A Cell Based Particle Method for Modeling Dynamic Interfaces

Yu-Sing Hon* Shingyu Leung† Hongkai Zhao‡

April 8, 2014

Abstract

We propose several modifications to the grid based particle method (GBPM) [21] for moving interface modeling. There are several nice features of the proposed algorithm. The new method can significantly improve the distribution of sampling particles on the evolving interface. Unlike the original GBPM where footpoints (sampling points) tend to cluster to each other, the sampling points in the new method tend to be better separated on the interface. Moreover, by replacing the grid-based discretization using the cell-based discretization, we naturally decompose the interface into segments so that we can easily approximate surface integrals. As a possible alternative to the local polynomial least square approximation, we also study a geometric basis for local reconstruction in the resampling step. We will show that such modification can simplify the overall implementations. Numerical examples in two- and three-dimensions will show that the algorithm is computationally efficient and accurate.

1 Introduction

Interfacial motion can be found in many applications in science. It is therefore very important to develop an efficient and flexible yet simple numerical representation for various modelings. Roughly speaking, we can classify existing numerical methods for moving interface problems into two categories according to the way how the interface is numerically represented. The first category are tracking methods, in which the interface is *explicitly* represented by Lagrangian marker particles and its dynamics is tracked by the motion of these particles. For examples, boundary integral methods [14, 28], boundary element methods, front tracking method [41, 10, 37], and etc. belong to this type. The second class is capturing methods, in which the interface is *implicitly* embedded in a scalar field function defined on a fixed mesh, such as a Cartesian grid. The interface dynamics is captured by the evolution of the scalar function in an Eulerian framework. Main representatives include the level set methods [27], phase field method [5, 1], volume of fluid method [13], to name just a few. Recently, there are various interesting works to combine these Lagrangian and Eulerian approaches for moving interface problems, such as the level contour reconstruction methods [33, 32], a front tracking method with an underlying grid [9], the particle level set method [6] and some related methods [12], the dynamic surface extension method [36], the fixed grid method in [30] and a closely related vector level set method in [40].

In [21] we have proposed a novel framework to model interface motions. The method naturally combines and takes advantages of both the Lagrangian (explicit) and the Eulerian (implicit) formulations. The basic idea is to represent and track the interface explicitly as in the usual Lagrangian methods using quasi-uniform meshless particles, while an underlying Eulerian grid serves as a reference for those particles. Even though the original GBPM [21, 20, 22] has already shown some promising results, we would like to propose several new modifications to the method to further improve its efficiency and its ease in implementation. According to [21], the GBPM provides an automatic redistribution of the sampling particle according to the underlying mesh. These sampling points (footpoints) are chosen to be the L^2-projection of all grid points in a neighborhood of the interface. We can therefore obtain a quasi-uniform sampling of the interface. Although we can control the maximum distance between two adjacent footpoints on the interface, the non-uniform behavior comes from the fact that there is no mechanism

*Department of Mathematics, the Hong Kong University of Science and Technology, Clear Water Bay, Hong Kong. Email: **yshon@ust.hk**

†Department of Mathematics, the Hong Kong University of Science and Technology, Clear Water Bay, Hong Kong. Email: **masyleung@ust.hk**

‡Department of Mathematics, University of California at Irvine, Irvine, CA 92697-3875, United States. Email: **zhao@math.uci.edu**

to bound the distance from below, i.e. we do not have any bound like $|\mathbf{y}_i - \mathbf{y}_{i+1}| > O(h)$ where $\mathbf{y}_i$ and $\mathbf{y}_{i+1}$ are two adjacent footpoints, and h is the underlying mesh size. For example, considering a uniform underlying Cartesian mesh in the two dimensional cases with a straight interface parallel to an axis direction, we obtain multiple footpoints projected onto same locations. This means that $(\mathbf{y}_i - \mathbf{y}_{i+1})$ could in fact be zero and the footpoint-grid point map may be far from one-to-one.

Another challenge to the GBPM is that it is not straight-forward to develop high order accurate numerical quadrature rules along the interface. Surface integrals arise in many circumstances including the boundary integral method [14, 28] and the boundary element method [3], where the solution is represented by integrals along its interface. The original GBPM proposed in [21] can indeed evaluate the integral by approximating the interface by piecewise connected linear function. For instance, we can keep track of a global parametrization associated to each footpoint. This provides an ordering of the interface which can be used to approximate Δs_i in the Trapezoidal rule

$$\int_\Sigma f(s)ds \simeq \sum_i f_i \Delta s_i \,.$$

The idea has recently been applied in [22] to integrate Gaussian beam solution to high frequency wave propagation. However, since the interface is approximated using only a low order reconstruction, the integration results are in general less accurate. Another approach has recently been used in [23] by first converting the GBPM representation to a signed distance function as in the level set method [27]. Then the integral can be computed using the δ-function formulation.

One main contribution of this paper is the following modifications to the original GBPM method which can naturally decompose the interface into a summation of high order disconnected segments. Given a computational domain Ω, we partition it into a union of *cells*, i.e. $\Omega = \cup_{i,j}\Omega_{i,j}$. If we divide the domain into a sum of rectangles with sides Δx and Δy, we can interpret this *cell*-type discretization as the staggered version of the original discretization described in [21]. The main difference here is that we denote $\mathbf{p}_{i,j}$ the cell-center, rather than the vertices of a cell (corresponding to a grid point in [21]). Moreover, in the case of triangulating Ω into a sum of triangles or tetrahedral, this *cell*-based is more natural. Relating to the *cell*-based domain partition, one simple modification is to activate only those *cells* which contain a segment of the interface. This activation criterion automatically partition the interface into a summation of segments. There are advantages of such an interface partition representation. For example, to visualize the solution in the level set representation, one has to post-process the level set function in order to extract the interface. In the original GBPM, one either simply plots the computed unconnected footpoints or reconstructs a level set representation in order to get an explicit curve for the same interface. In this proposed representation, we have a local closed form representation of the interface within each cell. This allows us to easily approximate any integral within each cell. In this paper, we will further apply the method to study some integral-dependent evolutions of an interface. This application will be important in various fields including multiphase flow modeling using a weak formulation [2], or high frequency asymptotic solutions to the wave equation or the Schrödinger equation [19, 18, 22]. Since this overall algorithm is a cell-based representation, we name the proposed numerical approach the Cell-Based Particle Method (CBPM).

To simplify the resulting algorithm from the above activation criterion, we can simply replace the L^2-projection in the definition of the footpoint [21] by the L^∞-projection in the Cartesian mesh, or the projection based on the area coordinates in the triangular mesh. If the underlying cells are Cartesian, we can determine the footpoints by

$$\min_s \|\mathbf{f}(s) - \mathbf{p}\|_\infty \,,$$

where $\mathbf{p}$ is the coordinates of the mesh point and $\mathbf{f}$ is the parametrization of the interface. If this L^∞-distance is smaller than $\Delta x/2$ (for $\Delta x = \Delta y$), the cell intersects with the interface. This condition therefore naturally matches with the activation condition in practice.

In the original GBPM, the reconstruction is done in a local coordinate system where one assumes the interface can be treated as a graph and is well-approximated by a local polynomial. This can still be easily done in the current CBPM representation. In this work, we also study an alternative to this *extrinsic* polynomial least squares fitting by considering the *intrinsic* fitting based on geometric basis. In particular, we locally approximate the interface in two dimensions using straight lines or circles. Therefore, no local coordinate system is required. Furthermore, since the interface is approximated using such simple geometric basis, it significantly simplifies many numerical differential operators defined on the manifold.

