A Second Order Accurate Level Set Method on Non-Graded Adaptive Cartesian Grids

Chohong Min * Frédéric Gibou †

30th November 2006

Abstract

We present a level set method on non-graded adaptive Cartesian grids, i.e. grids for which the ratio between adjacent cells is not constrained. We use quadtree and octree data structures to represent the grid and a simple algorithm to generate a mesh with the finest resolution at the interface. In particular, we present (1) a locally third order accurate reinitialization scheme that transforms an arbitrary level set function into a signed distance function, (2) a second order accurate semi-Lagrangian methods to evolve the linear level set advection equation under an externally generated velocity field, (3) a second order accurate upwind method to evolve the nonlinear level set equation under a normal velocity as well as to extrapolate scalar quantities across an interface in the normal direction, and (4) a semi-implicit scheme to evolve the interface under mean curvature. Combined, we obtain a level set method on adaptive Cartesian grids with a negligible amount of mass loss. We propose numerical examples in two and three spatial dimensions to demonstrate the accuracy of the method.

1 Introduction

Many problems in science and engineering can be described by a moving free boundary model. Examples include free surface flows, Stefan problems and multiphase flows to cite a few. The difficulty in solving these problems stems from the fact that: First, they involve dissimilar length scales. Second, the boundary position must be computed as part of the solution process. Third, the interface may be expected to undergo complex topological changes, such as the merging or the pinching of two fronts. Numerically, the interface that separates the two phases can be either explicitly tracked or implicitly captured. Several classes of successful methods exist with their own virtues and drawbacks.

Volume of Fluid methods [3, 4, 9, 27, 44, 66] have the advantage of being volume preserving since the mass fraction in each cell is being tracked. However, it is often difficult to extract geometrical properties such as curvatures due to the fact that it is challenging or even impossible to reconstruct a smooth enough function from the mass fractions alone. We note however that some recent improvement in interface reconstruction can be found in [11].

The main advantage of an explicit approach, e.g. front tracking [25, 28, 29, 62], is its accuracy. The main disadvantage is that additional treatments are needed for handling changes in the interface's topology. In turn, the explicit treatment of connectivity makes the method challenging to extend to three spatial dimensions. While researchers have produced remarkable results for a wide variety of applications using front tracking techniques, these difficulties make this approach not ideally suited for studying interface problems with changes in topology. Implicit representations such as the level set method or the phase-field method represent the front as an isocontour of a continuous function. Topological changes are consequently handled in a straightforward fashion, and thus these methods are readily implemented in both two and three spatial dimensions.

The main idea behind the phase-field method is to distinguish between phases with an order parameter (or phase-field) that is constant within each phase but varies smoothly across an interfacial region of finite thickness. The dynamics of the phase-field is then coupled to that of the solution in such a way that it tracks the interface motion and approximates the sharp interface limit when the order parameter vanishes. Phase-field methods are very popular techniques for simulating dendritic growth for example and have produced accurate quantitative results, e.g. [33, 32, 30, 41, 54]. However, these methods suffer from their own limitations: Phase-field methods have only an approximate

*Mathematics Department, University of California, Santa Barbara, CA 93106.

†Mechanical Engineering Department & Computer Science Department, University of California, Santa Barbara, CA 93106.

representation of the front location and thus the discretization of the diffusion field is less accurate near the front, resembling an enthalpy method [7]. Another consequence is the stringent time step restriction imposed by such methods. Karma and Rappel [31] have developed a thin-interface limit of the phase-field model with a significant improvement of the capillary length to interface thickness ratio constraint; however, the time step restriction is still on the order of the microscopic capillary length. Another disadvantage is the potential difficulty in relating the phase-field parameters to the physical parameters [64], although some progress is being made for some wider class of problems [12].

The main difference between the phase-field method and the level set approach [48, 55, 46] is that the level set method is a sharp interface model. The level set can therefore be used to exactly locate the interface in order to apply discretizations that depend on the exact interface location. Consequently, the sharp interface equation can be solved directly with no need for asymptotic analysis, which makes the method potentially more attractive in developing general tool box software for a wide range of applications. Another advantage is that only the standard time step restrictions for stability and consistency are required, making the method significantly more efficient. Level set methods have been extremely successful on uniform grids in the study of physical problems such as compressible flows, incompressible flows, multiphase flows (see e.g. [46, 55] and the references therein), Epitaxial growth (see e.g.[5, 23, 24, 50] and the references therein) or in image processing (see e.g. [47] and the references therein). One of the main problem of the level set method, namely its mass loss, has been partially solved with the advent of the particle level set method of Enright *et al.* [13]. Within this method, the interface is captured by the level set method and massless particles are added in order to reduce the mass loss. The massless particles are also used in the reinitialization process for obtaining smoother results for the reinitialized level set function. However, the use of particles adds to the CPU and the memory requirement and cannot be applied for flows producing shocks. Rather recently, there has been a thrust in developing level set methods on adaptive Cartesian grids. For example Losasso *et al.* [36] presented a particle level set based method to simulate free surface flows on non-graded Cartesian grids. Within this method, the interface between the liquid and the air is captured by the particle level set on a non graded octree data structure. Other interesting work on adaptive level-set methods can be found in [10] and [37].

In this paper, we present a general particle-less level set method on non-graded Cartesian grids that produces a negligible amount of mass loss. We apply this method to the level set evolution (1) with an externally generated velocity field, (2) in the normal direction and (3) under mean curvature. We also present a locally third order accurate reinitialization scheme that transforms an arbitrary function into a sign distance function as well as standard techniques to extrapolate a scalar quantities across an interface in its normal direction.

2 The Level Set Method

The level set method, introduced by Osher and Sethian [48] describes 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. For example, in two spatial dimension, a curve is define by $\{(x, y) : \phi(x, y) = 0\}$. Under a velocity field V, the interface deforms according to the level set equation

$$\phi_t + V \cdot \nabla\phi = 0. \tag{1}$$

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.* [61]

$$\phi_\tau + S(\phi_o)\left(|\nabla\phi| - 1\right) = 0 \tag{2}$$

is traditionally iterated for a few steps in fictitious time, τ. Here $S(\phi_o)$ is a smoothed out sign function. The level set function is used to compute the normal

$$\vec{n} = \nabla\phi/|\nabla\phi|,$$

and the mean curvature

$$\kappa = \nabla \cdot \vec{n}.$$

We refer the interested readers to the book by Osher and Fedkiw [46] as well to the book by Sethian [55] for more details on the level set method.

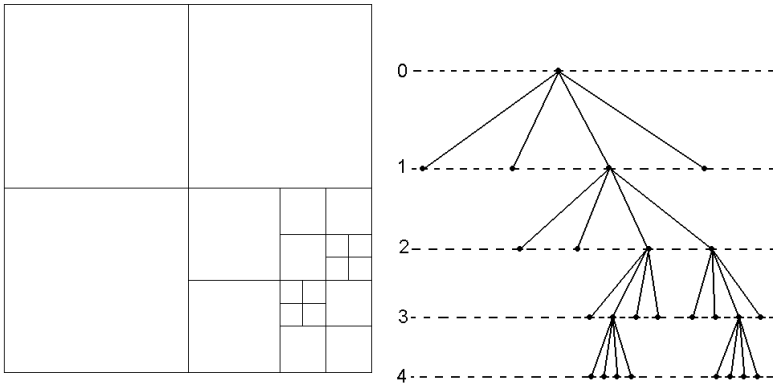

Figure 1: Discretization of a two dimensional domain (left) and its quadtree representation (right). The entire domain corresponds to the root of the tree (level 0). Then each cell can be recursively subdivided further into four children. In this example, the tree is ungraded since the difference of level between cells exceeds one.

3 Spatial Discretization and Refinement Criterion

We use a standard quadtree (resp. octree) data structure to represent the spatial discretization of the physical domain in two (resp. three) spatial dimensions as depicted in figure 1: Initially the root of the tree is associated with the entire domain, then we recursively split each cell into four children until the desired level of detail is achieved. This is done similarly in three spatial dimensions, except that cells are split into eight cubes (children). We refer the reader to the books of Samet [53, 52] for more details on quadtree/octree data structures.

By definition, the difference of level between a parent cell and its direct descendant is one. The level is then incremented by one for each new generation of children. A tree in which the difference of level between adjacent cells is at most one is called a graded tree. Meshes associated with graded trees are often used in the case of finite element methods in order to produce procedures that are easier to implement. Graded Cartesian grids are also used in the case of finite difference schemes, see for example the work of Popinet [49] for the study of incompressible flows. Graded meshes impose that extra grid cells must be added in regions where they are not necessarily needed, consuming some computational resources that cannot be spent elsewhere, eventually limiting the highest level of detail that can be achieved. Moore [40] demonstrates that the cost of transforming an arbitrary quadtree into a graded quadtree could involve 8 times as many grid nodes. Weiser [63] proposed a rough estimate for the three dimensional case and concluded that as much as 71 times as many grid nodes could be needed for balancing octrees. These estimates clearly represent the worse case scenarios that seldom exist in practical simulations. However, there is still a non negligible difference between graded and non graded grids. In addition, not imposing any constraint on the difference of level between two adjacent cells allows for easier/faster adaptive mesh generations.

In this work we choose to impose that the finest cells lie on the interface, since it is the region of interest for the level set method. In order to generate adaptive Cartesian grids, one can use the signed distance function to the interface along with the Whitney decomposition, as first proposed by Strain in [58]. Simply stated, one *"splits any cell whose edge length exceeds its distance to the interface"*. For a general function $\phi : \mathbb{R}^n \to \mathbb{R}$ with Lipschitz constant $Lip(\phi)$, the Whitney decomposition was extended in Min [39]: Starting from a root cell split any cell C if

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

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.

