as:

$$u_x = \frac{u_g - u_0}{s_g}\cdot\frac{s_5}{s_g+s_5} + \frac{u_0-u_5}{s_5}\cdot\frac{s_g}{s_g+s_5} - \frac{s_3s_4s_5}{2s_g(s_5+s_g)}\left(\frac{u_1-u_0}{s_1}+\frac{u_2-u_0}{s_2}\right)\cdot\frac{2}{s_2+s_1}, \tag{20}$$

$$u_y = \frac{u_1-u_0}{s_1}\cdot\frac{s_2}{s_2+s_1} + \frac{u_0-u_2}{s_2}\cdot\frac{s_1}{s_2+s_1}. \tag{21}$$

For nodes next to the interface, interface nodes (v_x and v_y in figure 16(a)) are used in equations (20) and (21) instead of neighboring nodes that are outside the domain. Similar equations are derived in the three-dimensional case, and we refer the interested reader to [26] for the exact formulas.

5.6 Treatment of Dirichlet Boundary Conditions on Irregular Domains

In the case of adaptive grids, it is not straightforward to obtain supra-convergence if the interface cuts the grid in a T-junction cell. Fortunately, as mentioned in section 5.1, a great many problems require that the finest mesh be located around the domain's boundary. As a consequence, one can require that the smallest cells be located near the interface and, at an insignificant computational cost, that a narrow band of uniform cells be located near the irregular domain's boundary. This refinement strategy allows one to readily apply the techniques presented in section 3 to impose Dirichlet boundary conditions at irregular domains, because the grid is locally uniform, as illustrated in figure 16(a).

6 Solving the Poisson and the Diffusion Equations on Adaptive Grids

The discretization of the Poisson and the diffusion equations on adaptive grids follows the strategy outlined in the case of uniform grids. A Crank-Nicholson scheme is used to discretize the time derivative in the case of the diffusion equation, and central difference formulas (16) are used to approximate the spatial derivatives. This leads to a linear system that can be inverted to obtained the desired solution. At nodes neighboring the interface, Dirichlet boundary conditions are imposed as described in section 5.6. As noted in [27], the linear system is non-symmetric, but still leads to an M-matrix, so there exists a unique solution that can be computed with fast iterative solvers [119]. We also note that multigrid methods have been developed that are significantly more efficient developed on quadtree/octree grids, see e.g. [123, 144]. In sections 6.1 and 6.2, we give an example of the typical results for the Poisson and the heat equations that are obtained with this approach. The grid is represented by its minimum and maximum resolution, which we refer to as (`MinRes`, `MaxRes`).

6.1 Typical Results for the Poisson Equation

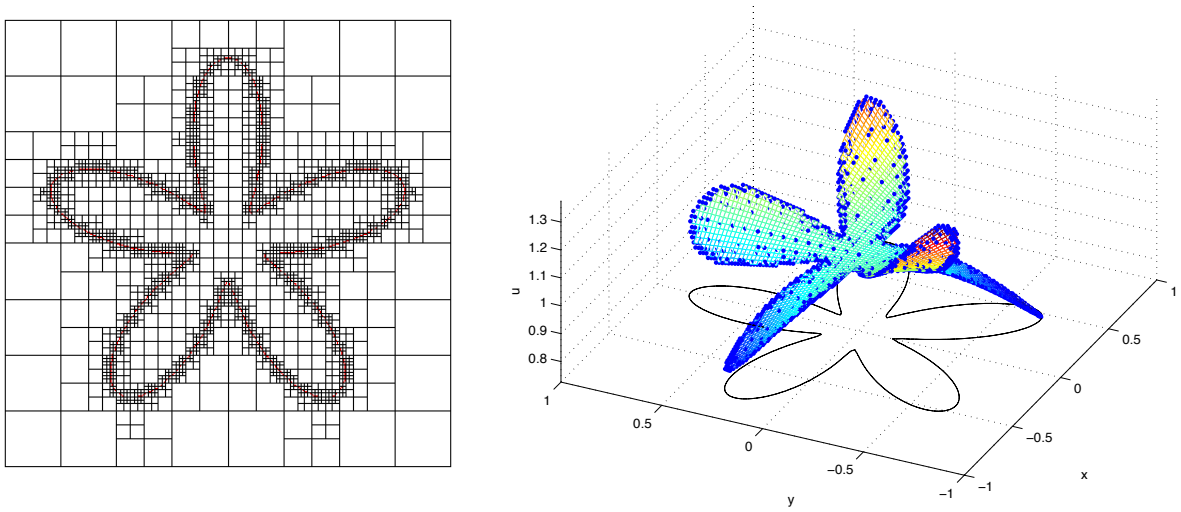
Consider the Poisson equation $\nabla \cdot (\beta \nabla u) = f$ on $\Omega = [-1, 1] \times [-1, 1]$ with an exact solution of $u = e^{xy}$, where $\beta = x^2 + y^2$. The interface is star-shaped, given by the set of points where $\phi = r - 0.5 - \frac{y^5 + 5x^4 y - 10x^2 y^3}{3r^5} = 0$, and $r = \sqrt{x^2 + y^2}$. A non-graded Cartesian grid with (`MinRes`, `MaxRes`) $= (8, 128)$, as well as the interface, is illustrated in figure 18(a). The numerical solution on this grid is plotted in figure 18(b). The numerical accuracy for the solution and its gradients are given in table 8 and table 9, respectively, demonstrating second-order accuracy in the L^∞-norm for both the solution and its gradients.

(`MinRes`, `MaxRes`)	L^∞ error	order	L^1 error	order
$(8, 128)$	5.897×10^{-4}	—	6.999×10^{-5}	—
$(16, 256)$	1.466×10^{-4}	2.008	1.600×10^{-5}	2.129
$(32, 512)$	3.468×10^{-5}	2.080	3.837×10^{-6}	2.060
$(64, 1024)$	8.278×10^{-6}	2.067	9.393×10^{-7}	2.030

Table 8: Accuracy results for the solution, u, in example 6.1.

(`MinRes`, `MaxRes`)	L^∞ error	order	L^1 error	order
$(8, 128)$	1.683×10^{-2}	—	2.500×10^{-3}	—
$(16, 256)$	4.237×10^{-3}	1.990	6.394×10^{-4}	1.967
$(32, 512)$	1.029×10^{-3}	2.041	1.613×10^{-4}	1.987
$(64, 1024)$	3.356×10^{-4}	1.617	4.054×10^{-5}	1.992

Table 9: Accuracy results for the solution's gradients, ∇u, in example 6.1.



(a) Quadtree Adaptive Mesh.

(b) Numerical (dots) and Exact Solutions (mesh plot).

Figure 18: Results for example 6.1.

6.2 Typical Results for the Diffusion Equation

Consider $u_t = \nabla \cdot (\beta \nabla u)$ on $\Omega = [-2, 2] \times [-2, 2]$ with an exact solution of $u = e^{-2\pi^2 \beta t} \cos(\pi x) \sin(\pi y)$, where $\beta = 0.2$. The interface is described by the level-set function $\phi = 16y^4 - x^4 - 32y^2 + 9x^2$. The numerical solution at $t = 0.25$ on a grid with $(\texttt{MinRes}, \texttt{MaxRes}) = (8, 64)$ is plotted in figure 19(b), while figure 19(a) depicts the grid used. The numerical accuracy for the solution and its gradients are given in table 10 and table 11, respectively. As it is the case for the Poisson equation, both the solution and its gradients are second-order accurate in the L^∞-norm.

$(\texttt{MinRes}, \texttt{MaxRes})$	L^∞ error	order	L^1 error	order
$(8, 64)$	1.741×10^{-2}	—	3.872×10^{-3}	—
$(16, 128)$	4.111×10^{-3}	2.083	8.922×10^{-4}	2.118
$(32, 256)$	1.011×10^{-3}	2.024	2.158×10^{-4}	2.048
$(64, 512)$	2.519×10^{-4}	2.005	5.304×10^{-5}	2.024

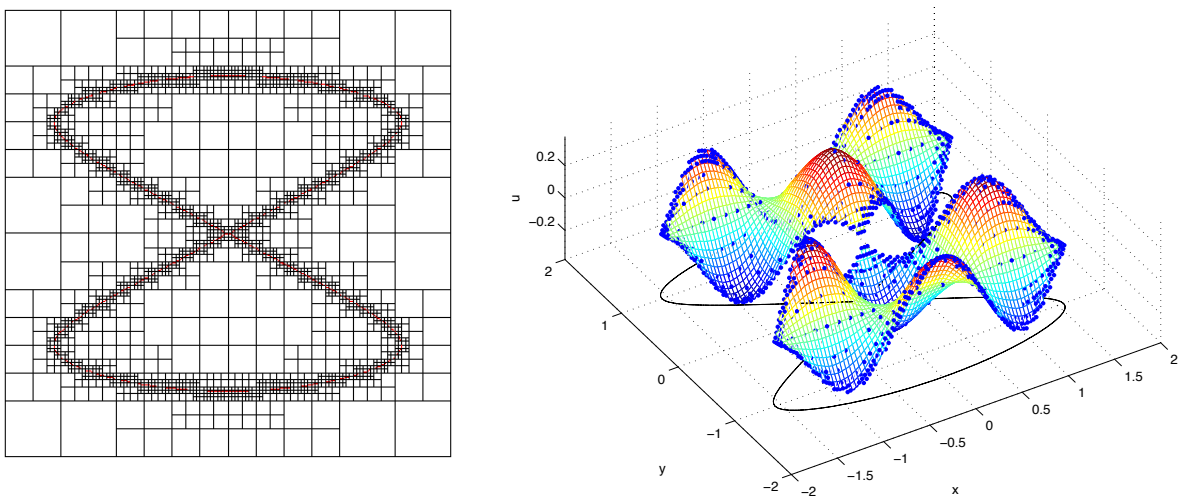
Table 10: Accuracy results for the solution u in example 6.2.