The paper is organized as follows. We will briefly summarize the GBPM in section 2. In section 3 we will introduce the proposed CBPM. Detailed descriptions of various steps in the algorithm will also be given. To further simplify various part of the algorithm, we propose in section 4 to incorporate the geometric basis for local reconstruction. Convergence tests and numerical examples are given in section 5 to demonstrate the effectiveness of the method.

2 Grid Based Particle Method

For the convenience of readers, we give a brief summary of the Grid Based Particle Method (GBPM). For a complete and detailed description of the algorithm, we refer the readers to [21, 20, 17] and thereafter.

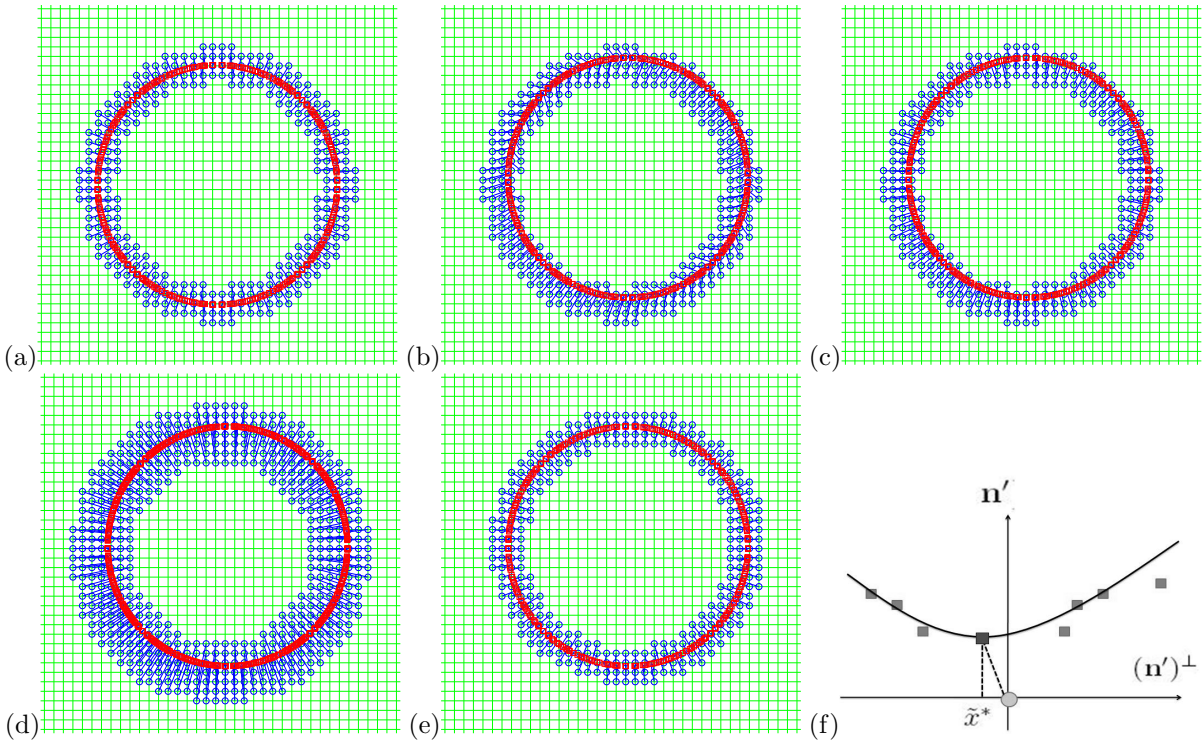


Figure 1: Grid Based Particle Method [21]. From left to right: (a) Initialization, (b) after motion, (c) after re-sampling, (d) after activating new grid points with their footpoints, (e) after inactivating grid points with their footpoints. (f) Determining the new footpoint using a local least-squares reconstruction of the interface.

In the GBPM, we represent the interface by meshless particles which are associated with an underlying Eulerian mesh. Each sampling particle on the interface is chosen to be the closest point from each underlying grid point in a small neighborhood of the interface. This one to one correspondence gives each particle an Eulerian reference during the evolution.

At the first step, we define an initial computational tube for active grid points and use their corresponding closest points as the sampling particles for the interface. A grid point $\mathbf{p}$ is called **active** if its distance to the interface is smaller than a given **tube radius**, ϵ, and we label the set containing all active grids Γ. To each of these active grid points, we associate the corresponding closest point on the interface, and denote this point by $\mathbf{y}$. This particle is called the **footpoint** associated to this active grid point. Furthermore, we can also compute and store certain Lagrangian information of the interface at the footpoints, including normal, curvature and parametrization, which will be useful in various applications.

This representation is illustrated in Figure 1 (a) using a circular manifold as an example. We plot the underlying mesh in solid line, all active grids using small circles and their associated footpoints on the manifold using squares. To each grid point near the interface (blue circles), we associate a footpoint on the interface (red squares). The relationship of each of these pairs is shown by a solid line link. To track the motion of the interface, we move all the sampling particles according to a given motion law. This motion law can be very general. Suppose the interface is moved under an external velocity field given

by $\mathbf{u}=\mathbf{u}(\mathbf{y})$. Since we have a collection of particles on the interface, we simply move these points just like all other particle-based methods, which is simple and computationally efficient. We can simply solve a set of ordinary differential equations using high order scheme which gives very accurate location of the interface. For more complicated motions, the velocity may depend on the geometry of the interface. This can be done through local interface reconstruction.

It should be noted that a footpoint $\mathbf{y}$ after motion may not be the closest point on the interface to its associated active grid point $\mathbf{p}$ anymore. For example, Figure 1 (b) shows the location of all particles on the interface after the constant motion $\mathbf{u}=(1,1)^T$ with a small time step. As we can see, these particles on the interface are not the closest point from these active grid points to the interface anymore. More importantly, the motion may cause those original footpoints to become unevenly distributed along the interface. This may introduce both stiffness, when particles are moving together, and large error, when particles are moving apart. To maintain a quasi-uniform distribution of particles, we need to resample the interface by recomputing the footpoints and updating the set of active grid points (Γ) during the evolution. During this resampling process, we locally reconstruct the interface, which involves communications among different particles on the interface. This local reconstruction also provides geometric and Lagrangian information at the recomputed footpoints on the interface.

The key step in the method is a least squares approximation of the interface using polynomials at each particle in a local coordinate system, $\{(\mathbf{n}')^{\perp},\mathbf{n}'\}$, with $\mathbf{y}$ as the origin, Figure 1 (f). Using this local reconstruction, we find the closest point from this active grid point to the local approximation of the interface. This gives the new footpoint location. Further, we also compute and update any necessary geometric and Lagrangian information, such as normal, curvature, and also possibly an updated parametrization of the interface at this new footpoint. For a detail description, we refer interested readers to [21].

3 Cell Based Particle Method

In this section, we first discuss several desired properties of our proposed representation, and then we will outline the general algorithm of our proposed method. Detailed descriptions of each step will be given in following subsections when the phase is different from the original GBPM.

3.1 Desired additional properties of the representation

Other than those nice properties mentioned in [21, 20] and thereafter, we would like to further incorporate several other new features in the new representation.

In the original GBPM, each sampling particle on the interface is chosen to be the closest point (defined in terms of the L^2-distance) from each underlying grid point adjacent to the interface. One nice thing about this representation is that we can obtain an upper bound on the distance between two adjacent sampling points. However for most locations of the interface, sampling points could be too close to each others, i.e. we do not have any lower bound on sampling distance. This could be problematic in some numerical computations. To stabilize the computations, in the previous local reconstruction, we required sampling points for least squares fitting have to be a significant distant apart.