4 Finite Difference Discretizations

In the case of non regular Cartesian grids, the main difficulty comes from deriving 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 2 depicts

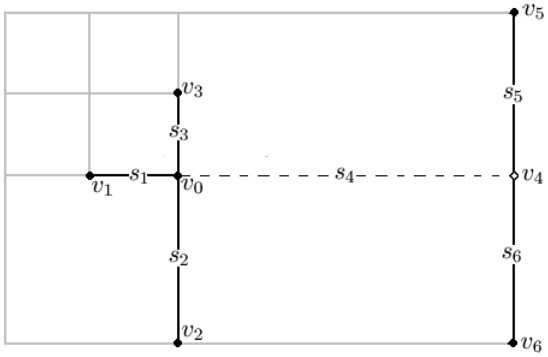

Figure 2: Neighboring nodes of a T-junction node, v_0.

a T-junction node v_0, with three neighboring nodes v_1, v_2 and v_3 aligned in the Cartesian directions and one ghost neighboring node v_4 replacing the missing grid node in the positive Cartesian direction. The value of a node-sampled function $\phi : \{v_i\} \rightarrow \mathbb{R}$ at the ghost node v_4 could for example be define by linear interpolation:

$$\phi_4^G = \frac{\phi_5 s_6 + \phi_6 s_5}{s_5 + s_6}. \tag{3}$$

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 3 is given by:

$$\phi_4^G = \frac{\phi_5 s_6 + \phi_6 s_5}{s_5 + s_6} = \phi(v_4) + \frac{s_5 s_6}{2}\phi_{yy}(v_0) + O(\triangle x_{\text{smallest}})^3, \tag{4}$$

where $\triangle x_{\text{smallest}}$ is the size of the smallest grid cell with vertex v_0. The term $\phi_{yy}(v_0)$ can be approximated using the standard first order accurate discretization $\frac{2}{s_2+s_3}\left(\frac{\phi_2-\phi_0}{s_2} + \frac{\phi_3-\phi_0}{s_3}\right)$ and cancelled out in equation 4 to give:

$$\phi_4^G = \frac{\phi_5 s_6 + \phi_6 s_5}{s_5 + s_6} - \frac{s_5 s_6}{s_2 + s_3}\left(\frac{\phi_2 - \phi_0}{s_2} + \frac{\phi_3 - \phi_0}{s_3}\right). \tag{5}$$

We also point out that this interpolation only uses the node values of the cells adjacent to v_0, which is particularly beneficial since access to cells not immediately adjacent to the current cell is more difficult and could add on CPU time and/or memory requirement.

In three spatial dimensions, similar interpolation procedures can be used to define the value of ϕ at ghost nodes. Referring to figure 3, a T-junction node v_0 has four regular neighboring nodes and two ghost nodes. The values of a node-sampled function $\phi : \{v_i\} \rightarrow \mathbb{R}$ at the ghost nodes v_4 and v_5 can be defined by second order linear and bilinear interpolations as:

$$\begin{aligned} \phi_4^G &= \frac{s_7\phi_8 + s_8\phi_7}{s_7 + s_8}, \\ \phi_5^G &= \frac{s_{11}s_{12}\phi_{11} + s_{11}s_9\phi_{12} + s_{10}s_{12}\phi_9 + s_{10}s_9\phi_{10}}{(s_{10} + s_{11})(s_9 + s_{12})}. \end{aligned} \tag{6}$$

As in the case of quadtrees, third order accurate interpolations can be derived by cancelling out the second order

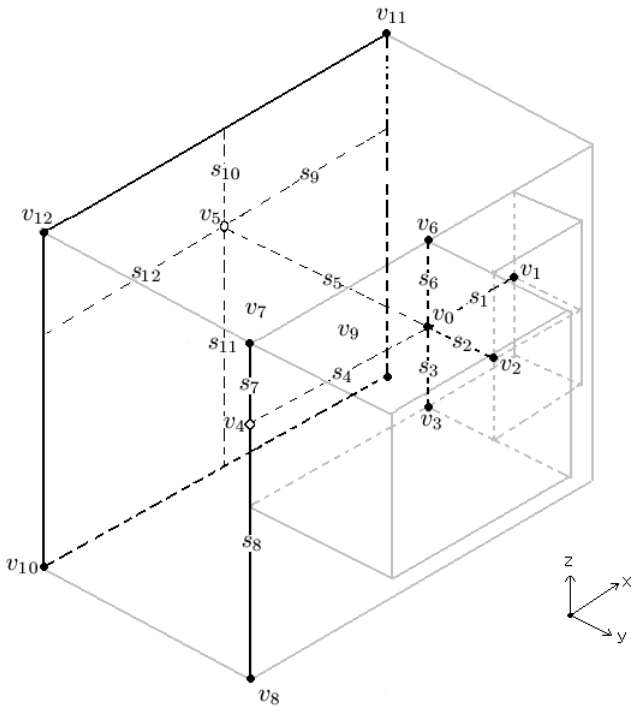

Figure 3: Neighboring vertices of a vertex three spatial dimensions.

derivatives in the error term to arrive at:

$$\begin{aligned}
\phi_4^G &= \frac{s_7\phi_8 + s_8\phi_7}{s_7 + s_8} - \frac{s_7 s_8}{s_3 + s_6}\left(\frac{\phi_3 - \phi_0}{s_3} + \frac{\phi_6 - \phi_0}{s_6}\right), \\
\phi_5^G &= \frac{s_{11}s_{12}\phi_{11} + s_{11}s_9\phi_{12} + s_{10}s_{12}\phi_9 + s_{10}s_9\phi_{10}}{(s_{10} + s_{11})(s_9 + s_{12})} \\
&\quad - \frac{s_{10}s_{11}}{s_3 + s_6}\left(\frac{\phi_3 - \phi_0}{s_3} + \frac{\phi_6 - \phi_0}{s_6}\right) \\
&\quad - \frac{s_9 s_{12}}{s_1 + s_4}\left(\frac{\phi_1 - \phi_0}{s_1} + \frac{\phi_4^G - \phi_0}{s_4}\right).
\end{aligned} \tag{7}$$

We emphasize that figure 3 represents the general configuration of neighboring nodes in the case of an octree as described in Min *et al.* [38].

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 then define finite differences for ϕ_x, ϕ_y, ϕ_z, ϕ_{xx}, ϕ_{yy} and ϕ_{zz} at every nodes using standard finite difference formulas in a dimension by dimension framework. For example, refering to figure 4, we use the standard discretization for ϕ_x and ϕ_{xx}, namely the central difference formulas:

$$\begin{aligned}
D_x^0\phi_0 &= \frac{\phi_2 - \phi_0}{s_2}\cdot\frac{s_1}{s_1 + s_2} + \frac{\phi_0 - \phi_1}{s_1}\cdot\frac{s_2}{s_1 + s_2}, \\
D_{xx}^0\phi_0 &= \frac{\phi_2 - \phi_0}{s_2}\cdot\frac{2}{s_1 + s_2} - \frac{\phi_0 - \phi_1}{s_1}\cdot\frac{2}{s_1 + s_2},
\end{aligned} \tag{8}$$

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

$$\begin{aligned}
D_x^+\phi_0 &= \frac{\phi_2 - \phi_0}{s_2}, \\
D_x^-\phi_0 &= \frac{\phi_0 - \phi_1}{s_1},
\end{aligned} \tag{9}$$

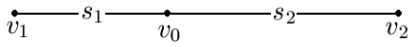

Figure 4: One dimensional adaptive grid

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

$$\begin{aligned} D_x^+\phi_0 &= \frac{\phi_2-\phi_0}{s_2} - \frac{s_2}{2}\text{minmod}\left(D_{xx}^0\phi_0, D_{xx}^0\phi_2\right), \\ D_x^-\phi_0 &= \frac{\phi_0-\phi_1}{s_1} + \frac{s_1}{2}\text{minmod}\left(D_{xx}^0\phi_0, D_{xx}^0\phi_1\right), \end{aligned} \tag{10}$$

where we use the minmod slope limiter [56, 34] because it produces more stable results in region where ϕ might present kinks. Similarly, approximations for first and second order derivatives are obtained in the y and z directions.

5 Interpolation Procedures

Some reserve must be provided to define data anywhere in a cell, for example in order to use semi-Lagrangian methods (see section 7). As pointed out in Strain [59], the most natural choice of interpolation in quadtree (resp. octree) data structures is the piecewise bilinear (resp. trilinear) interpolation: Consider a cell C with dimensions $[0,1]^2$, the bilinear interpolation at a point $x \in C$ using the values at the nodes reads:

$$\begin{aligned} \phi(x,y) = \phi(0,0)(1-x)(1-y) \\ + \phi(0,1)(1-x)(\quad y) \\ + \phi(1,0)(\quad x)(1-y) \\ + \phi(1,1)(\quad x)(\quad y) \end{aligned} \tag{11}$$

Quadratic interpolation can also easily be constructed using the data from the parent cell: Since the parent cell of any current cell of a quadtree (resp. octree) owns 2×2 children cells (resp. $2\times 2\times 2$) and 3×3 nodes (resp. $3\times 3\times 3$), one can defined the multidimensional Lagrange quadratic interpolation on the parent cell. For example in the case of a cell $[-1,1]^2$ in a quadtree, we can define the Lagrange interpolation as:

$$\begin{aligned} \phi(x,y) = \phi(\text{-}1,\text{-}1)\frac{x(x-1)}{2}\frac{y(y-1)}{2} + \phi(0,\text{-}1)(x^2-1)\frac{y(y-1)}{2} + \phi(1,\text{-}1)\frac{x(x+1)}{2}\frac{y(y-1)}{2} \\ + \phi(\text{-}1,\ 0)\frac{x(x-1)}{2}(y^2-1) \quad + \phi(0,\ 0)(x^2-1)(y^2-1) + \phi(1,\ 0)\frac{x(x+1)}{2}(y^2-1) \\ + \phi(\text{-}1,\ 1)\frac{x(x-1)}{2}\frac{y(y+1)}{2} + \phi(0,\ 1)(x^2-1)\frac{y(y+1)}{2} + \phi(1,\ 1)\frac{x(x+1)}{2}\frac{y(y+1)}{2}. \end{aligned}$$

However, this interpolation procedure is sensitive to nearby discontinuities, e.g. near kinks. We therefore prefer to define a quadratic interpolation by correcting equation (11) using second order derivatives. For a cell $[0,1]^2$, we have:

$$\begin{aligned} \phi(x,y) = \phi(0,0)(1-x)(1-y) \\ + \phi(0,1)(1-x)(\quad y) \\ + \phi(1,0)(\quad x)(1-y) \\ + \phi(1,1)(\quad x)(\quad y) - \phi_{xx}\frac{x(1-x)}{2} - \phi_{yy}\frac{y(1-y)}{2}, \end{aligned} \tag{12}$$

where we define

$$\begin{aligned} \phi_{xx} &= \min_{v\in\text{vertices}(C)}(|D_{xx}^0\phi_v|), \\ \phi_{yy} &= \min_{v\in\text{vertices}(C)}(|D_{yy}^0\phi_v|). \end{aligned} \tag{13}$$

Since a distant function is piecewise differentiable in general, the choice of the smallest in absolute value enhances the numerical stability of the interpolation.