$(\texttt{MinRes}, \texttt{MaxRes})$	L^∞ error	order	L^1 error	order
$(8, 64)$	1.155×10^{-1}	—	4.272×10^{-2}	—
$(16, 128)$	3.102×10^{-2}	1.896	1.073×10^{-2}	1.994
$(32, 256)$	8.436×10^{-3}	1.878	2.671×10^{-3}	2.006
$(64, 512)$	2.283×10^{-3}	1.886	6.670×10^{-4}	2.002

Table 11: Accuracy results for the solution's gradients ∇u in example 6.2.

7 The Level-Set Technology on Adaptive Grids

In the case of the Stefan problem, as in any free boundary problems, it is necessary to capture the interface motion. To do so, we use the level-set method. On quadtree/octree grids, it is straightforward to discretize the equations related to the level-set method using the discretizations of the first- and second-order derivatives



(a) Quadtree Adaptive Mesh.

(b) Numerical (dots) and Exact Solutions (mesh plot).

Figure 19: Results for example 6.2.

presented in section 5.3. For example, the geometrical quantities, namely the normals to the interface and the interface curvatures, can be easily discretized using the central differencing formulas, equations (15) and (16). The discretizations of the main level-set equations, i.e. equations (6) and (7), are given next.

7.1 Discretization of the Level-Set Equation

If the velocity field is externally generated[6], as for the Stefan problem, the level-set equation (6) is linear and semi-Lagrangian schemes can be used. The advantage of these schemes is that they are unconditionally stable and thus avoid the standard CFL condition of $\Delta t \approx \Delta x_s$, where Δx_s is the size of the smallest cell in the computational domain.

In [90], Min and Gibou solved the level-set equation using a second-order accurate semi-Lagrangian scheme. Semi-Lagrangian methods are based on the fact that solutions to hyperbolic problems are constant along characteristic curves; therefore, for any grid point $\mathbf{x}^{n+1}$, $\phi^{n+1}(\mathbf{x}^{n+1}) = \phi^n(\mathbf{x_d})$, where $\mathbf{x_d}$ is the *departure* point from which the characteristic curve carries the information to $\mathbf{x}^{n+1}$. Min and Gibou used the second-order accurate mid-point method for locating this departure point, as in [159]:

$$\begin{aligned} \hat{\mathbf{x}} &= \mathbf{x}^{n+1} - \frac{\Delta t}{2} \cdot \mathbf{V}^n(\mathbf{x}^{n+1}), \\ \mathbf{x_d} &= \mathbf{x}^{n+1} - \Delta t \cdot \mathbf{V}^{n+\frac{1}{2}}(\hat{\mathbf{x}}). \end{aligned}$$

The velocity field $\mathbf{V}^{n+\frac{1}{2}}$ at the mid-time step, $t^{n+\frac{1}{2}}$, is defined linearly from the previous velocity fields as $\mathbf{V}^{n+\frac{1}{2}} = \frac{3}{2}\mathbf{V}^n - \frac{1}{2}\mathbf{V}^{n-1}$. Finally, quantities at the locations $\mathbf{x_d}$ and $\hat{\mathbf{x}}$ are approximated using the non-oscillatory interpolation procedure given in equation (19).

7.2 Discretization of the Reinitialization Equation

In the case of the reinitialization equation (7), the Hamiltonian is a function of ϕ rendering the equation nonlinear in ϕ. In this case, equation (7) cannot be solved with semi-Lagrangian schemes; rather, we use a Godunov scheme to capture nonlinear phenomena. Specifically, the semi-discrete discretization is written as:

$$\frac{\mathrm{d}\,\phi}{\mathrm{d}\,\tau} + \mathrm{sgn}(\phi^0)[H_G(D_x^+\phi, D_x^-\phi, D_y^+\phi, D_y^-\phi) - 1] = 0, \tag{22}$$

[6]The definition of the velocity does not depend on ϕ.

where H_G is the Godunov Hamiltonian, defined as:

$$H_G(a,b,c,d) = \begin{cases} \sqrt{\max(|a^+|^2,|b^-|^2)+\max(|c^+|^2,|d^-|^2)} & \text{if } \operatorname{sgn}(\phi^0) \le 0, \\ \sqrt{\max(|a^-|^2,|b^+|^2)+\max(|c^-|^2,|d^+|^2)} & \text{if } \operatorname{sgn}(\phi^0) > 0, \end{cases}$$

with $a^+ = \max(a,0)$ and $a^- = \min(a,0)$. It is therefore sufficient to approximate the one-sided derivatives $D_x^{\pm}\phi$ and $D_y^{\pm}\phi$. On the node-based quadtree/octree framework, these are approximated using second-order accurate, one-sided finite difference formulas of equation (18). The semi-discrete equation (22) is discretized in time with the Total Variation Diminishing second-order Runge-Kutta (TVD-RK2) scheme of Shu and Osher [130]. I.e. define $\tilde{\phi}^{n+1}$ and $\tilde{\phi}^{n+2}$ with two consecutive Euler's steps:

$$\frac{\tilde{\phi}^{n+1}-\phi^n}{\Delta\tau} + \operatorname{sgn}(\phi^0)[H_G(D_x^+\phi^n, D_x^-\phi^n, D_y^+\phi^n, D_y^-\phi^n) - 1] = 0,$$
$$\frac{\tilde{\phi}^{n+2}-\tilde{\phi}^{n+1}}{\Delta\tau} + \operatorname{sgn}(\phi^0)[H_G(D_x^+\tilde{\phi}^{n+1}, D_x^-\tilde{\phi}^{n+1}, D_y^+\tilde{\phi}^{n+1}, D_y^-\tilde{\phi}^{n+1}) - 1] = 0,$$

and then define ϕ^{n+1} by averaging: $\phi^{n+1} = (\phi^n + \tilde{\phi}^{n+2})/2$.

Remark: As mentioned in section 4.4, the reinitialization must transform an arbitrary level-set function into a signed distance function while preserving the original interface's location. In the case of adaptive grids, this is enforced following the idea of Russo and Smereka [118] and its modifications from Min and Gibou [90].

7.3 Improvement on Mass Conservation

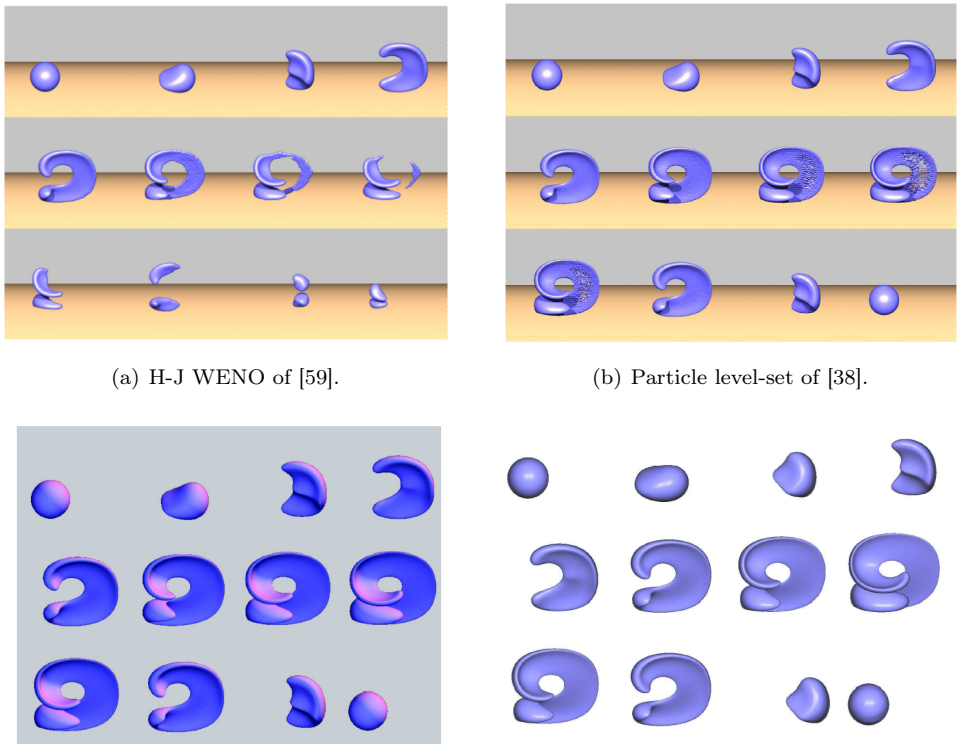
A well-known criticism of the level-set method is its inherent loss of mass. The source of the loss of mass is the lack of accuracy and the numerical dissipation of various approximations in solving equation (6). Successful approaches to combat the loss of mass involved hybridizing the level-set method with other methods that are known to be more accurate in terms of mass conservation [38, 139]. For example, figure 20(a) depicts the evolution of the level-set using the fifth-order HJ-WENO of [59], while figure 20(b) depicts the same evolution using the particle-level-set of [38]. In this example, the so-called Enright's test, the level-set is deformed according to the incompressible velocity field introduced in [74], before being rewound back to its initial position. Specifically, the velocity field $\mathbf{U} = (u,v,w)$ is given by:

$$\begin{aligned} u &= 2\sin^2(\pi x)\sin(2\pi y)\sin(2\pi z), \\ v &= -\sin(2\pi x)\sin^2(\pi y)\sin(2\pi z), \\ w &= -\sin(2\pi x)\sin(2\pi y)\sin^2(\pi z). \end{aligned}$$

The loss of mass is apparent in the case of the HJ-WENO in figure 20(a), where the shape of the initial sphere is not recovered at the end of the computation. This is in contrast with the particle-level-set of [38]. However, even in the case of [38], the lack of resolution prevents fine developing features (e.g. thin sheets) to be captured. A more refined grid will capture those features, but at a computational cost too high for practical applications. This is a case where adaptivity is a powerful technique since it allows fine resolution without the high computational footprint. Figure 20(c-d) gives the results of the evolution of the level-set with the adaptive particle-level-set approach of Losasso *et al.* [81] and the second-order accurate adaptive level-set of Min and Gibou [90]. In those cases, the fine octree grids enable the resolution of thin sheets.