In Figure 2 we have plotted numerically the cumulative distributions of the distances between two adjacent particles in the GBPM representation for different interface profiles using different grid sizes with $\Delta x=2^{-8}$, 2^{-10} and 2^{-12}. In Figure 2 (a), we determine analytically the L^2-projection of neighboring grid points onto a circle of radius $\pi/6$ using a rectangular Cartesian mesh with different grid sizes. In Figure 2 (b), we consider a five-folded star shape given by $r(\theta)=\pi/6+0.075\cos(5\theta)$. Here, we use a computational tube of radius $\gamma=1.5\Delta x$. We can clearly see that in the GBPM representation, the distribution does not depend heavily on the grid size. More importantly, significant amount of particles have very close neighbors and the cumulative distribution grows almost linearly in the sampling distance. In particular, almost 70% of sampling points have the closest neighbor closer than $0.4\Delta x$.

Other than a lower bound on the sampling distance, we would also like to introduce a partition of the interface so that we will be able to directly compute integrals on surfaces. In particular, we would like to obtain the surface element Δs_i such that the surface integral can be easily approximated by

$$\int_{\Sigma} f(s)ds \simeq \sum_i f(s_i)\Delta s_i,$$

where Δs_i is the length of each of these segments. Currently, there are two ways to determine a surface integral in the GBPM. The first one is based on a global parametrization. In [22], we keep track of a

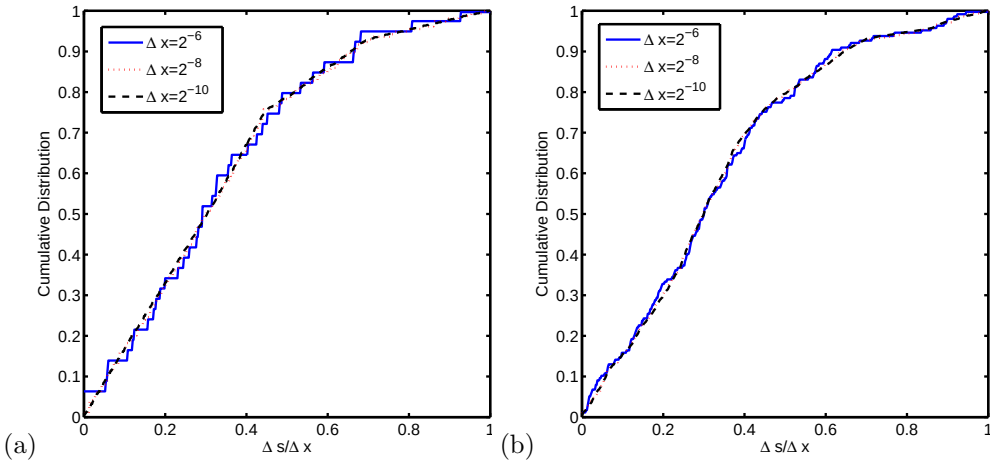


Figure 2: Cumulative distributions of the distance between two adjacent particles Δs_i in the GBPM with $\Delta x = 2^{-8}$, $\Delta x = 2^{-10}$ and $\Delta x = 2^{-12}$. (a) A circle represented by the GBPM. (b) A five-folded star shape represented by the GBPM.

global parametrization associated to each footpoint. This provides an ordering of the interface which can be used to approximate Δs_i in the Trapezoidal rule. However, this in fact introduces a piecewise linear approximation to the interface. Since the interface is approximated using only a low order reconstruction, the integration results are in general less accurate. Another approach for computing a surface integral is based on the level set representation. In [23], we convert the GBPM represented surface into a level set representation and approximate the integral using

$$\int_{\Sigma} ds = \int_{\mathbb{R}^d} \delta(\phi) d\mathbf{y},$$

where ϕ is defined on all grid points with value given by the distance between the grid point and its associated footpoint. Numerically, the δ-function is replaced by a smoothed version δ_ϵ. And so the accuracy might not be high enough for applications.

3.2 Algorithm

In this subsection, we outline the general algorithm of our proposed method. Detailed descriptions of each step will be given in following subsections.

Algorithm:

1. Initialization [Figure 3 (Left)]. Collect all cells containing a segment of the interface. From each of these cell centers, compute a projection onto the interface. We call these cells **active** and their corresponding particles on the interface **footpoint**. The interface is represented by these meshless footpoints (particles).
2. Motion [Figure 3 (Middle)]. Move all footpoints according to a given motion law.
3. Re-Sampling [Figure 3 (Right)]. For each active cell, re-compute the projection onto the interface reconstructed locally by those particles after the motion in step 2.
4. Update active cells. We activate any cell with an active neighboring cell and find their corresponding footpoints. Then, inactivate those cells which do not contain any interface segment.
5. Adaptation (Optional). Locally refine the underlying cells if necessary.
6. Iteration. Repeat steps 2-5 until the final computational time.

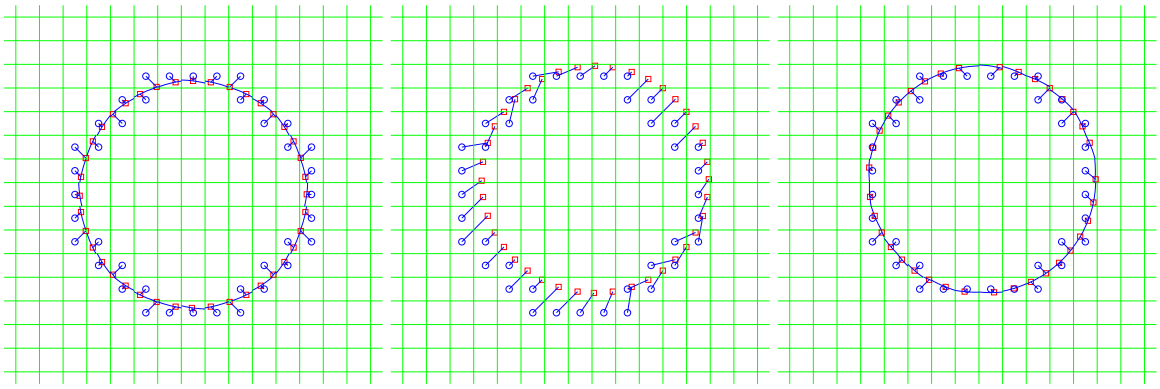


Figure 3: A simple demonstration of the Cell Based Particle Method: (Left) Initialization, (Middle) after a simple translation motion, (Right) after re-sampling and updating activated cells. We have demonstrated the evolution of a circle in the top-right direction using the proposed CBPM. We show the activated cell centers using blue circles, their associated sampling point using red square, the cell boundaries using green solid lines. In (Left), we have shown our interface representation. The activated cell centers are plotted using blue circles. Their corresponding projection onto the interface is shown using red square. In (Middle), we show our solution right after one motion step. The activated cells in (Left) are kept unchanged, while their corresponding sampling points, right squares, are moved according to the motion law. Then, we follow the resampling algorithm and the computational tube updating step to restore the projection property.