6 Reinitialization Scheme

In principle, the level function can be chosen as any Lipschitz continuous function. However, the so-called signed distance function is known to produce more robust numerical results, to improve mass conservation and to reduce errors in the computations of geometrical quantities such as the interface curvatures. Sussman *et al.* proposed in [60] to evolve the following partial differential equation to steady state in order to reinitialize a level set function $\phi^0 : \mathbb{R}^n \to \mathbb{R}$ into the signed distance function ϕ:

$$\phi_\tau + \mathrm{sgn}(\phi^0)\left(|\nabla\phi| - 1\right) = 0, \tag{14}$$

where τ represents the fictitious time. A standard discretization for this equation in its semi-discrete form is given by:

$$\frac{d\phi}{d\tau} + \mathrm{sgn}(\phi^0)\left[H_G\left(D_x^+\phi, D_x^-\phi, D_y^+\phi, D_y^-\phi\right) - 1\right] = 0, \tag{15}$$

where $\mathrm{sgn}(\phi^0)$ denotes the signum of ϕ^0 and H_G is the Godunov Hamiltonian defined as:

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

with $a^+ = \max(a, 0)$ and $a^- = \min(a, 0)$. The one-sided derivatives, $D_x^\pm\phi$ and $D_y^\pm\phi$ are discretized by the second order accurate one-sided finite differences defined in section 4. Equation (15) is evolved in time with the TVD RK-2 method given in Shu and Osher [56]: First define $\tilde{\phi}^{n+1}$ and $\tilde{\phi}^{n+2}$ by Euler steps

$$\begin{aligned} \frac{\tilde{\phi}^{n+1} - \phi^n}{\triangle\tau} &+ \mathrm{sgn}(\phi^0)\left[H_G\left(D_x^+\phi^n, D_x^-\phi^n, D_y^+\phi^n, D_y^-\phi^n\right) - 1\right] = 0, \\ \frac{\tilde{\phi}^{n+2} - \tilde{\phi}^{n+1}}{\triangle\tau} &+ \mathrm{sgn}(\phi^0)\left[H_G\left(D_x^+\tilde{\phi}^{n+1}, D_x^-\tilde{\phi}^{n+1}, D_y^+\tilde{\phi}^{n+1}, D_y^-\tilde{\phi}^{n+1}\right) - 1\right] = 0, \end{aligned}$$

and then define ϕ^{n+1} by averaging:

$$\phi^{n+1} = \frac{\phi^n + \tilde{\phi}^{n+2}}{2}.$$

In order to preserve area/volume, the reinitialization procedure is required not to move the original interface defined by ϕ_0. In their seminal work, Russo and Smereka [51] solved this problem by simply including the initial interface location (given by ϕ_0) in the stencils of the one-sided derivatives. Consider the case depicted by figure 4 and suppose that $\phi_0^0 \cdot \phi_2^0 < 0$, i.e. the interface is located between the nodes v_0 and v_2. The interface location v_I can be calculated by finding the root of the quadratic interpolation of ϕ^0 on the interval $\overline{v_0 v_2}$ with the origin at the center of the interval:

$$\phi^0(x) = c_2x^2 + c_1x + c_2, \text{ with } \begin{cases} c_2 &= \frac{1}{2}\mathrm{minmod}\left[D_{xx}^0\phi_0^0, D_{xx}^0\phi_2^0\right] \\ c_1 &= (\phi_2^0 - \phi_0^0)/s_2 \\ c_0 &= (\phi_2^0 + \phi_0^0)/2 - c_2s_2^2/4 \end{cases}.$$

The distance s_I between v_0 and the interface location is then defined by:

$$s_I = \frac{s_2}{2} + \begin{cases} -c_0/c_1 & \text{if } |c_2| < \epsilon \\ (-c_1 + \sqrt{c_1^2 - 4c_2c_0})/(2c_2) & \text{if } |c_2| \ge \epsilon \text{ and } \phi_0^0 < 0 \\ (-c_1 - \sqrt{c_1^2 - 4c_2c_0})/(2c_2) & \text{if } |c_2| \ge \epsilon \text{ and } \phi_0^0 > 0 \end{cases}.$$

The calculation of $D_x^+\phi_0^n$ is then modified using the interface location and the fact that $\phi = 0$ at the interface:

$$D_x^+\phi_0^n = \frac{0 - \phi_0^n}{s_I} - \frac{s_I}{2}\mathrm{minmod}\left(D_{xx}^0\phi_0^n, D_{xx}^0\phi_2^n\right).$$

We note that in the original work of Russo and Smereka [51], a cubic interpolation was employed to locate the interface, but that the above quadratic interpolation with the minmod operator acting on the second order derivatives

proved to be more stable in the case where the level set function presents a kink nearby. We also point out that in the original work of [51], the first order derivative $D_x^+\phi_0^n$ was discretized as:

$$D_x^+\phi_0^n = \frac{0-\phi_0^n}{s_I} - \frac{s_I}{2}\text{minmod}\left(D_{xx}^0\phi_0^n, D_{xx}^0\phi_I^n\right),$$

thus included v_I in the discretization of $D_{xx}^0\phi_I^n$. However, we found that this choice leads to unstable results when the interface is close to grid nodes. We thus slightly changed the discretization by only using the location of the interface in the first term in order to maintain the location of ϕ^0, not in the discretization of second order derivatives. Likewise, in the case where s_I is close to zero (hence ϕ_0^0 is close to zero) we simply set $\phi_0^n = 0$ to guarantee stability. This only introduces a negligible perturbation in the location of the zero level set.

The same process is then applied to $D_x^-\phi$ if there is a sign change between ϕ_0^0 and ϕ_1^0. The time step restriction for cells cut by the interface is then:

$$\Delta\tau = \begin{cases} \min(s_I, s_1, s_2) & \text{in } 1D, \\ \min(s_I, s_1, s_2, s_3, s_4)/2 & \text{in } 2D, \\ \min(s_I, s_1, s_2, s_3, s_4, s_5, s_6)/3 & \text{in } 3D. \end{cases} \tag{16}$$

6.1 Adaptive time stepping

We note that an adaptive time step is possible since only the steady state of (14) is sought. Since the time step restriction is adapted for each cell, the reinitialization procedure is fast: small cells with a stringent time step restriction are located near the interface and therefore only a few iterations are required to reach the steady state at those cells (characteristic information flow away *from* the interface); cells far away from the interface are large and therefore do not require a small time step restriction. For example, consider the example depicted in figure 5, for which the level set function is defined initially as -1 inside a square domain (not aligned with the grid cells) and +1 outside. This initial level set function is therefore very far from the signed distance function that we seek to define. However, on a grid where the smallest grid has size $dx = 1/2048$, the reinitialization procedure takes only 35 iterations to fully converge to the signed distance function in the entire domain. In practice, the initial level set is never that far to the signed distance function and therefore only about 5 iterations are required regardless of the resolution of the finest level. Figure 6 illustrates the difference in the number of iterations required between uniform time stepping and adaptive time stepping. In the case of a uniform time step, we take $\triangle t = \triangle x_{\text{smallest}}/2$, with $\triangle x_{\text{smallest}}$ the size of the smallest cell.

6.2 Third order accuracy

We also computed the convergence rates of the reinitialization algorithm for the test problem proposed in [51]: Consider the level set function initially defined as:

$$\phi^0(x,y) = \left(0.1 + (x-1)^2 + (y-1)^2\right)\left(\sqrt{x^2+y^2} - 1\right),$$

which defines the interface as a circle with center the origin and radius 1. In this case, ϕ^0 is not a signed distance function and its gradients vary widely. Figure 7 illustrates the gradual deformation of the cross-sections of ϕ^0 as it evolves to the signed distance function. Table 1 illustrates that the method is third order accurate in the L^1 and L^∞ norms near the interface, where we use the standard formulas for the L^1 and L^∞ norms:

$$||\phi||_\infty = \max_{v:|\phi(v)|<1.2\Delta x} |\phi(v) - \phi_{\text{exact}}(v)|,$$
$$||\phi||_1 = \text{average}_{v:|\phi(v)|<1.2\Delta x} |\phi(v) - \phi_{\text{exact}}(v)|,$$

where $\Delta x = \Delta x_{\text{smallest}}$. Note that after the reinitializing the level set function, the choice of $v : |\phi(v)| < 1.2\Delta x$ ensures the selection of all the nodes adjacent to the interface.

In the entire domain, the method is second order accurate if we keep refining all the cells. In the practical case where only cells near the interface are refined, the accuracy in regions far away from the interface is meaningless. In the case where the interface presents sharp corners, the accuracy is reduced to first order in the L^∞ norm.

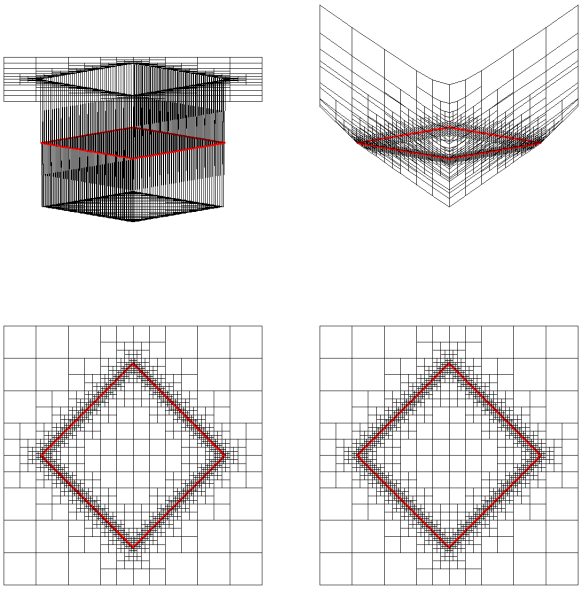

Figure 5: Reinitialization procedure. Left: Initial level set function (top) and its zero cross section (bottom) defining a square domain. Right: Reinitialized level set function (top) and its zero cross section (bottom). In particular, the difference in the zero level set between the initial and final stages is negligible. In this example, the level difference between adjacent cells is not restricted.

7 Motion Under an Externally Generated Velocity Field

7.1 Second Order Accurate Semi-Lagrangian Method

In the case where the velocity field is externally generated, the level set equation (1) is linear. In this case, one can use semi-Lagrangian methods. Semi-Lagrangian schemes are extensions of the Courant-Isaacson-Rees [8] method for hyperbolic equations and are unconditionally stable thus avoiding standard CFL condition of $\Delta t \approx \Delta x_{\text{smallest}}$. The general idea behind semi-Lagrangian methods is to reconstruct the solution by integrating numerically the equation along characteristic curves, starting from any grid point x_i and tracing back the departure point x_d in the upwind direction. Interpolation formulas are then used to recover the value of the solution at such points. In this work, we use a second order accurate Semi-Lagrangian method.