Note that the quadtree/octree adaptive framework is efficient at addressing the loss of mass since the high-resolution is only focused near the interface so that the complexity of the level-set equations scales with the area of a surface in three spatial dimensions instead of with its volume, which is the case of uniform grids. It is easier to illustrate this point in two spatial dimensions on a similar example as the one above. Figure 21 illustrates the evolution of the interface location initially (left), at $t = 3\pi$ (center) and when the interface is fully rewound (right). It also depicts the quadtree grid being adapted. At the end of the computation, the mass loss is about 0.3%.

We also note that local level-set methods can also address this problem, although it was shown in [21] that a local level-set based on hashtables are only on a par with the quadtree/octree node-based approach



(a) H-J WENO of [59].

(b) Particle level-set of [38].

(c) Adaptive particle level-set of [81].

(d) Adaptive level-set (no particle) of [90].

Figure 20: Comparison of the qualitative evolution of the level-set function for the flow field introduced in [74]. On uniform grids (here 100^3), even the high-order accurate WENO scheme leads to a significant volume loss (a). Adding particles significantly improves the accuracy but small features (here thin sheets) cannot be resolved with particles alone (b). In the case of the particle level-set, the number of particles used is typically 16 per cell in 2D and 32 per cell in 3D. In contrast, the high-resolution capabilities of adaptive grids enable small features to be preserved at low computational cost [(c) and (d)]. Here the smallest grid size corresponds to an effective resolution of 512^3. The work of [81] uses first-order accurate semi-Lagrangian for the evolution of the level-set function and a second-order accurate ODE solver for the advection of particles. The work of [90] uses a second-order accurate scheme for the evolution of the level-set and no particles. The level-set function is reinitialize at every time step in all cases.

of Min and Gibou [90] in terms of CPU and memory requirement. We also mention that other local level-set methods have been proposed, see e.g [105, 99, 72] and the references therein. Finally, we note that true local-level-set methods, i.e. methods that only encode local grids in memory, may not be as practical as quadtree/octree level-set methods in some applications, since a valid value of the level-set function is not known throughout the computational domain. For example, one cannot find the distance to the interface at locations outside the local band.

Remark: Other tracking schemes exist, either using completely different approaches or hybridizing existing schemes. These methods are highly efficient at conserving mass and tracking interfaces, each with their own pros and cons. In additions, adaptive framework have been introduced (see e.g. [138, 15, 3, 14, 143, 76, 51, 19, 137, 10, 92, 4, 111, 41, 25, 58, 128, 66] and the references therein).

8 Solving the Stefan Problem on Adaptive Grids

The Stefan problem and similar model equations are obvious choices for the node-based adaptive framework presented above, since the framework produces second-order accurate solutions in the L^∞-norm *and* second-

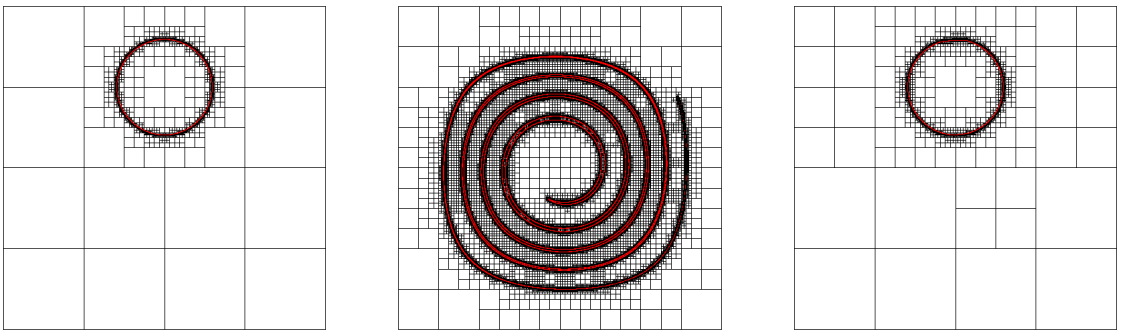


Figure 21: Level-set evolution with an effective resolution of 2048^2 at $t = 0$ (left), $t = 3\pi$ (center) and $t = 6\pi$ (right).

order accurate gradients; therefore it gives a second-order definition of the velocity field. It also enables efficient computations, since a three-dimensional simulation scales with the surface of the interface instead of its volume. Finally, non-graded grids can be readily considered, which leads to versatile grid generation. The procedure for solving the Stefan problem is given in algorithm 1. Here we use the technologies described in sections 5, 6 and 7. In the following sections, we present typical results that can be obtained with this framework.

8.1 Accuracy and Efficiency

We discuss the efficiency and accuracy on the known Frank-sphere exact solution of section 4.5. Figure 22 illustrates the evolution of the interface, as well as that of the adaptive grid, at different times using the method of Chen *et al.* [28]. The final time is $t = 10$, which demonstrates the robustness of the algorithm for large time computations. The computational domain is $\Omega = [-2, 2] \times [-2, 2]$ and the other parameters used in the computation are: $S_0 = 0.25$, $T_\infty = -0.05709187113307$ and (`MinRes`, `MaxRes`)=(8,64). The time step is $\triangle t = \triangle x_s$ where $\triangle x_s$ is the size of the finest cell.

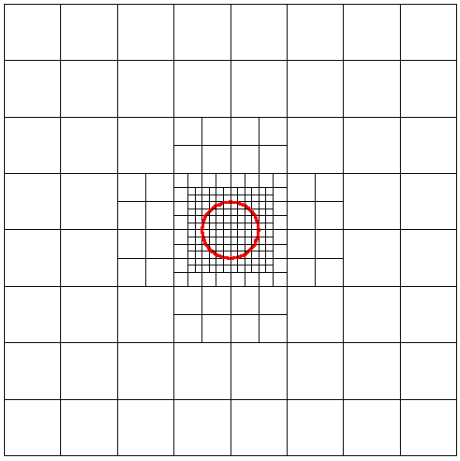
Tables 12–19 give the errors for the interface's location and the errors for the temperature field T in both the L^1- and the L^∞-norms for different combinations of (`MinRes`, `MaxRes`). The accuracy results given in these tables highlight the fact that the accuracy is largely driven by the resolution near the interface. In particular, a comparison of the errors in tables 12-13 with the errors in tables 18-19, indicates that the accuracy obtained on a uniform 256×256 grid is on a par with that obtained on a $(\texttt{MinRes}, \texttt{MaxRes}) = (32, 256)$ adaptive grid. This confirms the fact that the quadtree/octree adaptive mesh refinement approach is highly efficient for elliptic and parabolic problems in the case where the refinement criteria imposes the smallest cells on the interface Γ, while coarser and coarser cells are placed as the distance to the interface increases.

To demonstrate the saving of computational efforts through the use of adaptive grids, [28] computed the computational time on a 1.6 GHz laptop as a function of the maximum error in ϕ and T (see figure 23). In these plots, the degree of adaptivity is defined as `MaxRes`/`MinRes`. One can see that, for the same accuracy, the computational time on adaptive grids can be several orders of magnitude less than that on uniform grids.

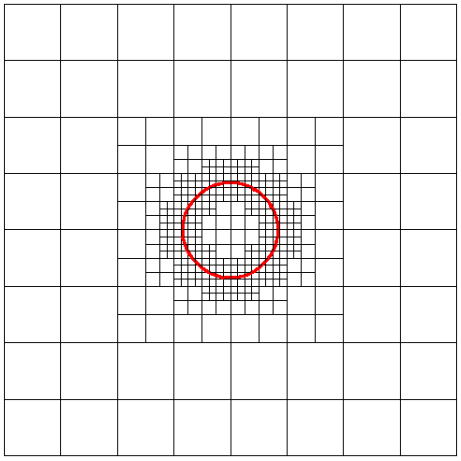
Remark: Although all the computations are carried out to second-order accuracy in the L^∞-norm, the resulting overall solution has a lower convergence rate (≈ 1.6). Chen *et al.* [28] attribute this loss of accuracy to the diverse approximations such as extrapolation and reinitialization procedures that are not iterated to steady-state. We also refer the reader to section 4.3 for a discussion on the time evolution.

8.2 Typical Numerical Results for Unstable Solidification

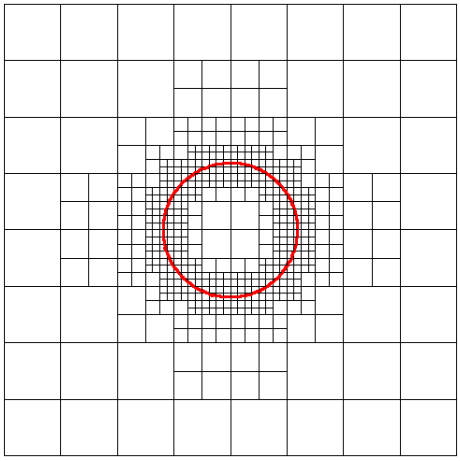
Unstable solidification from a seed in an undercooled liquid is typical of crystal growth. In what follows, we consider a temperature field initialized uniformly as the Stefan number $St < 0$ in the liquid phase, and



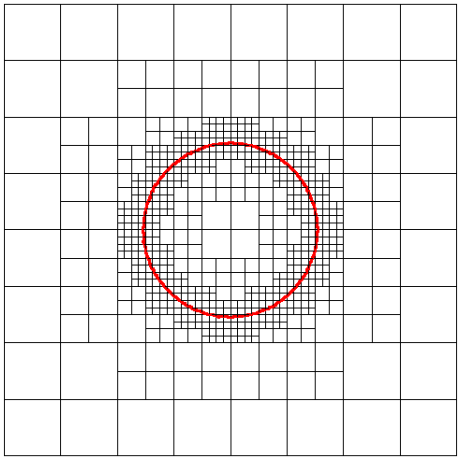
(a) t=1.



(b) t=3.



(c) t=6.



(d) t=10.

Figure 22: Evolution of the interface and corresponding adaptive grid for example 8.1.