3.3 Interface representation

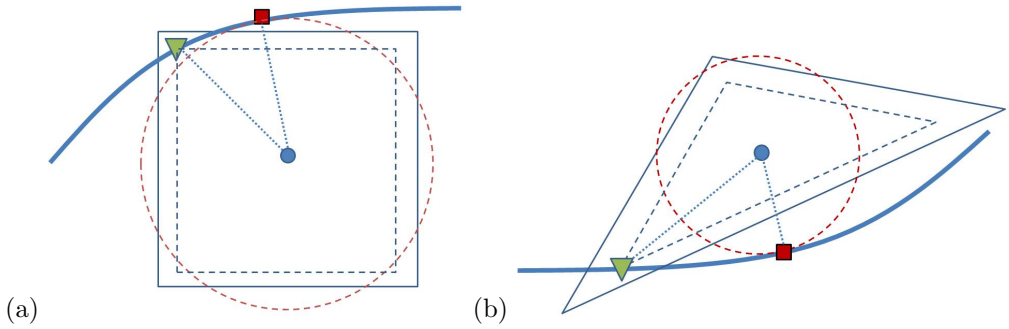


Figure 4: (a) Footpoint determined by the L^2- and L^∞-projections in a rectangular Cartesian cell. (b) Footpoint determined by the L^2-projection and the projection based on the area coordinates in a triangular cell. The interface are shown in thick solid lines. The cell boundary are drawn in thin solid lines. The cell centers are plotted using blue circles. The footpoints determined by the L^2-projection is shown using red squares. The footpoints determined by the L^∞-projection and the projection based on the area coordinates are shown using green triangles.

We first partition the computation domain into cells and we collect those cell centers $\mathbf{p}$ containing a segment of the interface. We will call these cells **active**, using the same terminology as in the GBPM. Numerically, the L^2-projection from the cell center is not a good indicator for identifying those cells which contain a segment of the interface. For example, even if the L^2-distance from the cell center to it's L^2-projection is greater than $\Delta x/2$, the interface may still intersect with the cell, as shown in Figure 4 (a). Instead, the projection should take care of the geometry of the underlying cell.

In the CBPM, if the underlying mesh is Cartesian with $\Delta x = \Delta y$, we propose to replace the L^2-projection from a grid point in the GBPM by the L^∞-projection from the cell center. If the L^∞-distance from the cell center to the L^∞-projection is less than $\Delta x/2$, this implies that there exists a point on the interface which stays inside the cell, and therefore the cell should contain a segment of the interface. Mathematically, if we have an explicit expression for approximating the interface locally, e.g., a local

parametrization $\mathbf{f}(s)$, we can minimize the following function to find the L^∞ projection

$$d(s;\mathbf{p}) = ||\mathbf{f}(s) - \mathbf{p}||_\infty . \tag{3.1}$$

This representation can be naturally extended to triangulated cells using the barycentric coordinates or the area coordinates, Figure 4 (b). In particular, any point $\mathbf{x}$ within a triangle with vertices $\mathbf{v}_i$ for $i=1,2,3$ can be written as a linear combination

$$\mathbf{x} = \sum_{i=1}^{3} \alpha_i \mathbf{v}_i .$$

The area coordinates of the point $\mathbf{x}$ is then given by $(\alpha_1,\alpha_2,\alpha_3)$ with the constraint $\sum_i \alpha_i = 1$. This implies that the cell center has the area coordinates $(1/3,1/3,1/3)$. We define the following distance from the cell center based on the area coordinates

$$\max\left[\left|\alpha_1 - \frac{1}{3}\right|, \left|\alpha_2 - \frac{1}{3}\right|, \left|\frac{2}{3} - \alpha_1 - \alpha_2\right|\right] . \tag{3.2}$$

Then the projection of the cell center onto the interface is defined to be a point in the area coordinates $(\alpha_1,\alpha_2,\alpha_3)$ which minimizes the distance given by (3.2).

To each of these active cells, we determine a representation by associating the minimizer to (3.1) on the interface, denoted by $\mathbf{y}$. This particle is called the **footpoint** associated to this active cell. In other words, a cell $\mathbf{p}$ is active only if the corresponding footpoint $\mathbf{y}$ is located inside the corresponding cell. This connection between an active cell and its footpoint is maintained during the evolution. Furthermore, like the GBPM, we can also compute and store certain Lagrangian information of the interface at the footpoints, including normal and curvature, which are useful in various applications.

Similar to the GBPM, the resolution of the particles in the CBPM on the interface relates to the mesh size. This provides an easy approach for interface representation adaptivity in the current algorithm. In the high curvature region or when two segments of the interface come close to each other, one just needs to locally refine the underlying Eulerian grid and add particles to resolve the interface better.

3.4 Motion

Also, with this cell centers and footpoints relation, we simply solve a set of ordinary differential equations using any high order scheme to evolve these footpoints just like the usual particle-based methods. This phase is the same as that in the GBPM and we are omitting the discussion here.

3.5 Re-Sampling

Like the GBPM, a footpoint $\mathbf{y}$ after motion might not be a good representative of the particular cell. In particular, the evolved point is usually no longer the L^∞-projection of its associated active cell center $\mathbf{p}$ onto the interface. So, these new sampling points might become unevenly distributed along the interface. To tackle with this problem, we resample the interface by recomputing the footpoints and updating all the active cells during the evolution in order to maintain a quasi-uniform distribution of particles. During this resampling process, we locally reconstruct the interface, which involves connections among neighboring particles on the interface. This local reconstruction also provides geometric and Lagrangian information at the recomputed footpoints on the interface. After the local reconstruction, we then find the new footpoint associated to each active cell center and compute any necessary Lagrangian information including normal and curvature. We will describe the procedure in details in the following subsections.

3.5.1 Remarks on choosing particles for local reconstruction

Similar to the GBPM, there are some remarks on the selection of m neighboring particles for local reconstruction in this new algorithm. Just like GBPM, we want these m particles to belong to the same segment of the interface. In the case when two segments of the interface are getting close to each other, these m neighboring particles should be consistently chosen to avoid using incorrect information in local reconstruction.

A more important criterion is that the particles collected have to be *significantly* distinct. If those m particles are very close to each other, the local reconstruction of the interface tends to be inaccurate and unstable. Moreover, choosing very close particles for reconstruction will usually lead to strict CFL time

step constraints. In the CBPM, on the other hand, choosing very close particles for reconstruction is unlikely to occur, since the footpoints are well separated for almost all scenarios (one example is shown in Figure 3). We found that the CBPM tends to give well distributed and well separated footpoints along the interface. As a result, it is less necessary to compare the distance with a particular particle with all other particles that we have collect and see if this new particle is too close to the others.

In Figure 2 we have plotted numerically the cumulative distributions of the distances between two adjacent particles in the GBPM representation for different initial profiles. We repeat the examples in Figure 5 for the CBPM by plotting the distributions with different mesh sizes using $\Delta x = 2^{-8}$, 2^{-10} and 2^{-12}. The new representation tends to have sampling rate similar to the mesh size. For example, at least from those two examples we are demonstrating, almost no two adjacent sampling points has distance smaller than $0.6\Delta x$. Similar to the observation from the GBPM, the distribution does not depend heavily on the mesh size. These observations suggest that the increase in the separation between two adjacent particles is intrinsic due to the sampling algorithm by implementing the L^∞-projection, rather than simply because of the number of sampling points in the representation. Indeed, we remark that the distribution in general depends on the interface itself. Moreover, similar to the GBPM, the CBPM also has no lower bound for distance between two adjacent particles. For some special cases, two adjacent cells could still have very close L^∞-projections on the interface. For example, when the interface intersects at the corner of a cell, two adjacent cells in the CBPM could still have the same L^∞-projection. In practice, however, the situation where two adjacent cells have close L^∞-projections are temporary (e.g. a single time-step) and is not expected to affect either the stability and the accuracy of the overall algorithm.