Consider the linear advection equation:

$$\phi_t + U \cdot \nabla \phi = 0, \tag{17}$$

where U is an externally generated velocity field. Then $\phi^{n+1}(x^{n+1}) = \phi^n(x_d)$, where x^{n+1} is any grid node and x_d is the corresponding departure point from which the characteristic curve originates. In this work, we use the second order mid-point method for locating the departure point, as in [65]:

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

where we define the velocity at the mid-time step $t^{n+\frac{1}{2}}$ by a linear combination of the velocities at the two previous time steps, i.e. $U^{n+\frac{1}{2}} = \frac{3}{2}U^n - \frac{1}{2}U^{n-1}$. Since $\hat{x}$ and x_d are not on grid nodes in general, interpolation procedures must be applied to define $U^{n+\frac{1}{2}}(\hat{x})$ and $\phi^n(x_d)$. We note that it is enough to define $U^{n+\frac{1}{2}}(\hat{x})$ with a multilinear interpolation (11) and $\phi^n(x_d)$ with the quadratic interpolation described by equations (12) and (13): Since a distance function has discontinuities in its derivative in general, the stabilized quadratic interpolation is preferred to the Hermite quadratic interpolation.

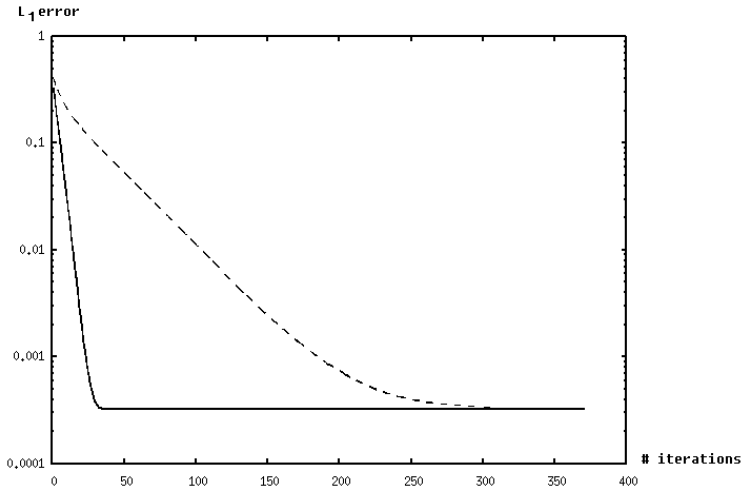

Figure 6: L_1 errors of the reinitialization algorithm in the case of the adaptive time step (solid line) and the uniform time step (dotted line).

			Finest Resolution				
			128^2	rate	256^2	rate	512^2
Uniform Refinement	Near Interface	L_1	4.36×10^{-6}	2.92	5.77×10^{-7}	3.02	7.12×10^{-8}
		L_∞	2.16×10^{-5}	2.74	3.24×10^{-6}	3.26	3.38×10^{-7}
	Whole Domain	L_1	3.27×10^{-4}	2.14	7.42×10^{-5}	2.11	1.71×10^{-5}
		L_∞	4.20×10^{-2}	1.56	1.43×10^{-2}	1.87	3.89×10^{-3}
Adaptive Refinement	Near Interface	L_1	4.36×10^{-6}	2.94	5.70×10^{-7}	3.00	7.14×10^{-8}
		L_∞	2.16×10^{-5}	2.87	2.96×10^{-6}	3.09	3.48×10^{-7}
	Whole Domain	L_1	3.27×10^{-4}	1.06	1.57×10^{-4}	1.01	7.82×10^{-5}
		L_∞	4.20×10^{-2}	0.00	4.20×10^{-2}	0.00	4.20×10^{-3}

Table 1: Convergence rates for the reinitialization for example 6.2. The initial grid is shown in figure 7. The condition for a node v_i to be '*near interface*' is chosen as $|\phi(v_i)| < \sqrt{2}\triangle x_{\text{smallest}}$, where $\triangle x_{\text{smallest}}$ is the size of the smallest cell. The '*whole domain*' excludes the region near the kink located at the origin, where accuracy drops to first order.

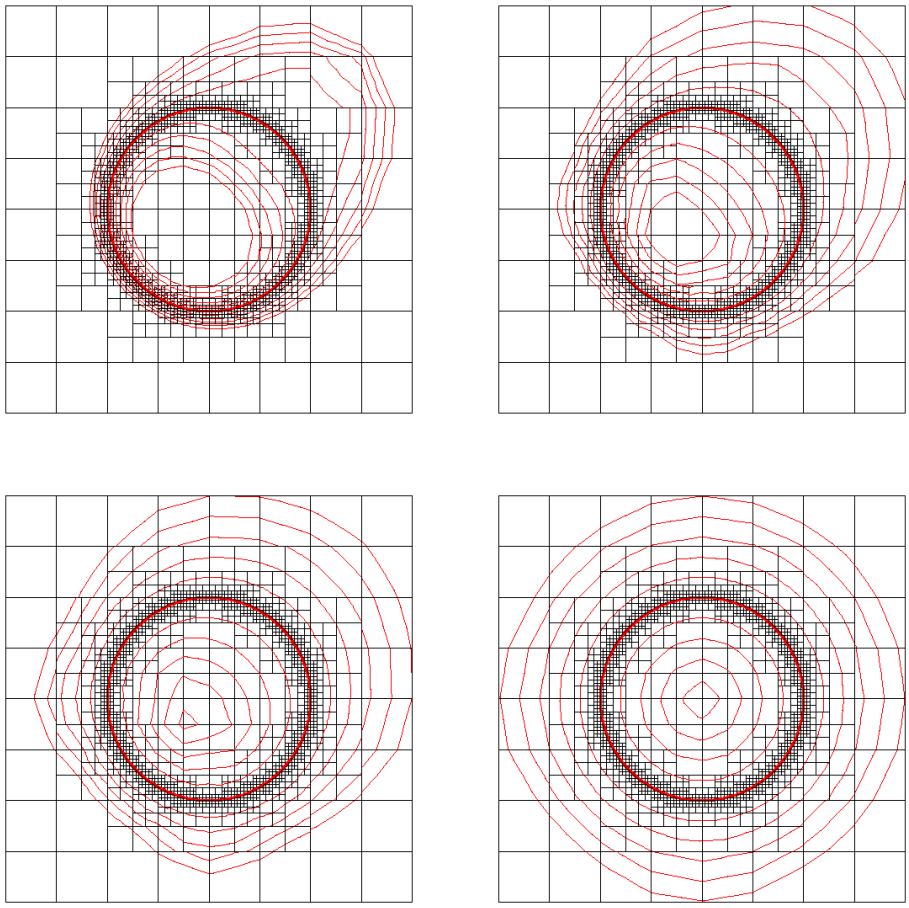

Figure 7: From top-left to bottom-right: Contours of the reinitialized level set function of example 6.2 after 0, 5, 10 and 20 iterations. The contours are evenly plotted from -1 to 1 with a thick line representing the zero contour.

Finest Resolution	L^∞ error of ϕ	rate	$L^1 error$ of ϕ	rate	Loss of volume(%)	rate
32^2	7.24×10^{-2}		3.11×10^{-2}		28.51	
64^2	1.78×10^{-2}	2.02	8.86×10^{-3}	1.81	7.21	1.98
128^2	4.52×10^{-3}	1.97	2.13×10^{-4}	2.05	1.78	2.01
256^2	1.13×10^{-3}	1.99	5.56×10^{-4}	1.93	0.45	1.98
512^2	2.85×10^{-4}	2.00	1.38×10^{-4}	2.01	0.11	2.03
1024^2	7.14×10^{-5}	2.00	3.46×10^{-5}	2.00	0.03	1.87
2048^2	1.78×10^{-5}	2.00	8.64×10^{-6}	2.00	0.007	2.01

Table 2: Convergence rates for example 7.2

	Finest Resolution						
	64^2	rate	128^2	rate	256^2	rate	512^2
L_1 error of ϕ	9.58×10^{-3}	2.80	1.38×10^{-3}	2.02	3.41×10^{-4}	2.08	8.09×10^{-5}
L_∞ error of ϕ	1.83×10^{-2}	1.57	6.17×10^{-3}	1.08	2.91×10^{-3}	1.39	1.11×10^{-3}
Volume Loss	4.48	2.36	0.874	1.40	0.331	1.79	0.0954
Max number of nodes	1045	1.10	2243	1.10	4815	1.09	10256
Min number of nodes	439	1.07	924	0.99	1831	1.01	3679
time(sec)	1.420	2.29	6.96	2.17	31.4	2.13	138
minimum memory(MB)	0.0448	1.07	0.0943	.97	0.185	1.00	0.371
maximum memory(MB)	0.101	1.30	0.248	1.06	0.517	1.03	1.06

Table 3: Convergence rates for example 7.3. The memory requirement increases linearly with effective resolution since most of the computational resources is focused near the one-dimensional interface, i.e. our method is an efficient implementation of local level set methods. The computational time increases quadratically with effective resolution, since the number of nodes is doubled and the time step is halved.

7.2 Test: Rotation in 2D

Consider a domain $\Omega = [-1, 1]^2$ and a disk of radius $R = .15$ and center initially at $(0, .75)$, rotating under the divergence free velocity field

$$\begin{aligned} u(x, y) &= -y \\ v(x, y) &= x \end{aligned}$$

The final time $t = 2\pi$ is the time when the rotation completes one revolution. In the simulation, the adaptive refinement is used, and the time step restriction is $\Delta t = 5\Delta x$. Table 2 demonstrates second order accuracy for the level set as well as for the mass conservation. We note that we only consider the grid nodes neighboring the interface in our computation of the accuracy for the level set function ϕ since only those points define the location of the interface.

7.3 Test: Vortex in 2D

In this example, we test our level set implementation on the more challenging flow proposed by Bell *et al.* [2]: Consider a domain $\Omega = [0, 1]^2$ and a disk of radius .15 and center $(.5, .75)$ as the initial zero level set contour. The level set is then deformed under the divergence free velocity field $U = (u, v)$ given by:

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

The disk is deformed forward until $t = 1$ and then backward to the original shape using the reverse velocity field with a time step restriction of $\triangle t = 5 \cdot \triangle x_{\text{smallest}}$.