Grid	L^∞ error	order	L^1 error	order
32×32	5.197×10^{-2}	—	3.253×10^{-2}	—
64×64	1.489×10^{-2}	1.804	1.100×10^{-2}	1.564
128×128	5.395×10^{-3}	1.464	3.535×10^{-3}	1.638
256×256	1.737×10^{-3}	1.635	1.100×10^{-3}	1.684

Table 12: Accuracy results on uniform grids for ϕ in example 8.1.

Grid	L^∞ error	order	L^1 error	order
32×32	3.016×10^{-3}	—	5.395×10^{-4}	—
64×64	1.094×10^{-3}	1.463	1.739×10^{-4}	1.633
128×128	4.476×10^{-4}	1.290	5.523×10^{-5}	1.655
256×256	1.498×10^{-4}	1.579	1.442×10^{-5}	1.938

Table 13: Accuracy results on uniform grids for T in example 8.1.

(`MinRes`, `MaxRes`)	L^∞ error	order	L^1 error	order
$(16, 32)$	5.503×10^{-2}	—	3.504×10^{-2}	—
$(32, 64)$	1.492×10^{-2}	1.883	1.102×10^{-2}	1.669
$(64, 128)$	5.410×10^{-3}	1.463	3.541×10^{-3}	1.638
$(128, 256)$	1.750×10^{-3}	1.629	1.112×10^{-3}	1.672

Table 14: Accuracy results on adaptive grids for ϕ in example 8.1 with `MaxRes`/`MinRes`=2.

(`MinRes`, `MaxRes`)	L^∞ error	order	L^1 error	order
$(16, 32)$	3.223×10^{-3}	—	8.818×10^{-4}	—
$(32, 64)$	1.100×10^{-3}	1.552	2.314×10^{-4}	1.930
$(64, 128)$	4.490×10^{-4}	1.292	7.122×10^{-5}	1.700
$(128, 256)$	1.509×10^{-4}	1.573	1.860×10^{-5}	1.937

Table 15: Accuracy results on adaptive grids for T in example 8.1 with `MaxRes`/`MinRes`=2.

(`MinRes`, `MaxRes`)	L^∞ error	order	L^1 error	order
$(8, 32)$	5.522×10^{-2}	—	3.519×10^{-2}	—
$(16, 64)$	1.494×10^{-2}	1.886	1.102×10^{-2}	1.676
$(32, 128)$	5.501×10^{-3}	1.441	3.661×10^{-3}	1.589
$(64, 256)$	1.801×10^{-3}	1.611	1.134×10^{-3}	1.691

Table 16: Accuracy results on adaptive grids for ϕ in example 8.1 with `MaxRes`/`MinRes`=4.

(`MinRes`, `MaxRes`)	L^∞ error	order	L^1 error	order
$(8, 32)$	3.231×10^{-3}	—	1.397×10^{-3}	—
$(16, 64)$	1.093×10^{-3}	1.564	3.756×10^{-4}	1.895
$(32, 128)$	4.567×10^{-4}	1.259	1.172×10^{-4}	1.680
$(64, 256)$	1.552×10^{-4}	1.557	3.059×10^{-5}	1.938

Table 17: Accuracy results on adaptive grids for T in example 8.1 with `MaxRes`/`MinRes`=4.

(`MinRes`, `MaxRes`)	L^∞ error	order	L^1 error	order
$(4, 32)$	5.521×10^{-2}	—	3.519×10^{-2}	—
$(8, 64)$	1.490×10^{-2}	1.890	1.098×10^{-2}	1.680
$(16, 128)$	5.559×10^{-3}	1.423	3.656×10^{-3}	1.586
$(32, 256)$	1.924×10^{-3}	1.530	1.209×10^{-3}	1.597

Table 18: Accuracy results on adaptive grids for ϕ in example 8.1 with `MaxRes`/`MinRes`=8.

(`MinRes`, `MaxRes`)	L^∞ error	order	L^1 error	order
$(4, 32)$	3.230×10^{-3}	—	1.484×10^{-3}	—
$(8, 64)$	1.095×10^{-3}	1.560	5.614×10^{-4}	1.403
$(16, 128)$	4.580×10^{-4}	1.258	1.661×10^{-4}	1.757
$(32, 256)$	1.659×10^{-4}	1.465	5.338×10^{-5}	1.638

Table 19: Accuracy results on adaptive grids for T in example 8.1 with `MaxRes`/`MinRes`=8.

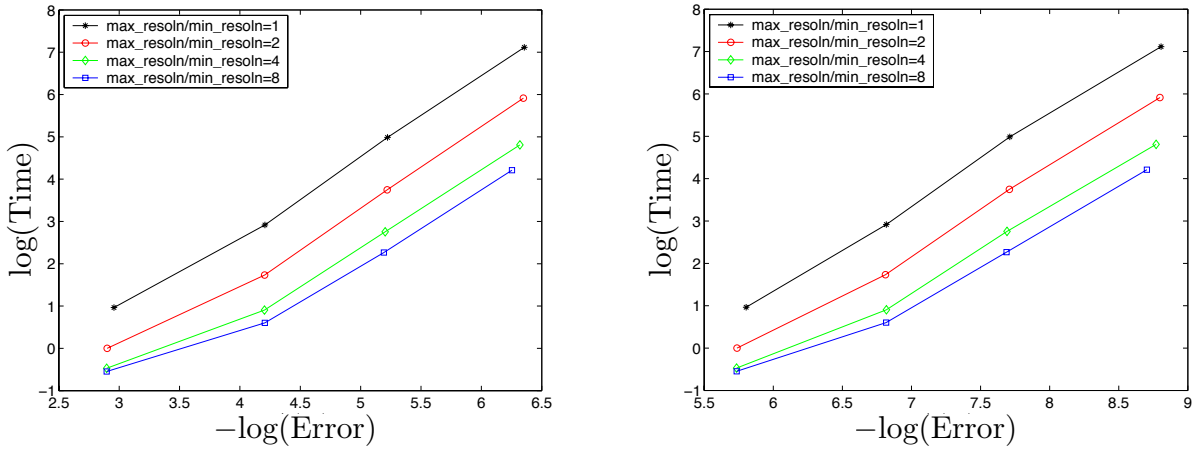
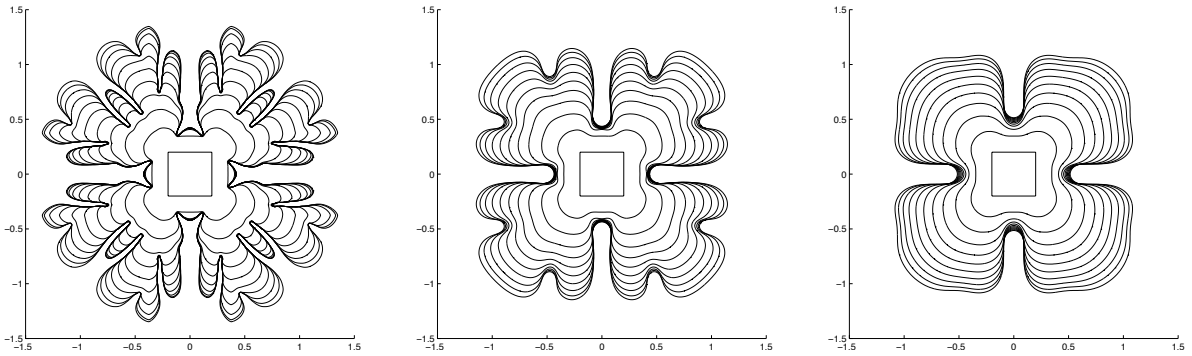


Figure 23: Log-log plot of the computational time as a function of the maximum error in ϕ (left) and the maximum error in T (right).

$T = 0$ in the solid region. Unless otherwise stated, the diffusion constant is the same in both phases and Neumann (adiabatic) boundary conditions are imposed on the four sides of the computational domain Ω.

8.2.1 Effect of Varying Isotropic Surface Tension

Surface tension forces are one of the main driving forces in solidification processes and are therefore important to simulate accurately. In the Stefan problem, surface tension is modeled through the $-\epsilon_c\kappa$ term in the Gibbs-Tompson boundary condition (4). Figure 24 depicts the growth history of a square solid seed. Instabilities naturally develop from the regions of high curvature (initial corners of the seed) and are (increasingly) damped by (increasing) surface tension forces (increasing ϵ_c). In this example, we consider isotropic surface tension, i.e we take $T = -\epsilon_c\kappa$ and vary the values of ϵ_c. The computational domain is $\Omega = [-1.5, 1.5] \times [-1.5, 1.5]$, the undercooled liquid has a Stefan number of $St = -0.5$ and the time step is $\triangle t = 0.004$.



(a) $\epsilon_c = 0$. (b) $\epsilon_c = 0.0005$. (c) $\epsilon_c = 0.001$.

Figure 24: Effect of varying the isotropic surface tension coefficient ϵ_c. The Gibbs-Thomson relation (4) with $\epsilon_v = 0$ and different ϵ_c values is imposed at the interface. The time levels shown are in uniform increments from $t = 0$ to $t = 0.4$.

8.2.2 Effect of Anisotropic Surface Tension

It is well known that crystals grow along preferred crystalline directions. In the Stefan problem, this is modeled by anisotropic surface tension forces, i.e. ϵ_c in equation (4) is a function of orientation. Figure 25 illustrates the evolution of an initially regular-pentagon-shaped seed placed in an undercooled liquid with the Stefan number $St = -0.5$ and a Gibbs-Thomson relation given by $T = -0.001(8/3\sin^4(2\alpha - \pi/2))\kappa$, where α is the angle between the normal to the interface and the x-axis. The boundary condition imposes a four-fold anisotropy, favoring the growth along the diagonal directions, while limiting it in the main Cartesian directions. For example, the initial instability triggered by the sharp corner in the positive y-direction is slowed down by the action of surface tension forces, promoting the subsequent side branching.