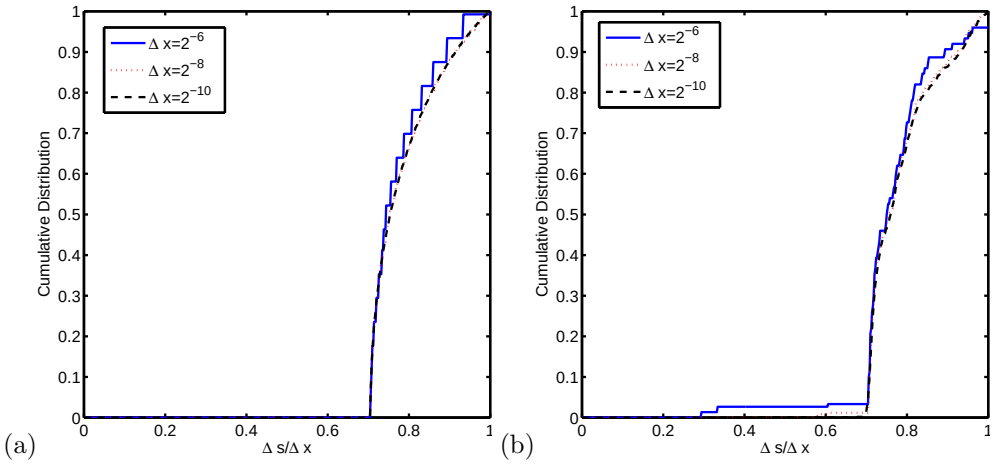


Figure 5: Cumulative distributions of the distance between two adjacent particles Δs_i in the CBPM with $\Delta x = 2^{-8}$, $\Delta x = 2^{-10}$ and $\Delta x = 2^{-12}$. (a) A circle represented by the proposed CBPM. (b) A five-folded star shape represented by the proposed CBPM.

3.5.2 New footpoints and their Lagrangian information

Following the algorithm in the GBPM, we use these m nearby footpoints to locally reconstruct the interface using a polynomial in a local coordinate system. We will skip the discussion of this reconstruction in the current work and refer the reader to [21, 20] and thereafter.

Once the local approximation of the interface is found, we determine the new footpoint $\mathbf{y}^*$ associated to $\mathbf{p}$. For simplicity, we concentrate this procedure in the two dimensions, higher dimensional generalization is straight-forward. Even though the L^∞ problem is not differentiable, we found that the L^∞-projection actually has a simple solution. Since the footpoint is determined by finding the smallest L^∞-ball which touches the local polynomial, we first compute two sets of potential projection points. The first set is all intersections between the diagonal lines of an active cell and its associated local polynomial. This can be easily found by solving a system of algebraic equations. However, finding only this set of points is not enough to determine the L^∞-projection since there could be cases where intersection does not exist or the L^∞-projection might not be along the cell diagonal, as shown in Figure 6. We have to consider also those points on the local approximation where the $x-$ or the $y-$ derivative is zero. Then, we sort these points according to their L^∞-distance to the cell center, and the minimizer will be

the new footpoint $\mathbf{y}^*$. Indeed, other approaches to solve the minimization problem are also possible. For example, one can approximate the L^∞-norm by the L^p-norm for some $p \gg 2$ and the minimizer can then be found easily by the Newton's iterations. But we will not further investigate the idea in the paper. If the L^∞-distance of this footpoint is great than $\Delta x/2$, the (approximated) interface has no intersection with the cell and we will simply deactivate the cell from the computational tube.

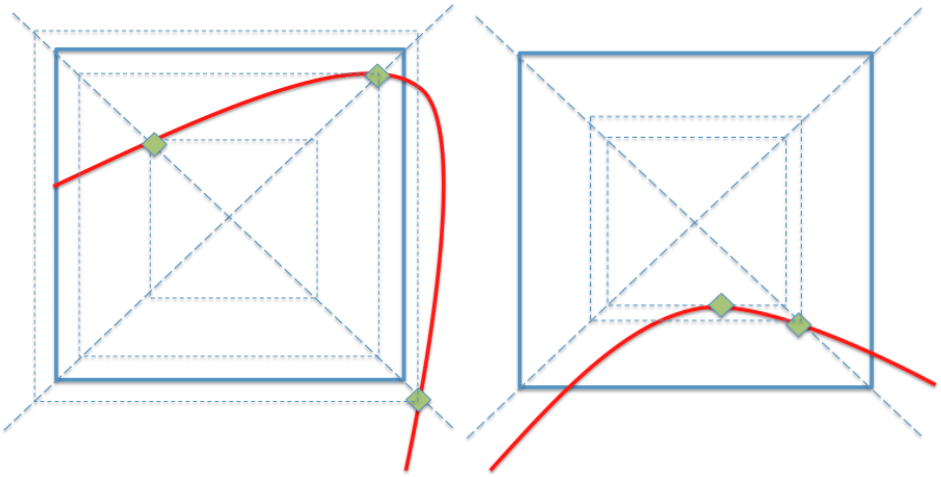


Figure 6: Determining the L^∞-projection of the cell center onto the local polynomial. The cell boundary (blue) and the local polynomial (red) are shown in solid lines. The intersections between the cell diagonals and the local polynomial reconstruction are shown in solid dots. For some situations, the L^∞-ball can touch the local reconstruction, i.e. at a point where the $x-$ or the $y-$ derivative of the curve is zero. The new footpoint $\mathbf{y}^*$ is chosen to be the closest intersection, or tangent, point.

3.6 Updating the computational tube

To maintain a complete description of the moving interface, one needs to keep adding and/or deleting active cells as well as their associated footpoints in time. This process consists of an activation phase and an deactivation phase, as in the original GBPM. After we have resampled the interface, we first activate those inactive cells $\mathbf{p}$ which are neighboring to an active grid. Then we deactivate those cells which do not contain any interface segment by simply checking if $\|\mathbf{y}-\mathbf{p}\|_\infty \le \Delta x/2$.

3.7 Computing integrals on surfaces

A highlight in our proposed algorithm is that we now have a naturally way to approximate an integral of the form $\int_\Sigma f(s)\,ds$ for the arc-length parameterization s for some regular integrand. We consider only the two-dimensional case here, approximation in higher dimensions is also possible.

For each active cell $\mathbf{p}_i$, we not only have a footpoint as a representative of the cell, but also a local polynomial approximation of the interface within the cell. In particular, we let Σ_i' be the segment of this local fitting which stays within the cell. So it's length $\Delta s_i = |\Sigma_i'|$ will be a high order approximation to the integral $\int_{\Sigma_i} ds$ with Σ_i be the portion of the interface Σ which stays within the cell centered at $\mathbf{p}_i$. Therefore, even though we are not claiming any high order accuracy for a general surface integral, a natural way to approximate a surface integral in the CBPM is given by

$$\int_\Sigma f(s)ds \approx \sum_i f(\mathbf{y}_i)\Delta s_i,$$

where $\mathbf{y}_i$ is the L^∞-projection of $\mathbf{p}_i$. In Figure 3, we have also shown these segments Σ_i' within all activated cells, even though they have not been used at all in that simple translation motion. One can clearly see that these segments give an extremely well approximation of the overall interface Σ. Similar approach can be done in higher dimensions, i.e. the surface element Δs_i can be approximated using the surface area of the portion of the local reconstruction which stays within the corresponding cell. Numerically, one can first convert the local reconstruction to an implicit representation within the

corresponding cell by finding the footpoints of the cell vertices and hence their singed distances. Then the surface area can be determined in MATLAB using the function isosurface and patch which explicitly extract the corresponding portion of the surface using triangulations. If a geometric basis is used for local reconstruction (which will be discussed in the next section), the area of that particular portion on a sphere can be found by first explicitly writing out the corresponding integral in the typical Cartesian coordinate system and then performing a numerical integration.