Table 3 demonstrate second order accuracy for L_1 error of ϕ and volume of loss, and linear increase in the maximum/minimum number of nodes. Note that the uniform grid of resolution 512^2 requires about 25 times more nodes

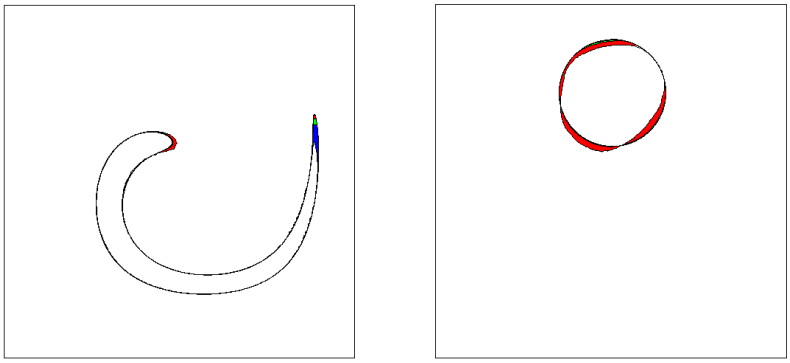

Figure 8: Contours of the zero level sets for example 7.3 with effective resolutions of 64^2, 128^2, 256^2 and 512^2 at $t = 1$ (left) and $t = 2$ (right). The colors red, green, blue represent the difference in the interface location between the resolutions 64^2 and 128^2, 128^2 and 256^2, 256^2 and 512^2, respectively.

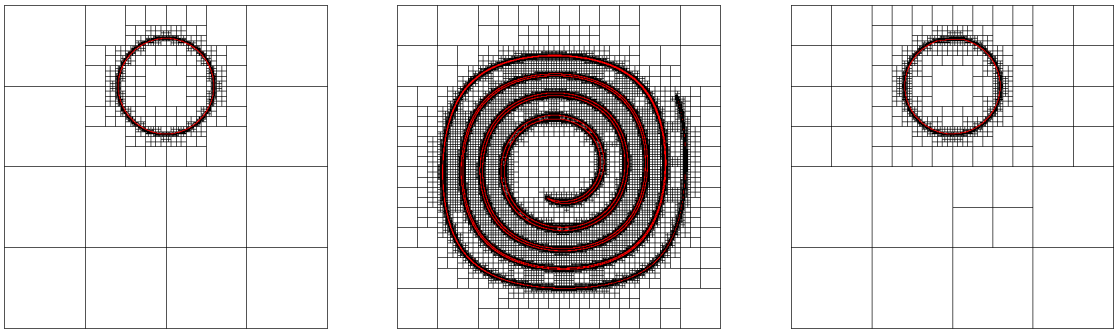

Figure 9: Level set evolution at $t = 0$ (left), $t = 3$ (center) and $t = 6$ (right). The effective resolution is 2048^2 and the mass is conserved within .3%

than the adaptive grid with the same resolution. Although second order accuracy was achieved in both the maximum and average norms in the previous example, the convergence rate of the maximum error is oscillating between one and two. This is due to the fact that as the interface deforms, some part of the interface are under resolved. Figure 8 illustrates this at $t = 1$: Here, the tail of the interface is not resolved accurately. This deterioration in accuracy was also reported in [45].

Figure 9 illustrates the evolution of the interface location initially (left), at $t = 6$ (center) and when the interface is fully rewinded (right). This example illustrates the ability of the present method to accurately capture the evolution of an interface undergoing large deformations and the ability to preserve mass effectively (mass loss $\approx .3\%$).

7.4 Test: Rotation in 3D

Consider a domain $\Omega = [-2, 2]^3$ and a sphere of radius $R = .5$ and center initially at $(0, 1, 0)$, rotating under the divergence free velocity field

$$\begin{aligned} u(x, y, z) &= -y \\ v(x, y, z) &= x \\ w(x, y, z) &= 0 \end{aligned}$$

	Finest Resolution						
	32^3	rate	64^3	rate	128^3	rate	256^3
L_1 error of ϕ	6.86×10^{-2}	1.88	1.87×10^{-2}	1.99	4.70×10^{-3}	1.99	1.18×10^{-3}
L_∞ error of ϕ	1.76×10^{-1}	2.02	4.35×10^{-2}	2.02	1.07×10^{-2}	2.02	2.65×10^{-3}
Volume loss(%)	23.1	2.16	5.14	2.12	1.18	2.07	0.282

Table 4: Convergence rates for the interface's location for example 7.4.

Finest Resolution	Time	Rate	Min # nodes	Rate	Max # nodes	Rate	Min Memory	Rate	Max Memory	Rate
128^3	237.5		44943		133308		4.54		13.7	
256^3	2214	3.23	173637	1.95	598264	2.17	17.6	1.96	61.5	2.17
512^3	19521	3.14	685220	1.98	2606710	2.12	69.6	1.98	268	2.12

Table 5: Memory and CPU requirements for example 7.5. The memory requirement increases quadratically with effective resolution since most of the computational resources is focused near the two-dimensional interface, i.e. our method is an efficient implementation of local level set methods. The computational time increases cubically with effective resolution, since the number of nodes is multiplied by four and the time step is halved.

The simulation is run until $t = 2\pi$, when the rotation completes one revolution. In the simulation, the adaptive refinement is used, and the time step restriction is $\Delta t = 6\Delta x_{\text{smallest}}$. Table 4 demonstrates second order accuracy for the level set as well as for the mass conservation. We note that we only consider the grid nodes neighboring the interface in our computation of the accuracy for the level set function ϕ. Figure 10 shows the adaptive grid for the rotating sphere.

7.5 Enright's Test in 3D

We consider the test proposed in Enright *et al.* [13]: A sphere of center $(0.35, 0.35, 0.35)$ and radius 0.15 in the domain of $[0, 1]^3$ is deformed under the following divergence free velocity field:

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

forward in time and then backward to its original shape with the reversed velocity. Figure 11 illustrates the interface motion with a time step restriction of $\triangle t = 5 \cdot \triangle x_{\text{smallest}}$. We note that, in the simulation with an effective resolution of 512^3, minimum the number of nodes used was 685220 and the maximum was 2606710. In contrast, the number of nodes in the case of a uniform grid with the same resolution, the number of nodes would be about 50 times larger. The volume loss is 3.21% for an effective resolution of 256^3 and 0.739% for an effective resolution of 512^3. Figure 12 compares the interface evolution with an effective resolution of 128^3, 256^3 and 512^3. Table 5 describes the memory and CPU requirements, and table 6 describes the volume loss and the accuracy of the interface location after reconvering the original shape.

The Enright's test is a canonical example to test the amount of numerical dissipation of level set methods. The particle level set method reported 2.6% volume loss on a 100^3 uniform grid together with Lagrangian particles [13]. Our results show that we obtain a loss of mass of $.74\%$ in the case of a 512^3 effective resolution, which corresponds to a 137^3 uniform grid in term of number of nodes. The particle level set was further improved in [14] using octree data structures in addition to particles. Although [14] does not report any quantitative results, we find that our result for the Enright's test is visually comparable to that obtained in [14] for the same effective resolution and compares favorably with the results in [26].

We note that the jump in rate in table 6 can be explained by the lack of resolution for describing the developing thin film. This is related to the Nyquist-Shannon sampling theorem that states that in order to fully reconstruct a signal the sampling frequency should be at least twice the signal bandwidth. In our case, the fact that the thin film

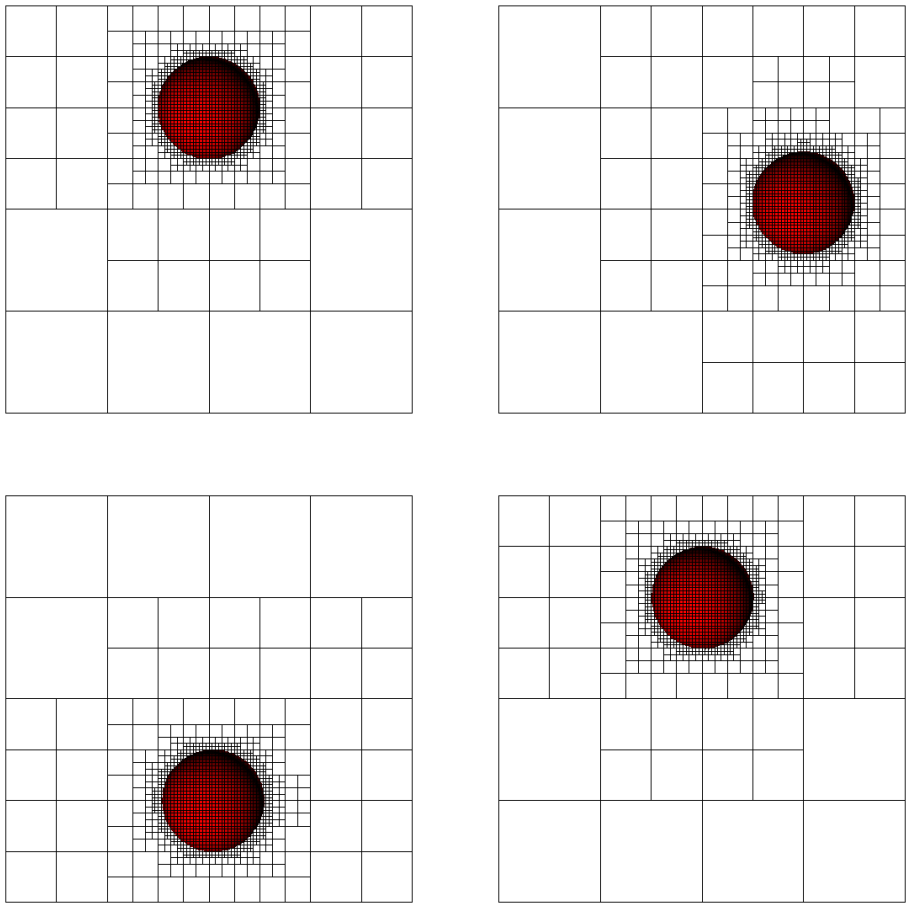

Figure 10: Evolution of the interface for example 7.4: Initial data (top-left), interface after a quarter turn (top-right), interface after a half turn (bottom-left) and final location (bottom-right). The finest resolution is 128^3.

Finest Resolution	Volume Loss	Rate	L^1 error of ϕ	Rate	L^∞ error of ϕ	Rate
128^3	16.02%		1.96×10^{-2}		1.54×10^{-1}	
256^3	3.21%	2.32	2.83×10^{-3}	2.79	1.06×10^{-1}	.88
512^3	.739 %	2.12	4.38×10^{-4}	2.69	5.74×10^{-3}	7.52

Table 6: Convergence rates for the interface's location for example 7.5.