Figure 25 also illustrates the evolution of the interface on both a uniform 256×256 grid and adaptive moving grids with (`MinRes`, `MaxRes`)=(32,256). The results are almost identical, but the computation on adaptive grids is significantly more efficient in terms of memory and CPU. In fact, figure 26 depicts the grids used at the final time. The number of nodes in the case of the adaptive grid is only 23% of the uniform grid. This translates into a computational time of about 15% of that of a uniform 256×256 grid.

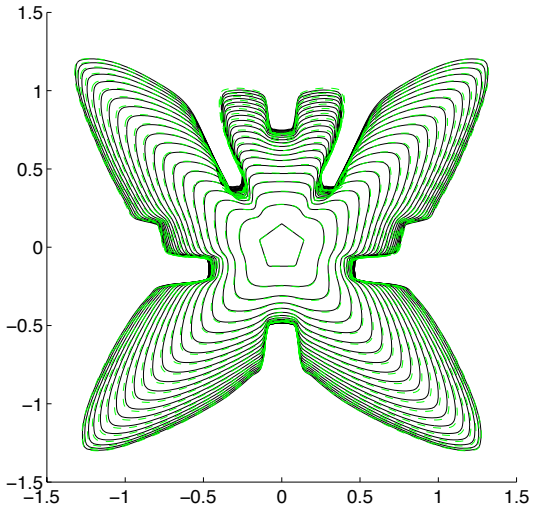
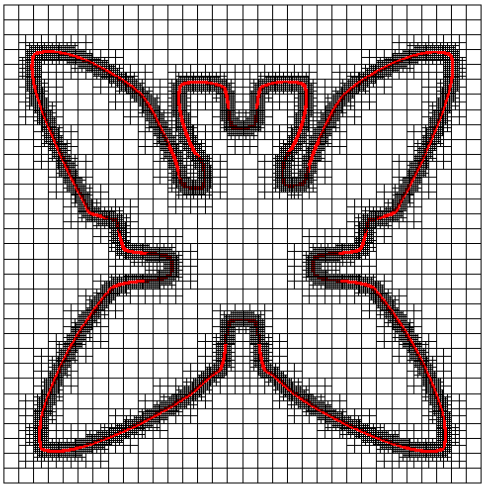


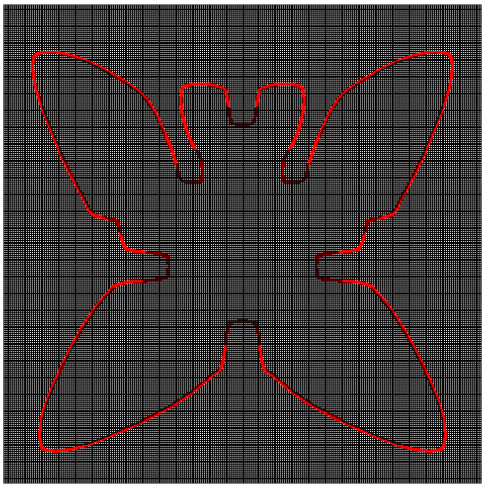
Figure 25: Effect of anisotropic surface tension. The Gibbs-Thomson relation $T = -0.001(8/3\sin^4(2\alpha - \pi/2))$ is imposed on the interface. The black solid lines depict the interface growth history on a uniform 256×256 grid, and the green dashed lines depict the interface growth history on adaptive moving grids. (Color online).

8.2.3 Comparison with the Microscopic Solvability Predictions

The Stefan problem is difficult to solve analytically. Simple one-dimensional analytical solutions can be easily derived, but analytical results in two and three spatial dimensions are rare. We considered the Frank-sphere exact solution in section 8.1. Here, we present another set-up for which solvability theory can predict the steady-state speed of the dendrite's tip. Consider a circular seed of radius 0.05 at the center of a $\Omega = [-6, 6]^2$ computational domain. The undercooled liquid has a Stefan number of $St = -0.45$ and the Gibbs-Thomson relation (4) on the solid-liquid is given by $T = -0.001[1 + 0.4(1 - \cos 4\alpha)]\kappa$, where α is still the angle between the normal to the interface and the x-axis. Figure 27(a) depicts the evolution of the interface from $t = 0$ to $t = 2.2$ on a (`MinRes`, `MaxRes`)=(64, 1024) moving grids, while figure 27(b) plots the tip velocity as a function of time. The tip non-dimensional velocity reaches a steady-state value of 1.7, in agreement with solvability theory [86].

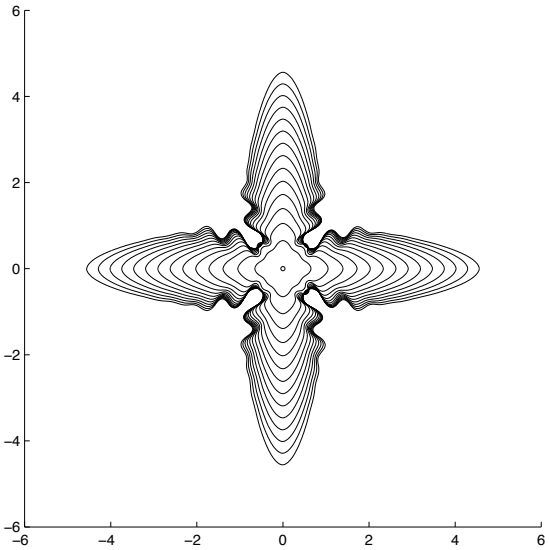


(a) Adaptive grid used in example 8.2.2 at $t = 0.4$.

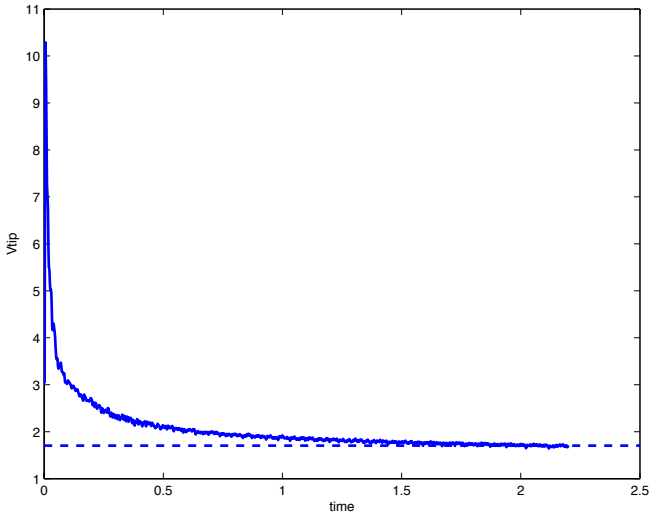


(b) Uniform grid used in example 8.2.2 at $t = 0.4$.

Figure 26: Comparison of the grids used in example 8.2.2.



(a) Growth history.



(b) Tip velocity

Figure 27: (a) Growth history of a circular seed growing under standard four-fold anisotropic surface tension. (b) Tip velocity as a function of time converging to a steady-state value of 1.7.

9 Conclusion

We have provided the essence of how to solve the Poisson and the diffusion equations on irregular domains as well as the Stefan problem using a level-set framework to keep track of the interface and the ghost-fluid method to apply Dirichlet boundary conditions. We have also presented the node-based quadtree/octree framework of Min and Gibou and its application to the model problems mentioned above. Finally, we have provided some implementation details for these algorithms and pointed out some common misconceptions and pitfalls in implementation. Overall, the methods presented provide highly efficient numerical solvers that are robust, simple to implement and produce second-order accurate solutions in the L^∞-norm on non-graded adaptive grids.

10 Acknowledgement

The research of F. Gibou was supported in part by ONR N00014-11-1-0027, NSF CHE 1027817, DOE DE-FG02-08ER15991, ICB W911NF-09-D-0001 and by the W.M. Keck Foundation. The research of C. Min was supported in part by the Priority Research Centers Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (2010-0028298) and by the Korea Research Foundation Grant funded by the Korean Government (KRF-2011-0013649). The research of R. Fedkiw was supported in part by ONR N00014-09-1-0101, ONR N-00014-11-1-0027, ARL AHPCRC W911NF-07-0027, NSF IIS-1048573, and the Intel Science and Technology Center for Visual Computing.

References

[1] D. Adalsteinsson and J. Sethian. A Fast Level Set Method for Propagating Interfaces. *J. Comput. Phys.*, 118:269–277, 1995.

[2] D. Adalsteinsson and J. Sethian. The Fast Construction of Extension Velocities in Level Set Methods. *J. Comput. Phys.*, 148:2–22, 1999.

[3] M. J. Aftosmis, M. J. Berger, and J. E. Melton. Adaptive Cartesian Mesh Generation. In *CRC Handbook of Mesh Generation (Contributed Chapter)*, 1998.

[4] H. T. Ahn and M. Shashkov. Adaptive moment-of-fluid method. *Journal of Computational Physics*, 228(8):2792–2821, 5 2009.

[5] V. Alexiades and A. D. Solomon. *Mathematical Modeling of Melting and Freezing Processes*. Hemisphere, Washingotn, DC, 1993.

[6] V. Alexiades, A. D. Solomon, and D. G. Wilson. The formation of a solid nucleus in supercooled liquid. I. *J, Non-Equilib. Thermodyn.*, 13:281–300, 1988.

[7] R. Almgren. Variational Algorithms and Pattern Formation in Dendritic Solidification. *Journal of Computational Physics*, 106(2):337–354, June 1993.

[8] T. Aslam. A partial differential equation approach to multidimensional extrapolation. *J. Comput. Phys.*, 193:349–355, 2004.

[9] E. Aulisa, S. Manservisi, R. Scardovelli, and S. Zaleski. A geometrical area-preserving volume-of-fluid advection method. *Journal of Computational Physics*, 192(1):355 – 364, 2003.

[10] B. N. Azarenok. A method of constructing adaptive hexahedral moving grids. *Journal of Computational Physics*, 226(1):1102–1121, 9 2007.

[11] J. Bedrossian, J. H. von Brecht, S. Zhu, E. Sifakis, and J. M. Teran. A second order virtual node method for elliptic problems with interfaces and irregular domains. *Journal of Computational Physics*, 229(18):6405 – 6426, 2010.