As a simple application, we consider the following projection method for imposing a volume preserving constraint on various flows. Such volume preserving constraint is important in many applications, such as the modeling of lipid vesicles flow [39, 31]. Preserving the enclosed volume of the region Ω bounded by $\partial\Omega=\Sigma$, we have

$$\int_\Omega \nabla\cdot\mathbf{u}=\int_\Sigma \mathbf{n}\cdot\mathbf{u}=\int_\Sigma u_n=0. \tag{3.3}$$

To impose the constraint to an arbitrary normal motion law u_n^0 induced by some given energy $E^o(\Sigma)$, we propose a simple projection method. The idea is to determine a correction v_n to the normal velocity $u_n^n=u_n^o+v_n$ so that the new normal velocity satisfies the constraint (3.3). This implies

$$v_n=\frac{\int_\Sigma u_n^o\,ds}{\int_\Sigma ds}\approx\frac{\sum_i u_n^o(\mathbf{y}_i)\Delta s_i}{\sum_i \Delta s_i}. \tag{3.4}$$

4 Local reconstruction using a geometric basis

In the GBPM and the CBPM discussed in previous sections, the resampling step is done on a local coordinate system where we approximate the interface locally using the least squares polynomial fitting. In this section, we study the possibility of replacing such local polynomial basis for local reconstruction using a geometric basis. Instead of fitting $f(s)\in\mathbb{P}^n$ a degree-n polynomial, we consider using a geometric basis which offers an *intrinsic* method for interface approximation. We will demonstrate that such basis can simplify several parts of the algorithm and we will also discuss some short-comings of this approach in this section.

4.1 A geometric basis

In two dimensions, a straight line is the lowest order element in our geometric basis, i.e. the curvature $\kappa(s)=0$ for the arc-length parametrization s. This reconstruction is the same as our previous local approximation using a degree-1 polynomial. The next order element in the geometric basis is a circle, i.e. the curvature $\kappa(s)=\kappa_0$. Higher order elements will be higher order polynomials in the curvature, i.e. they have the form of $\kappa(s)=\sum_{i=0}^n \kappa_i s^i$. In this work, we will concentrate mostly only circles in the geometric basis. It is possible to use higher order elements such as an Euler spiral ($\kappa(s)=\kappa_0+\kappa_1 s$) for local intrinsic approximation, but we find that using a circle for local reconstruction already gives enough accuracy for most applications and we do not pursue those higher order elements in this work. In three dimensions, one can use geometric objects like an sphere or an ellipsoid for local reconstruction.

We are definitely not the first one to consider the above geometric basis. For example, [34] has proposed to incorporate geometrical objects with ENO for shock capturing *interpolation*. We are not, however, aware of using such a geometric basis in the *least squares fitting* framework for modeling interface dynamics.

4.2 Local reconstruction

Mathematically, we apply the following re-sampling procedure to determine such local circle approximation. For each active cell $\mathbf{p}$ with the associated sampling particle $\mathbf{y}$ (no longer be the L^∞-projection of $\mathbf{p}$ onto the interface after motion), we first collect at least m active cells $(\mathbf{p},\mathbf{p}_1,\cdots)$, in a small neighborhood of $\mathbf{p}$, and also their associated sampling particles on the interface $(\mathbf{y},\mathbf{y}_1,\cdots)$. Then we approximate the interface locally by geometric objects using nonlinear least squares fitting. For a circle in two dimensions in particular, we fit a circle by determining its center $\mathbf{x}_c=(x_c,y_c)$ and its radius r by minimizing the sum of the squares of the geometrical distance from the sampling particles to their corresponding L^2-projection onto the circle, i.e.

$$\min_{\mathbf{x}_c,r_c}\left[(\|\mathbf{y}-\mathbf{x}_c\|_2-r_c)^2+\sum_i(\|\mathbf{y}_i-\mathbf{x}_c\|_2-r_c)^2\right].$$

Numerically, this nonlinear function can be minimized by Newton's method [8]. For a small timestep evolution in the interface, the local geometry of the interface does not significantly altered and so we actually have a relatively good initial guess of the iteration. We find that the minimizer can be achieved with very good accuracy in only few iterations.

In some three dimensional cases when the difference in the principle curvatures of the interface are important for the corresponding motion, we could use an ellipsoid for local reconstruction instead of a sphere. Similar to the fitting for circles in two dimensions, we consider the nonlinear least squares fitting approach based on the geometrical distance. We adopt the approach in [38] and the nonlinear function in the of ellipsoids can also be minimized numerically by the Newton's method. As demonstrated as in the example section, however, we found that using spheres for local reconstruction already gives enough accuracy and, therefore, we do not pursue this high order approach in this work.

4.3 Normal

Except that the fitting is independent of a local coordinate system, the geometric basis can also simplify various expressions in the algorithm. For example, the normal vector at the footpoint $\mathbf{y}$ can be approximated by the normal vector of the local reconstructed circle at the desired location, which is simply given by the expression

$$\mathbf{n}(\mathbf{y}) = \frac{\pm 1}{r_c}(\mathbf{y}-\mathbf{x}_c),$$

where the $\pm$-sign is determined in such a way that the new normal vector is consistent with the old normal at the associated sampling point before the local reconstruction step. For example, one can simply use

$$\mathbf{n}(\mathbf{y}) = \frac{1}{r_c}\operatorname{sgn}[(\mathbf{y}-\mathbf{x}_c)\cdot\mathbf{n}'](\mathbf{y}-\mathbf{x}_c),$$

where $\mathbf{n}'$ is the normal at the previous footpoint.

4.4 Local curvature

The curvature at the new footpoint can be easily approximated using the curvature of the least squares circle, i.e.

$$\kappa(\mathbf{y}) = \frac{\pm 1}{r_c},$$

where the sign is again determined by matching with the sign of the curvature at the associated sampling point before local reconstruction. Three dimensional problems can be done using a local sphere as an approximation. In dimensions higher than 2, however, a hyper-sphere is indeed not the best candidate for the local approximation, since the reconstruction is approximating the hyper-surface locally using the same value for all principal curvatures. To obtain a more accurate local approximation, one can use other geometric basis which contains elements with different principal curvatures such as ellipsoids to reconstruct the hyper-surface. Unfortunately, it is not easy to design a good least squares solver for the ellipsoid fitting problem [15, 4, 7, 26, 24]. We will simply fit a hyper-sphere even in higher dimensions in this paper.

4.5 Surface Laplacian

Considering an interface Σ parametrized by (s_1, s_2), the surface Laplacian of a function f in GBPM is given be the following expression

$$\Delta_\Sigma f = \sum_{i,j=1}^{2} \frac{1}{\sqrt{g}}\frac{\partial}{\partial s_i}(\sqrt{g}g^{ij}\frac{\partial f}{\partial s_j}),$$

where the coefficient g^{ij} are the components of the inverse of the metric tensor $[g_{ij}]$, whose components are given by

$$g_{ij} = \frac{\partial \mathbf{X}}{\partial s_i}\frac{\partial \mathbf{X}}{\partial s_j},$$

with the surface Σ is given by $\mathbf{X}(\mathbf{s_1}, \mathbf{s_2})$. In our algorithm using the geometric basis, the expression for the surface Laplacian can be significantly simplified. For two dimensional cases and having a circle as a

locally reconstructed approximation to the interface, we can simply rewrite the surface Laplacian in the polar coordinate system centered at the fitting circle

$$\Delta_\Sigma f = \frac{1}{r_c^2}\frac{\partial^2 f}{\partial \theta^2}.$$

Since the function f is defined on the footpoints $\mathbf{y}$, we fit f using a local squares quadratic polynomial, i.e. $f(\theta)=a_0+a_1\theta+a_2\theta^2$. This implies that the surface Laplacian on the least squares circle can be approximated by $\Delta_\Sigma f = 2a_2/r_c^2$. Generalizing this approximation to three dimensions is straightforward. Instead of using the polar coordinate system, we consider the spherical coordinate centered at the center of the least squares fitting sphere for a locally reconstructed sphere in a cell

$$\Delta f = \frac{\partial^2 f}{\partial r^2}+\frac{2}{r}\frac{\partial f}{\partial r}+\frac{1}{r^2\sin^2\phi}\frac{\partial^2 f}{\partial \theta^2}+\frac{\cos\phi}{r^2\sin^2\phi}\frac{\partial f}{\partial \phi}+\frac{1}{r^2}\frac{\partial^2 f}{\partial \phi^2}.$$