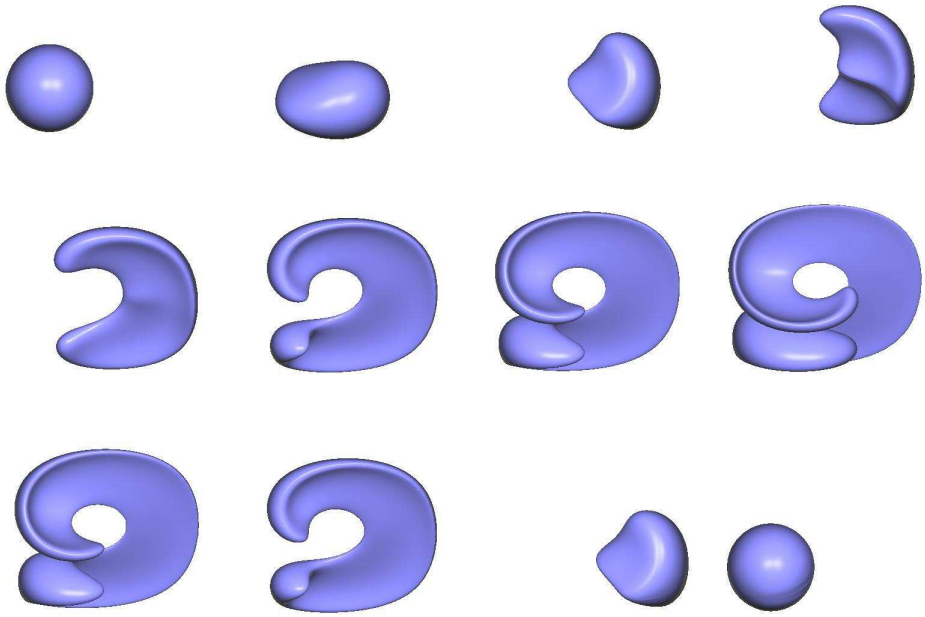

Figure 11: Evolution of the interface for the Enright's test with finest resolution of 512^3.

is under-resolved prevents subcell resolution of the reinitialization scheme. For higher resolutions, we would expect second-order accuracy.

8 Motion in the Normal Direction and Curvature Driven Flow

The equation describing an interface propagating in its normal direction and under its mean curvature is given by [48]:

$$\phi_t + (\alpha - \beta\kappa)|\nabla\phi| = 0, \tag{18}$$

where κ is the mean curvature of the interface $\kappa = \nabla\cdot(\nabla\phi/|\nabla\phi|)$. The coefficients α and $\beta \geq 0$ control the magnitude of the speed in the normal direction and the strength of the curvature dependence, respectively. The case where $\beta < 0$ is ill-posed and therefore we do not consider it here.

8.1 Motion in the Normal Direction

First, we discuss the case when $\beta = 0$. Using the second order one-sided derivatives described in section 4 and discretizing the Hamiltonian using a Godunov scheme, we semi-discretize the equation as :

$$\frac{d\phi}{dt} + \alpha \cdot H_G(\phi) = 0$$

where the Godunov Hamiltonian H_G is defined as:

$$H_G(\phi) = \begin{cases} \sqrt{\max(|(D_x^+\phi)^-|^2, |(D_x^-\phi)^+|^2) + \max(|(D_y^+\phi)^-|^2, |(D_y^-\phi)^+|^2)} & \text{if } \alpha > 0 \\ \sqrt{\max(|(D_x^+\phi)^+|^2, |(D_x^-\phi)^-|^2) + \max(|(D_y^+\phi)^+|^2, |(D_y^-\phi)^-|^2)} & \text{otherwise} \end{cases}$$

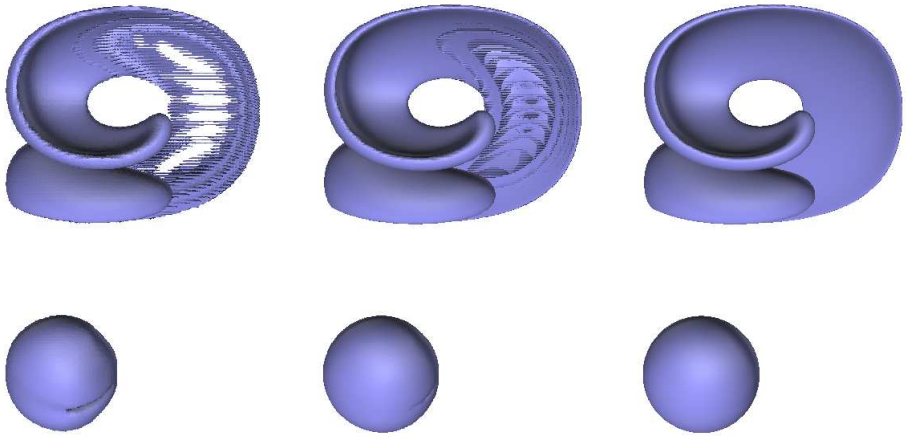

Figure 12: Effect of refinement on the Enright's test: The top figures correspond to the interface fully stretched and the bottom figures correspond to the interface rewinded to the original sphere. The finest resolutions are 128^3 (left), 256^3 (center) and 512^3 (right).

	Finest Resolution						
	64^2	rate	128^2	rate	256^2	rate	512^2
L_1 error of ϕ	1.46×10^{-3}	2.02	3.58×10^{-4}	2.00	8.56×10^{-4}	1.98	2.26×10^{-5}
L_∞ error of ϕ	2.77×10^{-3}	2.04	6.72×10^{-4}	1.95	1.73×10^{-4}	1.99	4.36×10^{-5}

Table 7: Convergence rate for a circle shrinking with unit normal velocity. consider a domain of $[-2, 2]^2$ and an interface initially described by a circle centered at the origin with radius $R = 1$. The interface is evolved until $t = 0.5$.

This equation is discretized in time using the second order TVD Runge-Kutta method (see [56, 34]):

$$\frac{\tilde{\phi}^{n+1} - \phi^n}{\triangle t} + \alpha \cdot H_G\left(\phi^n\right) = 0 \tag{19}$$

$$\frac{\tilde{\phi}^{n+2} - \tilde{\phi}^n}{\triangle t} + \alpha \cdot H_G\left(\tilde{\phi}^{n+1}\right) = 0 \tag{20}$$

$$\phi^{n+1} = \frac{\phi^n + \tilde{\phi}^{n+2}}{2} \tag{21}$$

Table 7 illustrates that the method described above is second order accurate in both the maximum and the average norms for smooth data. In the case where the interface presents sharp corners, figure 13 illustrates that the method converges to the correct viscosity solution [48].

8.2 Adding Motion by Mean Curvature

Now we discuss the case when $\beta > 0$. The curvature term can be discretized explicitly or implicitly. In the case where the curvature term is discretized explicitly, the corresponding time step restriction of $\Delta t \approx \Delta x^2$ is too stringent to be practical since it would be constrained by the size of the smallest grid cell in the grid. In [57], Smereka proposed an implicit discretization of the curvature term in the case of uniform grids: Using the following operator splitting:

$$\kappa|\nabla\phi| = \Delta\phi - \frac{\nabla\phi}{|\nabla\phi|}\nabla(|\nabla\phi|),$$

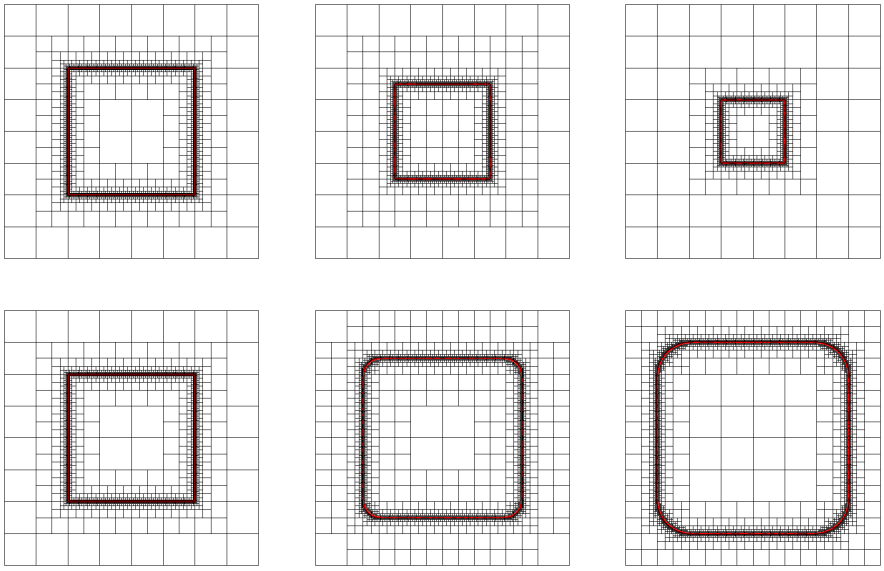

Figure 13: Shrinking square in the first row, and expanding square in the second row

	Finest Resolution						
	128^2	rate	256^2	rate	512^2	rate	1024^2
L_1 error of ϕ	5.22×10^{-3}	1.00	2.60×10^{-3}	0.96	1.33×10^{-3}	0.95	6.95×10^{-4}
L_∞ error of ϕ	5.47×10^{-3}	0.93	2.86×10^{-3}	0.86	1.56×10^{-3}	0.81	8.91×10^{-4}

Table 8: Convergence rate for a circle with curvature dependent speed of $\alpha = 1.5$ and $\beta = 1$. Initially circle is centered at $(0, 0)$ with radius one in a domain of $[-2, 2]^2$. Test was run until 0.5.The radius $r(t)$ of the circle satisfies $r' = \alpha - \frac{\beta}{r}$ with $r(0) = 1$. $r(0.5)$ is approximated as 1.3108122 from the ordinary differential equation within error bound of 10^{-7}.

equation (18) is discretized as:

$$\frac{\phi^{n+1} - \phi^n}{\triangle t} + \alpha H_G(\phi^n) = \beta \Delta \phi^{n+1} - \beta \frac{\nabla \phi^n}{|\nabla \phi^n|} \cdot \nabla(|\nabla \phi^n|).$$

In this work, we used a Backward Euler step to treat the linear term, and a Forward Euler step for the nonlinear term. The derivatives Δ and ∇ are discretized by the central finite differences described in section 4. Discretizing implicitly the Laplacian requires a linear system that we solve using the supra convergent method presented in Min, Gibou and Ceniceros [38]. As noted in [57], the semi-implicit discretization on the curvature term allows for a big time step, so that the time step restriction is that of the convection part, i.e.

$$\Delta t = \frac{\Delta x_{\text{smallest}}}{\alpha \cdot \#\text{ dimensions}}, \tag{22}$$

where # dimensions is the number of dimensions.