[12] D. Benson. Computational methods in Lagrangian and Eulerian hydrocodes. *Comput. Meth. in Appl. Mech. and Eng.*, 99:235–394, 1992.

[13] D. Benson. Volume of Fluid Interface Reconstruction Methods for Multimaterial Problems. *Applied Mechanics Reviews*, 52:151–165, 2002.

[14] M. Berger and P. Colella. Local adaptive mesh refinement for shock hydrodynamics. *J. Comput. Phys.*, 82:64–84, 1989.

[15] M. Berger and J. Oliger. Adaptive mesh refinement for hyperbolic partial differential equations. *J. Comput. Phys.*, 53:484–512, 1984.

[16] R. Beyer and R. LeVeque. Analysis of a one-dimensional model for the immersed boundary method. *SIAM J. Numer. Anal.*, 29:332, 1992.

[17] F. Black and M. Scholes. The pricing of options and corporate liabilities. *Journal of Political Economy*, 81(3), 1973.

[18] W. Bo, X. Liu, J. Glimm, and X. Li. A robust front tracking method: Verification and application to simulation of the primary breakup of a liquid jet. *SIAM J. Sci. Comput.*, 33(4):1505–1524, 2011.

[19] G. Bornia, A. Cervone, S. Manservisi, R. Scardovelli, and S. Zaleski. On the properties and limitations of the height function method in two-dimensional cartesian geometry. *Journal of Computational Physics*, 230(4):851–862, 2 2011.

[20] R. Braun and M. Murray. Adaptive Phase-Field Computations of Dendritic Crystal Growth. *J. Cryst Growth.*, 174:41, 1997.

[21] E. Brun, A. Guittet, and F. Gibou. A Local Level-Set Method Using a Hash Table Data Structure. *Journal of Computational Physics*, 2011.

[22] C. Burstedde, L. C. Wilcox, and O. Ghattas. `p4est`: Scalable algorithms for parallel adaptive mesh refinement on forests of octrees. *SIAM Journal on Scientific Computing*, 33(3):1103–1133, 2011.

[23] R. Caflisch, M. Gyure, B. Merriman, S. Osher, C. Ratsch, D. Vvedensky, and J. Zinck. Island Dynamics and the Level Set Method for Epitaxial Growth. *Applied Mathematics Letters*, 12:13, 1999.

[24] R. Caiden, R. Fedkiw, and C. Anderson. A Numerical Method for Two Phase Flow Consisting of Separate Compressible and Incompressible Regions. *J. Comput. Phys.*, 166:1–27, 2001.

[25] H. D. Ceniceros, R. L. Nós, and A. M. Roma. Three-dimensional, fully adaptive simulations of phase-field fluid models. *Journal of Computational Physics*, 229(17):6135–6155, 8 2010.

[26] H. Chen, C. Min, and F. Gibou. A Second-Order Accurate FDM for the Heat Equation on Irregular Domains and Adaptive Grids. In *Proceedings of the Materials Research Society Symposium, San Francisco, CA, USA*, volume 910, pages 907–910, 2006.

[27] H. Chen, C. Min, and F. Gibou. A Supra-Convergent Finite Difference Scheme for the Poisson and Heat Equations on Irregular Domains and Non-Graded Adaptive Cartesian Grids. *Journal of Scientific Computing*, 31(1-2):19–60, Mar. 2007.

[28] H. Chen, C. Min, and F. Gibou. A numerical scheme for the Stefan problem on adaptive Cartesian grids with supralinear convergence rate. *J. Comput. Phys.*, 228(16):5803–5818, 2009.

[29] S. Chen, B. Merriman, S. Osher, and P. Smereka. A Simple Level Set Method for Solving {S}tefan Problems. *J. Comput. Phys.*, 135:8–29, 1997.

[30] A. Chéné, C. Min, and F. Gibou. Second-Order Accurate Computation of Curvatures in a Level Set Framework Using Novel High-Order Reinitialization Schemes. *Journal of Scientific Computing*, 35(2-3):114–131, Dec. 2007.

[31] L.-T. Cheng and Y.-H. Tsai. Redistancing by flow of time dependent eikonal equation redistancing by flow of time dependent eikonal equation redistancing by flow of time dependent eikonal equation redistancing by flow of time dependent Eikonal equation. *J. Comp. Phys.*, pages 4002–4017, 2008.

[32] R. Crockett, P. Colella, and D. Graves. A cartesian grid embedded boundary method for solving the poisson and heat equations with discontinuous coefficients in three dimensions. *Journal of Computational Physics*, 230(7):2451 – 2469, 2011.

[33] S. Davis. *Theory of Solidification*. Cambridge University Press, Cambridge, England, 2001.

[34] R. DeBar. Fundamentals of the KRAKEN code. Technical report, Lawrence Livermore National Laboratory (UCID- 17366), 1974.

[35] dip Can and A. Prosperetti. A level set method for vapor bubble dynamics. *J. Comput. Phys.*, 231:1533–1552, 2012.

[36] V. Dyadechko and M. Shashkov. Moment-of-Fluid Interface Reconstruction. Technical report, Los Alamos National Laboratory (LA-UR-05-7571), 2006.

[37] K. Elder, M. Grant, N. Provatas, and J. Kosterlitz. Sharp interface limits of phase-field models. *SIAM J. Appl. Math*, 64:21604, 2001.

[38] D. Enright, R. Fedkiw, J. Ferziger, and I. Mitchell. A Hybrid Particle Level Set Method for Improved Interface Capturing. *J. Comput. Phys.*, 183:83–116, 2002.

[39] D. Enright, S. Marschner, and R. Fedkiw. Animation and Rendering of Complex Water Surfaces. *ACM Trans. Graph. (SIGGRAPH Proc.)*, 21(3):736–744, 2002.

[40] D. Enright, D. Nguyen, F. Gibou, and R. Fedkiw. Using the Particle Level Set Method and a Second Order Accurate Pressure Boundary Condition for Free Surface Flows. In *Proc. 4th ASME-JSME Joint Fluids Eng. Conf.*, number FEDSM2003–45144. ASME, 2003.

[41] D. Estep, S. Tavener, and T. Wildey. A posteriori error estimation and adaptive mesh refinement for a multiscale operator decomposition approach to fluid–solid heat transfer. *Journal of Computational Physics*, 229(11):4143–4158, 6 2010.

[42] R. Fedkiw, T. Aslam, B. Merriman, and S. Osher. A non-oscillatory Eulerian approach to interfaces in multimaterial flows (the ghost fluid method). *J. Comput. Phys.*, 152:457–492, 1999.

[43] R. Fedkiw, T. Aslam, and S. Xu. The Ghost Fluid Method for Deflagration and Detonation Discontinuities. *J. Comput. Phys.*, 154:393–427, 1999.

[44] F. Gibou, L. Chen, D. Nguyen, and S. Banerjee. A level set based sharp interface method for the multiphase incompressible navier–stokes equations with phase change. *Journal of Computational Physics*, 222(2):536–555, Mar. 2007.

[45] F. Gibou and R. Fedkiw. A Fourth Order Accurate Discretization for the Laplace and Heat Equations on Arbitrary Domains, with Applications to the Stefan Problem. *J. Comput. Phys.*, 202:577–601, 2005.

[46] F. Gibou, R. Fedkiw, R. Caflisch, and S. Osher. A Level Set Approach for the Numerical Simulation of Dendritic Growth. *Journal of Scientific Computing*, 19:183–199, 2003.

[47] F. Gibou, R. Fedkiw, L.-T. Cheng, and M. Kang. A Second–Order–Accurate Symmetric Discretization of the Poisson Equation on Irregular Domains. *J. Comput. Phys.*, 176:205–227, 2002.

[48] I. Glassman and R. Yetter. *Combustion*. A.P., 2008.

[49] G. Golub and C. Loan. *Matrix Computations*. The John Hopkins University Press, 1989.

[50] L. Greengard and J.-Y. Lee. Electrostatics and heat conduction in high contrast composite materials. *Journal of Computational Physics*, 211(1):64 – 76, 2006.

[51] B. E. Griffith, R. D. Hornung, D. M. McQueen, and C. S. Peskin. An adaptive, formally second order accurate version of the immersed boundary method. *Journal of Computational Physics*, 223(1):10–49, 4 2007.

[52] J. Heinrich and P. Zhao. Front Tracking Finite Element Method for Dendritic Solidification. *J. Comput. Phys.*, 173:765–796, 2001.

[53] J. Helmsen, E. G. Puckett, P. Colella, and M. Dorr. Two new methods for simulating photolithography development in 3D. In *Proceedings of the SPIE - The International Society for Optical Engineering Optical Microlithography IX, Santa Clara, CA, USA*, volume 13-15, pages 253–261.

[54] C. Hirt and B. Nichols. Volume of Fluid (VOF) Method for the Dynamics of Free Boundaries. *J. Comput. Phys.*, 39:201–225, 1981.

[55] S. Hou and X.-D. Liu. A numerical method for solving variable coefficient elliptic equation with interfaces. *Journal of Computational Physics*, 202(2):411 – 445, 2005.

[56] F. P. Incropera and D. P. DeWitt. *Fundamentals of heat and mass transfer*. John Wiley & Sons, 2007.

[57] J.-H. Jeong, N. Goldenfeld, and J. Dantzig. Phase Field Model for Three-Dimensional Dendritic Growth with Fluid Flow. *Phys. Rev. E*, 64:41602, 2001.

[58] H. Ji, F.-S. Lien, and E. Yee. A new adaptive mesh refinement data structure with an application to detonation. *Journal of Computational Physics*, 229(23):8981–8993, 11 2010.

[59] G.-S. Jiang and D. Peng. Weighted ENO Schemes for Hamilton-Jacobi Equations. *SIAM J. Sci. Comput.*, 21:2126–2143, 2000.

[60] G.-S. Jiang and C.-W. Shu. Efficient Implementation of Weighted ENO Schemes. *J. Comput. Phys.*, 126:202–228, 1996.