Again, this expression can be simplified if we locally use spheres in the reconstruction,

$$\Delta_\Sigma f = \frac{1}{r^2\sin^2\phi}\frac{\partial^2 f}{\partial \theta^2}+\frac{\cos\phi}{r^2\sin^2\phi}\frac{\partial f}{\partial \phi}+\frac{1}{r^2}\frac{\partial^2 f}{\partial \phi^2}.$$

In three dimension, we assume the f is a function of θ and ϕ. We again fit f using a local squares quadratic polynomial, i.e.

$$f(\theta,\phi)=\sum_{i=0}^{2}\sum_{0\le i+j\le 2} a_{ij}\theta^i\phi^j.$$

Therefore, the surface Laplacian can then be approximated by computing the corresponding derivatives of this local polynomial approximation. Comparing to those expressions in [17], these corresponding formulas in the CBPM are now greatly simplified because of the geometric objects such as circles and spheres we used in the local reconstruction.

4.6 Comparison of the local fitting using local polynomial basis and geometric basis

	Local Quadratic Polynomial	Geometric Basis
Fitting function	$\mathbf{X}'(t)=a_0+a_1t+a_2t^2$	$\|\mathbf{y}-\mathbf{x}_c\|_2=r_c$
Local coordinate system	$(\mathbf{n},\mathbf{n}^\perp)$	intrinsic
Unknowns in the fitting	a_0,a_1,a_2	$\mathbf{x}_c=(x_c,y_c),r_c$
Normal at the footpoint $\mathbf{y}$	$\pm\frac{(\mathbf{X}'(s^*))^\perp}{\|\mathbf{X}'(s^*)\|}$	$\pm\frac{\mathbf{y}-\mathbf{x}_2}{r_c}$
Curvature at the footpoint $\mathbf{y}$	$\pm\|\mathbf{X}''(s^*)\|$	$\pm\frac{1}{r_c}$

Table 1: Comparison of the local least squares fitting (left) using a degree-2 polynomial and (right) using a circle.

In table 1, we have summarized and compared several quantities computed using a degree-2 polynomial and a circle for local reconstruction, respectively. Because of the geometry used in the local fitting, all of these computations in the local reconstruction have been significantly simplified. One further remark is that the degrees of freedom in the fitting are the same for both approaches only in fitting a co-dimensional one object in two dimensions. For higher dimensions, on the other hand, the geometric basis has significantly smaller number of unknowns in the fitting. In particular, fitting a co-dimensional one surface in the d-dimensional space, the local degree-2 polynomial has d^2-d+1 unknowns, while the local sphere fitting has only $d+1$ unknowns. Therefore, local least squares fitting using a geometric basis is particular attractive for high dimensional modeling.

4.7 A remark on choosing footpoints for extremely high curvature region

In the new CBPM, choosing the correct footpoints from the geometric object after local reconstruction requires special treatment when the local curvature is extremely high (when $|\kappa|\sim 1/\Delta x$). After local

reconstruction on a high curvature segment, we found that the new footpoint $\mathbf{y}^*$ might actually be located on the wrong segment of the reconstructed surface, as indicated in Figure 7. Figure 7 shows a falsely active cell with a footpoint located inside (the dark cross), while the reasonable one is outside the cell (the dark square).

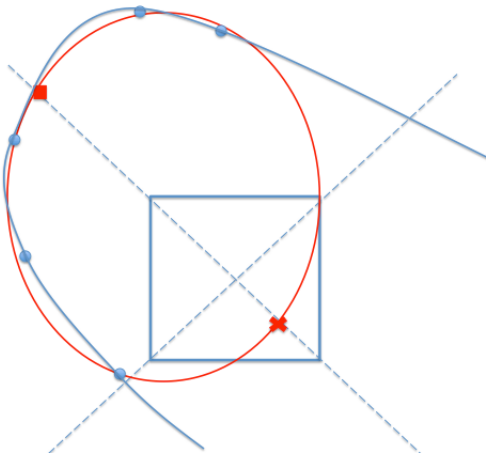


Figure 7: A falsely activated cell and its footpoint (dark cross within the cell). The interface (black) and the local reconstruction (red circle) are shown in solid lines. The sampled points (blue dots) are used in the local construction and the reasonable potential footpoint (dark square) is shown on the geometrical object outside the cell.

Different approaches can be applied to determine the more reasonable footpoints. The first one is to follow the approach in [20] for dealing with open surfaces in the GBPM representation. One imposes an extra constraint to make sure the new footpoint is obtained by *interpolation* but not *extrapolation* in the local coordinate system. In 2D where the interface is a curve, one needs to make sure the footpoint corresponds to a parametrization within a finite interval. In 3D where the interface is a surface, one makes sure that the new representation comes within the convex hull formed by neighboring points.

To make sure the new footpoint is on the same side of the sphere, we can also impose the following criterion:

$$\min_i \left[\frac{(\mathbf{y}_i - \mathbf{x}_c)\cdot(\mathbf{y}^* - \mathbf{x}_c)}{r_c \|\mathbf{y}_i - \mathbf{x}_c\|_2} \right] \geq \cos\theta_0 ,$$

for some $0 < \theta_0 < \pi/2$ (we pick $\cos\theta_0 = 0.7$ in our current algorithm). In other words, we require the angle between $(\mathbf{y}^* - \mathbf{x}_c)$ and all others $(\mathbf{y}_i - \mathbf{x}_c)$ to be less than θ_0 for all i. If the criterion is not hold, we will look at other candidates in the set of potential footpoints as discussed in Section 3.5.2. If the set is empty, we simply inactivate the cell.

5 Examples

In this section, we will first consider several accuracy issues of the geometric basis fitting and also the overall algorithm. Several motions will be tested including advection motion, motion in the normal direction, motion by mean curvature, and higher order geometrical motions like the motion by surface diffusion. We will also demonstrate that our method can control the change in the topology in the evolution.

5.1 Local reconstruction using a geometric basis

To demonstrate the feasibility of the proposed geometric fitting for high curvature interface, we first consider a two dimensional continuous test case in which a circle is used to fit the following parabola in the neighborhood of $x = 0$

$$y = 1 - \frac{x^2}{h^2}$$

with $h=0.25$. We determine a circle centered at $(0,y_c)$ with radius r_c to minimize its geometric distance from the parabola, i.e. mathematically we minimize

$$\left[\int_{-k}^{k}\left[r_c-\sqrt{\left(1-\frac{x^2}{h^2}-y_c^2\right)^2+x^2}\right]^2 dx\right]^{\frac{1}{2}} \tag{5.5}$$

with respect to y_c and r_c.