Table 8 demonstrates that the method is first order accurate in the average norm for smooth a interface. The deterioration in the maximum norm probably comes from the Elliptic part of the solver, which propagates the errors from the regions where the grid cells are coarse and unrefined to the regions where the grid cells are refined. Figure 14 illustrates the motion of an interface under mean curvature for the example presented in [57].

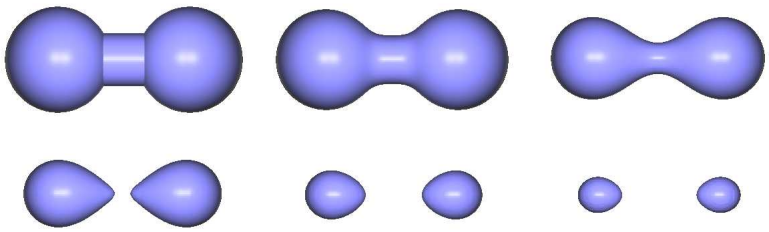

Figure 14: Motion with curvature flow for a barbel shape. $\alpha = 0, \beta = 1$ in 256^3 resolution. From top-left to bottom-right, the times are 0, 0.023, 0.093, 0.140, 0.304 and 0.323. The CFL condition is $\Delta t = \Delta x_{\text{smallest}}$.

9 Adaptive Grid Generation

As the interface deforms some provisions must be given to refine the grid near the interface while coarsening in regions farther away. The grid is constructed in such a way that the smallest grid cells lie on the interface as described in section 3. This construction depends on an input function, $\tilde{\phi}^{n+1} : \mathbb{R}^n \to \mathbb{R}$ that is close to the signed distance function at any point in space. This function can be constructed in two different ways: (1) In the case where semi-Lagrangian methods are used, a function $\tilde{\phi}^{n+1}$ can be defined as $\tilde{\phi}^{n+1} = \phi^n(x_d)$, where x_d is found by tracing back the characteristic curves and where $\phi^n(x_d)$ is interpolated from the node values of ϕ^n as described in section 7. (2) In the case where the velocity field is nonlinear, semi-Lagrangian methods cannot be used. In this case the level set function is first evolved from ϕ^n to ϕ^{n+1} on the same grid G^n. Then ϕ^{n+1} is reinitialized into a signed distance function using the algorithm described in section 6. Now at every point in space, we can define $\tilde{\phi}^{n+1} : \mathbb{R}^n \to \mathbb{R}$ by interpolation of ϕ^{n+1}. Once the function $\tilde{\phi}^{n+1} : \mathbb{R}^n \to \mathbb{R}$ can be define anywhere in space, the new grid G^{n+1} is generated by simply splitting a cell if the Lipschitz condition:

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

is satisfied. In practice, instead of generating G^{n+1} from the root cell, we start from G^n and apply the procedure detailed in Algorithm 1, i.e. starting the recursion from the root cell of G^{n+1}, the cell is recursively split if the refinement criteria is satisfied, otherwise all of its children are merged.

Algorithm 1 : 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}

10 High Order Extrapolation in the Normal Direction

The ghost fluid method, introduced by Fedkiw *et al.* [17], is a technique for imposing boundary conditions at the interface in a level set framework and has been successfully applied to a wide range of applications (see e.g. [18, 16,

15, 42, 43, 6, 35, 22, 21, 20, 19] and the references therein). One basic component of this method is the extrapolation of some scalar quantities in the normal direction. In some cases (see e.g. [20]), high order extrapolations in the normal direction are needed. This can be performed in a series of steps, as proposed in Aslam [1]. For example, suppose that we seek to extrapolate u quadratically from the region where $\phi \leq 0$ to the region where $\phi > 0$. We first compute $u_{nn} = \vec{n} \cdot \nabla (\vec{n} \cdot \nabla u)$ in the region where $\phi \leq 0$ and extrapolate (constant extrapolation) this quantity across the interface by solving the following partial differential equation:

$$\frac{\partial u_{nn}}{\partial \tau} + H(\phi, u_{nn})(\vec{n} \cdot \nabla u_{nn}) = 0,$$

where $H(\phi, u_{nn})$ is the Heaviside function defined below and τ is the fictitious time step. Then we define u_n in the region where $\phi > 0$ in such a way its normal derivative is u_{nn}. This can be accomplished by solving the following PDE:

$$\frac{\partial u_n}{\partial \tau} + H(\phi, u_n)(\vec{n} \cdot \nabla u_n - u_{nn}) = 0.$$

Finally we can define u in such a way its normal derivative is u_n by solving:

$$\frac{\partial u}{\partial \tau} + H(\phi, u)(\vec{n} \cdot \nabla u - u_n) = 0.$$

Numerically the Heaviside function $H(\phi, S)(v_i)$ associated with a quantity S at the node v_i is set to zero if the nodes involved in the computation of S are all in the region where $\phi < 0$. Otherwise, it is set to 1. Therefore we define the Heaviside functions $H(\phi, u)$, $H(\phi, u_n)$ and $H(\phi, u_{nn})$ as follows:

$$H(\phi, u)(v_i) = \begin{cases} 0 & \text{if } \phi(v_i) < 0 \\ 1 & \text{otherwise} \end{cases},$$

$$H(\phi, u_n)(v_i) = \begin{cases} 0 & \text{if } H(\phi, u)(v_j) = 0 \text{ for all } v_j \in \text{ngbd}(v_i) \\ 1 & \text{otherwise} \end{cases},$$

$$H(\phi, u_{nn})(v_i) = \begin{cases} 0 & \text{if } H(\phi, u_n)(v_j) = 0 \text{ for all } v_j \in \text{ngbd}(v_i) \\ 1 & \text{otherwise} \end{cases},$$

where $\text{ngbd}(v_i)$ denotes the set of direct neighboring nodes of v_i. The quantity $u_n = \vec{n} \cdot \nabla u$ is computed by the central finite differences described in section 4 for all the nodes where $H(\phi, u_n) = 0$. Likewise, using the values of u_{nn} is then computed by central differencing for all the nodes where $H(\phi, u_{nn}) = 0$. The three partial differential equations above are discretized in a dimension by dimension framework using the upwind schemes and the one-sided finite differences of section 4, i.e. the discretizations in a semi-discrete form read:

$$\frac{d}{d\tau} u_{nn} + H(\phi, u_{nn}) \left(n_x^+ D_x^- u_{nn} + n_x^- D_x^+ u_{nn} \right) = 0,$$

$$\frac{d}{d\tau} u_n + H(\phi, u_n) \left(n_x^+ D_x^- u_n + n_x^- D_x^+ u_n \right) = H(\phi, u_n) u_{nn},$$

and

$$\frac{d}{d\tau} u + H(\phi, u) \left(n_x^+ D_x^- u + n_x^- D_x^+ u \right) = H(\phi, u) u_n.$$

These semi-discrete equations are then evolved in time using the same TVD RK-2 method of section 6. Since the equations are evolved in fictitious time, we can take the same time step restriction as in the reinitialization procedure of section 6.

Figure 15 illustrates the constant, linear and quadratic extrapolation obtained with the algorithm described above: Consider a computational domain $\Omega = (-\pi, \pi) \times (-\pi, \pi)$ separated into two regions: Ω^- defined as the interior of a disk with center at the origin and radius two, and $\Omega^+ = \Omega \setminus \Omega^-$. The function u to be extrapolated from Ω^- to Ω^+ is defined as $u = \cos(x) \sin(y)$ for $x \in \Omega^-$. We have extrapolated u in the entire region in this example for the sake of

Finest Resolution	L^∞ error of ϕ	rate	$L^1 error$ of ϕ	rate
32^2	5.11×10^{-1}		1.11×10^{-1}	
64^2	2.55×10^{-1}	1.01	3.75×10^{-2}	1.56
128^2	1.25×10^{-1}	1.02	1.06×10^{-2}	1.82
256^2	6.20×10^{-2}	1.01	2.89×10^{-3}	1.87
512^2	3.14×10^{-2}	.97	7.59×10^{-4}	1.92
1024^2	1.59×10^{-2}	.98	2.01×10^{-4}	1.91

Table 9: Convergence rate for the constant extrapolation

Effective Resolution	L^∞ error of ϕ	rate	$L^1 error$ of ϕ	rate
32^2	2.02×10^{-1}		3.52×10^{-2}	
64^2	7.21×10^{-2}	1.48	6.09×10^{-3}	2.53
128^2	1.78×10^{-2}	2.00	8.79×10^{-4}	2.79
256^2	5.27×10^{-3}	1.76	1.19×10^{-4}	2.88
512^2	1.12×10^{-3}	2.22	1.54×10^{-5}	2.94
1024^2	2.83×10^{-4}	1.99	2.04×10^{-6}	2.91

Table 10: Convergence rate for the linear extrapolation

presentation but we emphasize that in practice the extrapolation is performed only in a neighborhood of the interface. Tables 9, 10 and 11 demonstrate the first order accuracy for the constant extrapolation, the second order accuracy for the linear extrapolation and the third order accuracy for the quadratic extrapolation. We note that it is enough to discretize $D_x^{\pm} u_{nn}$ and $D_x^{\pm} u_n$ with the first order accurate finite difference of section 4, and $D_x^{\pm} u$ with the second order accurate finite difference in section 4 to achieve third order accuracy in u in the case of a quadratic extrapolation. The same accuracy would be achieved in the case where the second order accurate finite differences were used for $D_x^{\pm} u_{nn}$, $D_x^{\pm} u_n$, and $D_x^{\pm} u_n$. However, using the first order accurate finite difference schemes for $D_x^{\pm} u_{nn}$ and $D_x^{\pm} u_n$ yields more robust results since u_{nn} and u_n may be noisy unless u is a very smooth function.