[61] H. Johansen. *Cartesian Grid Embedded Boundary Finite Difference Methods for Elliptic and Parabolic Differential Equations on Irregular Domains.* PhD thesis, Berkeley, 1997.

[62] H. Johansen and P. Colella. A Cartesian Grid Embedded Boundary Method for Poisson's Equation on Irregular Domains. *J. Comput. Phys.*, 147:60–85, 1998.

[63] J. L. H. Jr., L. Wang, E. Sifakis, and J. M. Teran. A second order virtual node method for elliptic problems with interfaces and irregular domains in three dimensions. *Journal of Computational Physics*, 231(4):2015 – 2048, 2012.

[64] D. Juric and G. Tryggvason. A Front Tracking Method for Dendritic Solidification. *J. Comput. Phys*, 123:127–148, 1996.

[65] D. Juric and G. Tryggvason. Computations of Boiling Flows. *Int. J. Multiphase. Flow.*, 24:387–410, 1998.

[66] S. J. Kamkar, A. M. Wissink, V. Sankaran, and A. Jameson. Feature-driven cartesian adaptive mesh refinement for vortex-dominated flows. *Journal of Computational Physics*, 230(16):6271–6298, 7 2011.

[67] A. Karma. Phase-Field Formulation for Quantitative Modeling of Alloy Solidification. *Phys. Rev. Lett.*, 87:115701, 2001.

[68] A. Karma and W.-J. Rappel. Phase-Field Modeling Method for Computationally Efficient Modeling of Solidification with Arbitrary Interface Kinetics. *Phys. Rev. E*, 53, 1996.

[69] A. Karma and W.-J. Rappel. Quantitative Phase-Field Modeling of Dendritic Growth in Two and Three Dimensions. *Phys. Rev. E*, 57:4323–4349, 1997.

[70] Y.-T. Kim, N. Goldenfeld, and J. Dantzig. Computation of Dendritic Microstructures using a Level Set Method. *Phys. Rev. E*, 62:2471–2474, 2000.

[71] H. O. Kreiss, H.-O. Manteuffel, T. A. Schwartz, B. Wendroff, and A. B. W. Jr. Supra-convergent schemes on irregular grids. *Math. Comp.*, 47:537–554, 1986.

[72] W. J. Laan, A. C. Jalba, and J. B. T. M. Roerdink. A Memory and Computation Efficient Sparse Level-Set Method. *Journal of Scientific Computing*, 46(2):243–264, July 2010.

[73] L. G. Leal. *Advanced transport Phenomena - Fluid mechanics and convective transport processes.* Cambridge series in chemical engineering, 2007.

[74] R. Leveque. High-Resolution Conservative Algorithms For Advection In Incompressible Flow. *SIAM J. Numer. Anal.*, 33:627–665, 1996.

[75] R. LeVeque and Z. Li. The Immersed Interface Method for Elliptic Equations with Discontinuous Coefficients and Singular Sources 31:1019–1044, 1994. *SIAM J. Numer. Anal.*, 31:1019–1044, 1994.

[76] S. Li and L. Petzold. Adjoint sensitivity analysis for time-dependent partial differential equations with adaptive mesh refinement. *Journal of Computational Physics*, 198(1):310–325, 7 2004.

[77] Z. Li and K. Ito. *The Immersed Interface Method – Numerical Solutions of PDEs Involving Interfaces and Irregular Domains*, volume 33. SIAM Frontiers in Applied mathematics, 2006.

[78] X. D. Liu, R. Fedkiw, and M. Kang. A Boundary Condition Capturing Method for Poisson's Equation on Irregular Domains. *J. Comput. Phys.*, 154:151, 2000.

[79] X.-D. Liu, S. Osher, and T. Chan. Weighted Essentially Non-Oscillatory Schemes. *J. Comput. Phys.*, 126:202–212, 1996.

[80] F. Losasso, R. Fedkiw, and S. Osher. Spatially Adaptive Techniques for Level Set Methods and Incompressible Flow. *Computers and Fluids*, 35:995–1010, 2006.

[81] F. Losasso, F. Gibou, and R. Fedkiw. Simulating Water and Smoke with an Octree Data Structure. *ACM Trans. Graph. (SIGGRAPH Proc.)*, pages 457–462, 2004.

[82] T. Manteuffel and A. White. The Numerical Solution of Second-Order Boundary Value Problems on Nonuniform Meshes. *Math. Comput.*, 47 (176):511–535, 1986.

[83] A. N. Marques, J.-C. Nave, and R. R. Rosales. A correction function method for poisson problems with interface jump conditions. *Journal of Computational Physics*, 230(20):7567 – 7597, 2011.

[84] A. Mayo. The Fast Solution of Poisson's and the Biharmonic Equations on Irregular Regions. *SIAM J. Numer. Anal.*, 21:285–299, 1984.

[85] P. McCorquodale, P. Colella, D. Grote, and J.-L. Vay. A Node-Centered Local Refinement Algorithm for Poisson's Equation in Complex Geometries. *J. Comput. Phys.*, 201:34–60, 2004.

[86] D. Meiron. Selection of Steady-States in the Two-Dimensional Symmetric Model of Dendritic Growth. *Phys. Rev. A.*, 33:2704, 1986.

[87] C. Min. Local level set method in high dimension and codimension. *J. Comput. Phys.*, 200:368–382, 2004.

[88] C. Min. On reinitializing level set functions. *Journal of Computational Physics*, 229(8):2764–2772, Apr. 2010.

[89] C. Min and F. Gibou. A second order accurate projection method for the incompressible navier–stokes equations on non-graded adaptive grids. *Journal of Computational Physics*, 219(2):912–929, Dec. 2006.

[90] C. Min and F. Gibou. A second order accurate level set method on non-graded adaptive Cartesian grids. *Journal of Computational Physics*, 225(1):300–321, 2007.

[91] C. Min, F. Gibou, and H. Ceniceros. A supra-convergent finite difference scheme for the variable coefficient Poisson equation on non-graded grids. *Journal of Computational Physics*, 218(1):123–140, Oct. 2006.

[92] F. Miniati and P. Colella. Block structured adaptive mesh and time refinement for hybrid, hyperbolic, n-body systems. *Journal of Computational Physics*, 227(1):400–430, 11 2007.

[93] D. Moore. The cost of balancing generalized quadtrees. In *Proceedings of the third ACM symposium on Solid modeling and applications*, pages 305–312, 1995.

[94] B. Nestler, D. Danilov, and P. Galenko. Crystal growth of pure substances: Phase-field simulations in comparison with analytical and experimental results. *J. Comput. Phys.*, 207:221–239, 2005.

[95] Y. T. Ng, C. Min, and F. Gibou. An efficient fluid–solid coupling algorithm for single-phase flows. *Journal of Computational Physics*, 228(23):8807–8829, Dec. 2009.

[96] D. Nguyen, R. Fedkiw, and H. Jensen. Physically based modeling and animation of fire. In *ACM Trans. Graph. (SIGGRAPH Proc.)*, volume 29, pages 721–728, 2002.

[97] D. Nguyen, R. Fedkiw, and M. Kang. A Boundary Condition Capturing Method for Incompressible Flame Discontinuities. *J. Comput. Phys.*, 172:71–98, 2001.

[98] D. Nguyen, F. Gibou, and R. Fedkiw. A Fully Conservative Ghost Fluid Method and Stiff Detonation Waves. In *12th Int. Detonation Symposium, San Diego, CA*, 2002.

[99] M. B. Nielsen and K. Museth. Dynamic Tubular Grid: An Efficient Data Structure and Algorithms for High Resolution Level Sets. *Journal of Scientific Computing*, 26(3):261–299, Jan. 2006.

[100] W. Noh and P. Woodward. SLIC (simple line interface calculation). In *5th International Conference on Numerical Methods in Fluid Dynamics*, pages 330–340, 1976.

[101] S. Osher and R. Fedkiw. *Level Set Methods and Dynamic Implicit Surfaces*. Springer-Verlag, 2002.

[102] S. Osher and N. Paragios. *Geometric Level Set Methods in Imaging, Vision, and Graphics*. Springer-Verlag, 2003.

[103] S. Osher and J. A. Sethian. Fronts propagating with curvature dependent speed: Algorithms based on hamilton-jacobi formulations. *J. Comput. Phys.*, 79(1):12–49, 1988.

[104] J. Papac, F. Gibou, and C. Ratsch. Efficient symmetric discretization for the Poisson, heat and Stefan-type problems with Robin boundary conditions. *Journal of Computational Physics*, 229(3):875–889, Feb. 2010.

[105] D. Peng, B. Merriman, S. Osher, H. Zhao, and M. Kang. A PDE-based fast local level set method. *J. Comput. Phys.*, 155:410–438, 1999.

[106] P. Perona and J. Malik. Scale-space and edge detection using anisotropic diffusion. *Proceedings of IEEE Computer Society Workshop on Computer Vision*, pages 16–22, 1987.

[107] C. Peskin. Flow Patterns Around Heart Valves: A Numerical Method. *J. Comput. Phys.*, 10:252–271, 1972.

[108] C. Peskin. Numerical Analysis of Blood Flow in the Heart. *J. Comput. Phys.*, 25:220–252, 1977.

[109] C. Peskin. The immersed boundary method. *Acta Numerica*, 11:479–517, 2002.

[110] S. Popinet. Gerris: A Tree-Based Adaptive Solver for the Incompressible Euler Equations in Complex Geometries. *J. Comput. Phys.*, 190:572–600, 2003.

[111] S. Popinet. An accurate adaptive solver for surface-tension-driven interfacial flows. *Journal of Computational Physics*, 228(16):5838–5866, September 2009.

[112] N. Provatas, N. Goldenfeld, and J. Dantzig. Efficient Computation of Dendritic Microstructures using Adaptive Mesh Refinement. *Phys. Rev. Lett.*, 80:3308, 1998.

[113] N. Provatas, N. Goldenfeld, and J. Dantzig. Adaptive Mesh Refinement Computation of Solidification Microstructure using Dynamic Data Structures. *J. Comput. Phys*, 148:265, 1999.