11 Conclusion

We have presented a level set method on non-graded adaptive Cartesian grids, i.e. grids for which the ratio between adjacent cells is not constrained. We use quadtree and octree data structures to represent the grid and a simple algorithm to generate a mesh with the finest resolution at the interface. We have presented (1) a locally third order accurate reinitialization scheme that transforms an arbitrary level set function into a signed distance function, (2) a second order accurate semi-Lagrangian methods to evolve the linear level set advection equation under an externally generated

Effective Resolution	L^∞ error of ϕ	rate	$L^1 error$ of ϕ	rate
32^2	1.62×10^{-1}		2.26×10^{-2}	
64^2	2.31×10^{-2}	2.82	2.21×10^{-3}	3.36
128^2	2.95×10^{-3}	2.96	1.65×10^{-4}	3.73
256^2	3.81×10^{-4}	2.95	1.17×10^{-5}	3.81
512^2	4.89×10^{-5}	2.96	7.82×10^{-7}	3.91
1024^2	6.19×10^{-6}	2.98	5.31×10^{-8}	3.88

Table 11: Convergence rate for the quadratic extrapolation

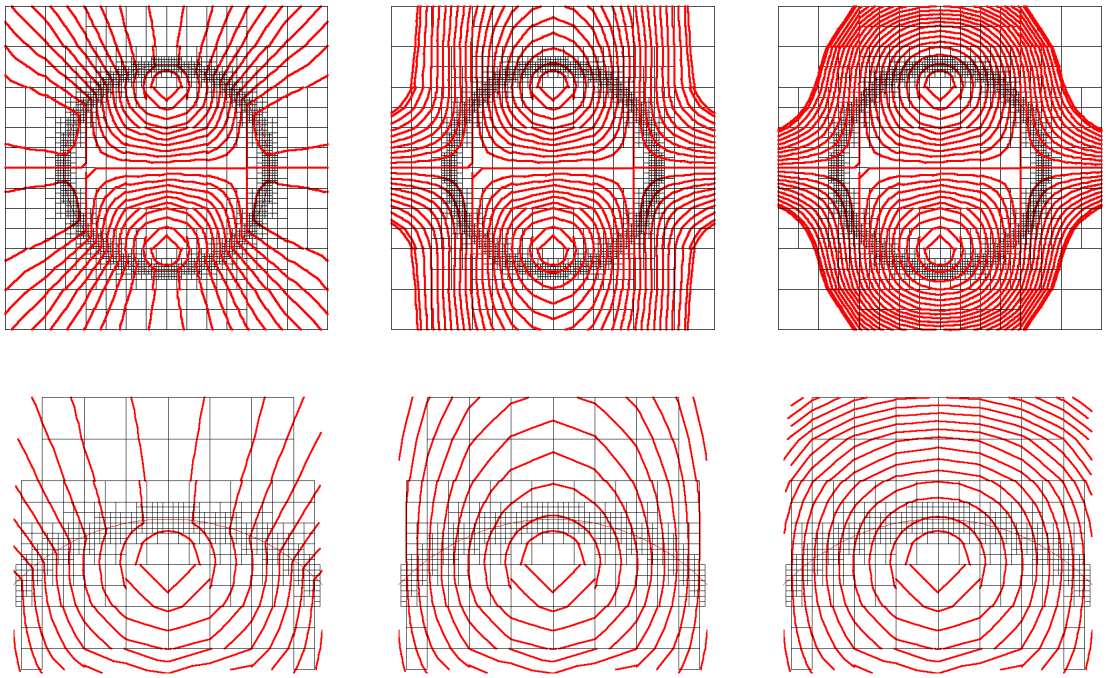

Figure 15: Contours of the solution after it has been extrapolated across the interface with a constant (left), linear (center) and quadratic (right) extrapolations across an interface. The top row illustrates the extrapolation on the entire domain and the bottom row is a zoom near the interface. The exact solution is given inside the circle centered at the origin and with radius 2 and is extrapolated outside in the normal direction. We then plot the level curves of the solution.

velocity field, (3) a second order accurate upwind method to evolve the nonlinear level set equation under a normal velocity as well as to extrapolate scalar quantities across an interface in the normal direction, and (4) a semi-implicit scheme to evolve the interface under mean curvature. This method produces results with a negligible amount of mass loss. We have proposed numerical examples in two and three spatial dimensions to demonstrate the accuracy of the method.

12 Acknowledgment

The research of F. Gibou was supported in part by the Alfred P. Sloan Foundation through a research fellowship in Mathematics.

References

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

[2] J. B. Bell, P. Colella, and H. M. Glaz. A second order projection method for the incompressible Navier-Stokes equations. *J. Comput. Phys*, 85:257–283, 1989.

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

[4] D. Benson. Volume of fluid interface reconstruction methods for multimaterial problems. *Applied Mechanics Reviews*, 52:151–165, 2002.

[5] 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.

[6] 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.

[7] A. Chorin. A Numerical Method for Solving Incompressible Viscous Flow Problems. *J. Comput. Phys.*, 2:12–26, 1967.

[8] R. Courant, E. Isaacson, and M. Rees. On the solution of nonlinear hyperbolic differential equations by finite differences. *Comm. Pure Appl. Math.*, 5:243–255, 1952.

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

[10] Marc Droske, Bernhard Meyer, Martin Rumpf, and Carlo Schaller. An adaptive level set method for medical image segmentation. *Lecture Notes in Computer Science*, 2082:416–422, 2001.

[11] V. Dyadechko and M. Shashkov. Moment-of-fluid interface reconstruction. Technical report, Los Alamos National Laboratory (LA-UR-05-7571), 2006.

[12] K. Elder, M. Grant, N. Provatas, and J. Kosterlitz. Sharp interface limits of phase-field models. *SIAM J. Appl. Math*, 64:021604, 2001.

[13] 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.

[14] D. Enright, S. Marschner, and R. Fedkiw. Animation and rendering of complex water surfaces. *ACM Trans. Graph. (SIGGRAPH Proc.)*, 21(3):736–744, 2002.

[15] R. Fedkiw. *The Ghost Fluid Method for Discontinuities and Interfaces*. Godunov Methods, edited by E.F. Toro, New York, 2001.

[16] R. Fedkiw. Coupling an Eulerian fluid calculation to a Lagrangian solid calculation with the ghost fluid method. *J. Comput. Phys.*, 175:200–224, 2002.

[17] 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.

[18] R. Fedkiw, J. Stam, and H. Jensen. Visual simulation of smoke. In *Proc. of ACM SIGGRAPH 2001*, pages 15–22, 2001.

[19] F. Gibou, L. Chen, D. Nguyen, and S. Banerjee. A level set based sharp interface method for incompressible flows with phase change. *J. Comput. Phys. (In press)*.

[20] 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.

[21] F. Gibou, R. Fedkiw, R. Caflisch, and S. Osher. A level set approach for the numerical simulation of dendritic growth. *J. Sci. Comput.*, 19:183–199, 2003.

[22] 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.

[23] F. Gibou, C. Ratsch, and R. Caflisch. Capture numbers in rate equations and scaling laws for epitaxial growth. *Phys. Rev. B.*, 67:155403, 2003.

[24] F. Gibou, C. Ratsch, S. Chen, M. Gyure, and R. Caflisch. Rate equations and capture numbers with implicit island correlations. *Phys. Rev. B.*, 63:115401, 2001.

[25] J. Glimm, J. W. Grove, X. L. Li, and N.Zhao. Simple front tracking. *Contemporary Math.*, 238:133–149, 1999.

[26] S. Hieber and P. Koumoutsakos. A Lagrangian particle level set method. *J. Comput. Phys.*, 210:342–367, 2005.

[27] C. Hirt and B. Nichols. Volume of fluid (vof) method for the dynamics of free boundaries. *J. Comput. Phys.*, 39:201–225, 1981.

[28] D. Juric and G. Tryggvason. A front tracking method for dendritic solidification. *J. Comput. Phys*, 123:127–148, 1996.

[29] D. Juric and G. Tryggvason. Computations of boiling flows. *Int. J. Multiphase. Flow.*, 24:387–410, 1998.

[30] A. Karma. Phase-field formulation for quantitative modeling of alloy solidification. *Phys. Rev. Lett.*, 87:115701, 2001.

[31] 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.

[32] 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.

[33] J. S. Langer. *Directions in Condensed Matter Physics*. G. Grinstein and G Mazenko, World Scientific Singapore, 1986.

[34] X.-D. Liu, S. Osher, and T. Chan. Weighted essentially non-oscillatory schemes. *J. Comput. Phys.*, 126:202–212, 1996.

[35] 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.

[36] 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.

[37] B. Milne. *Adaptive Level Set Methods Interfaces*. PhD thesis, University of California at Berkeley, June 1995.

[38] C. Min, F. Gibou, and H. Ceniceros. A supra-convergent finite difference scheme for the variable coefficient Poisson equation on non-graded grids. *J. Comput. Phys.*, 218:123–140, 2006.

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

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

[41] 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.

[42] D. Nguyen, R. Fedkiw, and M. Kang. A boundary condition capturing method for incompressible flame discontinuities. *J. Comput. Phys.*, 172:71–98, 2001.

[43] 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.

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

[45] E. Olsson and G. Kreiss. A conservative level set method for two phase flow. *J. Comput. Phys.*, 210:225–246, 2005.

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

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

[48] S. Osher and J. Sethian. Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations. *J. Comput. Phys.*, 79:12–49, 1988.

[49] S. Popinet. Gerris: A tree-based adaptive solver for the incompressible euler equations in complex geometries. *J. Comput. Phys.*, 190:572–600, 2003.

[50] 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.

[51] G. Russo and P. Smereka. A remark on computing distance functions. *J. Comput. Phys.*, 163:51–67, 2000.

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

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

[54] A. Schmidt. Computation of three dimensional dendrites with finite elements. *J. Comput. Phys.*, 125:293–312, 1996.

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

[56] C.-W. Shu and S. Osher. Efficient implementation of essentially non-oscillatory shock capturing schemes. *J. Comput. Phys.*, 77:439–471, 1988.

[57] P. Smereka. Semi-implicit level set methods for curvature and surface diffusion motion. *J. Sci. Comput.*, 19:439 – 456, 2003.

[58] J. Strain. Tree methods for moving interfaces. *J. Comput. Phys.*, 151:616–648, 1999.

[59] J. Strain. A fast modular semi-Lagrangian method for moving interfaces. *J. Comput. Phys.*, 161:512–536, 2000.

[60] M. Sussman, E. Fatemi, P. Smereka, and S. Osher. An improved level set method for incompressible two-phase flows. *Computers and Fluids*, 27:663–680, 1998.

[61] M. Sussman, P. Smereka, and S. Osher. A level set approach for computing solutions to incompressible two-phase flow. *J. Comput. Phys.*, 114:146–159, 1994.

[62] 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.

[63] 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.

[64] A. Wheeler. *Hanbook of Crystal Growth*. D. T. Hurle, 1993.

[65] D. Xiu and G. Karniadakis. A semi-Lagrangian high-order method for Navier-Stokes equations. *J. Comput. Phys*, 172:658–684, 2001.

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