[114] J. W. Purvis and J. E. Burkhalter. Prediction of critical mach number for store configurations. *AIAA J.*, 17:1170–1177, 1979.

[115] J. Qian, G. Tryggvason, and C. Law. A Front Tracking Method for the Motion of Premixed Flames. *Journal of Computational Physics*, 144(1):52–69, July 1998.

[116] C. Ratsch, M. Gyure, F. Gibou, M. Petersen, M. Kang, J. Garcia, and D. Vvedensky. Level-Set Method for Island Dynamics in Epitaxial Growth. *Phys. Rev. B.*, 65:195403, 2002.

[117] Y. Renardy and M. Renardy. Prost: A parabolic reconstruction of surface tension for the volume-of-fluid method. *Journal of Computational Physics*, 183(2):400 – 421, 2002.

[118] G. Russo and P. Smereka. A Remark on Computing Distance Functions. *J. Comput. Phys.*, 163:51–67, 2000.

[119] Y. Saad. *Iterative methods for sparse linear systems*. PWS Publishing, 1996.

[120] Y. Saad. *Iterative Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics, 2003.

[121] H. Samet. *The Design and Analysis of Spatial Data Structures*. Addison-Wesley, New York, 1989.

[122] H. Samet. *Applications of Spatial Data Structures: Computer Graphics, Image Processing and GIS*. Addison-Wesley, New York, 1990.

[123] R. S. Sampath and G. Biros. A parallel geometric multigrid method for finite elements on octree meshes, siam journal on scientific computing. *SIAM J. of Scientific Comput.*, 32(3):1361–1392, 2010.

[124] J. Sethian. A Fast Marching Level Set Method for Monotonically Advancing Fronts. *Proc. Natl. Acad. Sci.*, 93:1591–1595, 1996.

[125] J. Sethian. Fast Marching Methods. *SIAM Review*, 41:199–235, 1999.

[126] J. Sethian and J. Strain. Crystal Growth and Dendritic Solidification. *J. Comput. Phys.*, 98:231–253, 1992.

[127] J. A. Sethian. *Level set methods and fast marching methods*. Cambridge University Press, 1999.

[128] C. Shen, J.-M. Qiu, and A. Christlieb. Adaptive mesh refinement based on high order finite difference weno scheme for multi-scale simulations. *Journal of Computational Physics*, 230(10):3780–3802, 5 2011.

[129] G. H. Shortley and R. Weller. Numerical solution of laplace's equation. *J. Appl. Phys.*, 9:334–348, 1938.

[130] C.-W. Shu and S. Osher. Efficient Implementation of Essentially Non-Oscillatory Shock Capturing Schemes II. *J. Comput. Phys.*, 83:32–78, 1989.

[131] G. Son and V. Dhir. Numerical Simulation of Saturated Film Boiling on a Horizontal Surface. *J. Heat. Transfer.*, 119:525, 1997.

[132] G. Son and V. Dhir. Numerical Simulation of Film Boiling Near Critical Pressures with a Level Set Method. *J. Heat. Transfer.*, 120:183, 1998.

[133] G. Son and V. Dhir. A level set method for analysis of film boiling on an immersed liquid surface. *Numer. Heat Transf.*, B52:153–177, 2007.

[134] G. Son and V. Dhir. Numerical simulation of nucleate boiling on a horizontal surface at high heat fluxes. *Int. J. Heat Mass Transf.*, 51:2566–2582, 2008.

[135] J. Strain. Fast Tree-Based Redistancing for Level Set Computations. *J. Comput. Phys.*, 152:664–686, 1999.

[136] J. Strain. Tree Methods for Moving Interfaces. *J. Comput. Phys.*, 151:616–648, 1999.

[137] P. Sun, R. D. Russell, and J. Xu. A new adaptive local mesh refinement algorithm and its application on fourth order thin film flow problem. *Journal of Computational Physics*, 224(2):1021–1048, 6 2007.

[138] M. Sussman, A. Almgren, J. Bell, P. Colella, L. Howell, and M. Welcome. An adaptive level set approach for incompressible two-phase flows. *J. Comput. Phys.*, 148:81–124, 1999.

[139] M. Sussman and E. G. Puckett. A coupled level set and volume-of-fluid method for computing 3d and axisymmetric incompressible two-phase flows. *Journal of Computational Physics*, 162(2):301 – 337, 2000.

[140] M. Sussman, P. Smereka, and S. Osher. A Level Set Approach for Computing Solutions to Incompressible Two-Phase Flow. *Journal of Computational Physics*, 114:146–159, 1994.

[141] L. Tan and N. Zabaras. A level set simulation of dendritic solidification of multi-component alloys. *Journal of Computational Physics*, 221(1):9–40, Jan. 2007.

[142] L. Tan and N. Zabaras. Modeling the growth and interaction of multiple dendrites in solidification using a level set method. *Journal of Computational Physics*, 226(1):131–155, Sept. 2007.

[143] H.-Z. Tang, T. Tang, and P. Zhang. An adaptive mesh redistribution method for nonlinear Hamilton-Jacobi equations in two- and three-dimensions. *J. Comput. Phys.*, 188:543–572, 2003.

[144] M. Theillard, C. H. Rycroft, and F. Gibou. A multigrid method on non-graded adaptive octree and quadtree cartesian grids. *J. Scientific Comput.*, 2012.

[145] G. Tomar, G. Biswas, A. Sharma, and A. Agrawal. Numerical simulation of bubble growth in film boiling using a coupled level-set and volume-of-fluid method. *Phys. of Fluid*, 17:112103, 2005.

[146] G. Tryggvason, B. Bunner, A. Esmaeeli, D. Juric, N. Al-Rawahi, W. Tauber, J. Han, S. Nas, and Y.-J. Jan. A Front-Tracking Method for the Computations of Multiphase Flow. *J. Comput. Phys.*, 169:708–759, 2001.

[147] Y.-H. Tsai, L.-T. Cheng, and S. Osher. Fast sweeping algorithms for a class of Hamilton-Jacobi equations. *SIAM J. Numer. Anal.*, 41(2):673–694, 2003.

[148] Y.-H. R. Tsai. Rapid and accurate computation of the distance function using grids. *J. Comp. Phys.*, 178(1):175–195, 2002.

[149] J. Tsitsiklis. Efficient Algorithms for Globally Optimal Trajectories. *IEEE Trans. on Automatic Control*, 40:1528–1538, 1995.

[150] H. Udaykumar, H.-C. Kan, W. Shyy, and R. Tran-Son-Tay. Multiphase dynamics in arbitrary geometries on fixed cartesian grids. *Journal of Computational Physics*, 137(2):366 – 405, 1997.

[151] H. Udaykumar, S. Marella, and S. Krishnan. Sharp-interface simulation of dendritic growth with convection: benchmarks. *International Journal of Heat and Mass Transfer*, 46(14):2615 – 2627, 2003.

[152] H. Udaykumar, H. Mittal, and W. Shyy. Computation of Solid-Liquid Phase Fronts in the Sharp Interface Limit on Fixed Grids. *J. Comput. Phys.*, 153:535–574, 1999.

[153] S. O. Unverdi and G. Tryggvason. A front-tracking method for viscous, incompressible, multifluid flows. *J. Comput. Phys.*, 100:25–37, 1992.

[154] Z. Wang, J. Yang, and F. Stern. A new volume-of-fluid method with a constructed distance function on general structured grids. *Journal of Computational Physics*, 231(9):3703 – 3722, 2012.

[155] A. Weiser. *Local-Mesh, Local-Order, Adaptive Finite Element Methods with a Posteriori Error Estimators for Elliptic Parital Differential Equations.* PhD thesis, Yale University, June 1981.

[156] S. Welch and J. Wilson. A volume of fluid based method for fluid flows with phase change. *J. Comput. Phys.*, 160:662–682, 2000.

[157] S. Whitaker. *Introduction to fluid mechanics.* Prentice Hall, 1968.

[158] N.-A. A. X.-Y. Hu. Scale separation for implicit large eddy simulation. *J. Comp. Phys.*, 230(19):7240–7249, 2011.

[159] D. Xiu and G. Karniadakis. A Semi-Lagrangian High-Order Method for Navier-Stokes Equations. *J. Comput. Phys*, 172:658–684, 2001.

[160] G. Z. Yang and N. Zabaras. The adjoint method for an inverse design problem in the directional solidification of binary alloys. *Journal of Computational Physics*, 140(2):432 – 452, 1998.

[161] X. Yang, A. J. James, J. Lowengrub, X. Zheng, and V. Cristini. An adaptive coupled level-set/volume-of-fluid interface capturing method for unstructured triangular grids. *Journal of Computational Physics*, 217(2):364 – 394, 2006.

[162] Y. Yang and H. Udaykumar. Sharp interface cartesian grid method iii: Solidification of pure materials and binary solutions. *Journal of Computational Physics*, 210(1):55 – 74, 2005.

[163] D. Youngs. An interface tracking method for a 3D Eulerian hydrodynamics code. Technical report, AWRE (44/92/35), 1984.

[164] N. Zabaras, B. Ganapathysubramanian, and L. Tan. Modelling dendritic solidification with melt convection using the extended finite element method. *Journal of Computational Physics*, 218(1):200–227, Oct. 2006.

[165] Q. Zhang and P. Liu. Handling solid-fluid interfaces for viscous flows: Explicit jump approaximation vs. ghost cell approaches. *J. Comput. Phys.*, 229(11):4225–4246, 2010.

[166] H. Zhao. A fast sweeping method for eikonal equations. *Mathematics of Computation*, 74:603–627, 2004.

[167] P. Zhao, M. Vénere, J. Heinrich, and D. Poirier. Modeling dendritic growth of a binary alloy. *Journal of Computational Physics*, 188(2):434–461, July 2003.

[168] L. Zhu and C. Peskin. Simulation of a Flapping Flexible Filament in a Flowing Soap Film by the Immersed Boundary Method. *J. Comput. Phys.*, 179:452–468, 2002.