Figure 8 shows different least squares circles constructed with different values of k. As expected, the smaller the value of k (the more *local* the region for local reconstruction), the better the circle fits the parabola and, therefore, the better the geometrical approximation to the high curvature region of the interface. To better quantify the mismatch, we plot in Figure 9 the corresponding change in the quantity (5.5) and also the L^∞-error

$$\max_{x\in[-k,k]}\left|r_c-\sqrt{\left(1-\frac{x^2}{h^2}-y_c^2\right)^2+x^2}\right|$$

against the range of domain k. In the figure, we have also plotted two reference curves corresponding to the convergence in $O(k^6)$ and $O(k^8)$, respectively. As long as the sampling footpoints in the local reconstruction are close enough to capture the local high curvature of the interface, the geometric approximation will give highly accurate results.

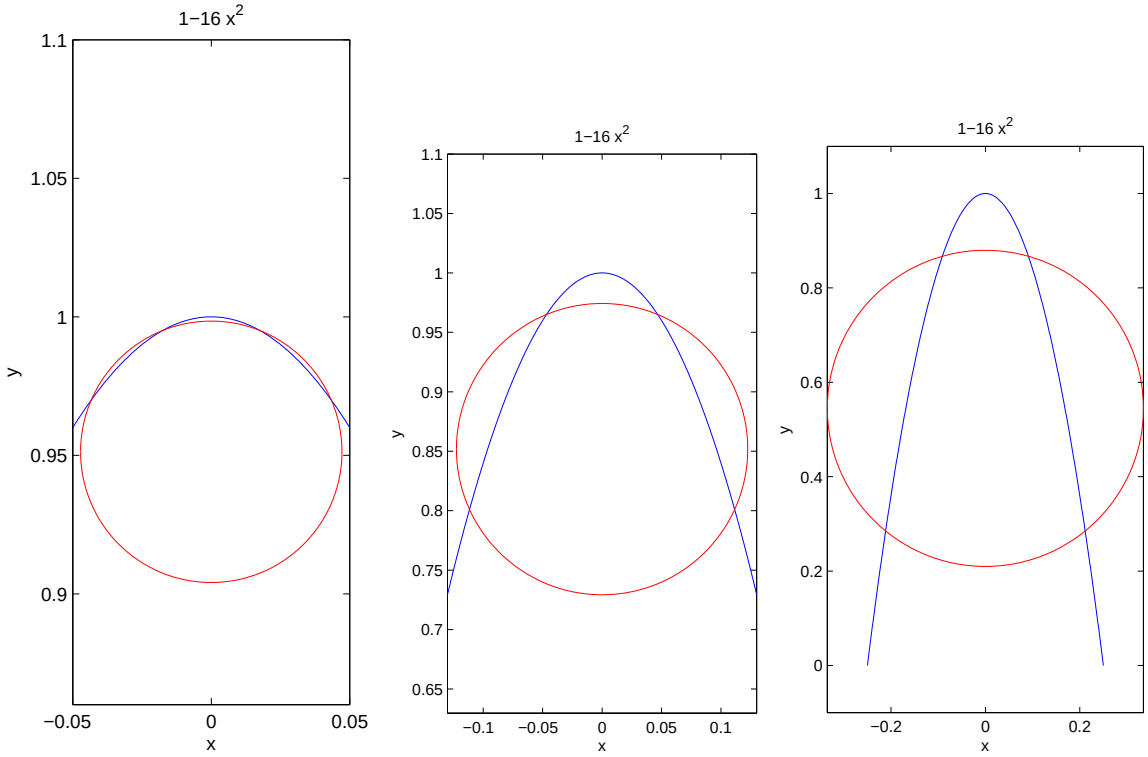


Figure 8: The geometric fitting for the test case with (left) $k=0.05$, (middle) 0.11 and (right) 0.25.

A more important case is the three dimensions. We repeat a similar example as in the two dimensions and least squares fit a sphere to the following paraboloid

$$z=1-\frac{x^2}{h_x^2}-\frac{y^2}{h_y^2}.$$

Mathematically, we compute the geometric distance between the paraboloid and the sphere by minimizing

$$\left[\int_{-h}^{h}\int_{-k}^{k}\left[r_c-\sqrt{\left(1-\frac{x^2}{h_x^2}-\frac{y^2}{h_y^2}-z_c^2\right)+x^2+y^2}\right]^2 dxdy\right]^{\frac{1}{2}} \tag{5.6}$$

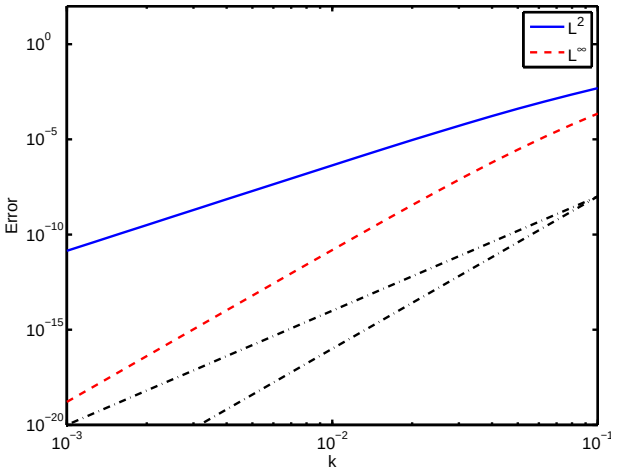


Figure 9: The L^2- and the L^∞-errors as a function of the size of the fitting region (in terms of k) in the example from Figure 8. The two black dot-dash lines on the bottom are reference curves for $O(k^6)$ and $O(k^8)$, respectively.

over the domain $[-k,k]\times[-h,h]$ with respect to the center and the radius of the sphere. Figure 10 shows a paraboloid with $h_x=h_y=0.25$ together with several least squares spheres using various h and k. In Figure 11 (a), we have computed the change in the L^2- and the L^∞-errors with respect to the size of the least squares fitting region (in terms of k with $k=h$). Again, the result concludes that the geometrical approximation will have less error when k is small enough. Concerning the convergence rate, we also plot two reference curves which correspond to the $O(k^2)$ and $O(k^6)$, respectively.

A more challenging case is actually when the two principal curvatures of the quadratic surface at $(0,0)$ are significantly different. For example, we consider $h_x=0.25$ and $h_y=10$ and so the surface is locally almost cylindrical. Since we are using only a sphere for local reconstruction, we do not expect the reconstruction can accurately compute the curvatures (the mean curvature and the Gaussian curvature). Nevertheless, as demonstrated in Figure 11 (b), we found that the least squares sphere still gives a high order approximation to a surface which has two significantly different principle curvatures.

5.2 Convergence study

In the previous section, we consider only the overall fitting error in the least squares approximation using circles and spheres. In this section, we investigate the error and its convergence in the location of the footpoints by the L^∞-projection. Unfortunately, similar to the GBPM, it is difficult to make a rigorous analysis of this approximation error due to the lack of pre-determined structure on sampling particles. We will instead show the accuracy in the re-construction step by a series of numerical tests.

5.2.1 Footpoints

We first consider a six-folded symmetric interface profile given by

$$\mathrm{r}=\mathrm{r}_0+\epsilon\cos(6\theta) \tag{5.7}$$

at $\mathbf{x}_c=(0.5,0.5)$, for $0\le\theta\le 2\pi$, where $\mathrm{r}_0=0.2$ and $\epsilon=\pi/150$. For each active cell $\mathbf{p}$ near the star-shaped interface, we first assign the exact L^∞-projection on the interface. We then locally construct a circle using our re-sampling algorithm with five points in the local reconstruction, i.e. $m=5$. For each fixed Δx, we define the following L^2- and L^∞-errors defined by

$$E^2_{\Delta x}=\left[\int_\Sigma|\mathcal{P}_\Sigma(\mathbf{y}_{\Delta x})-\mathbf{y}_{\Delta x}|^2ds\right]^{1/2},$$
$$E^\infty_{\Delta x}=\max_s|\mathcal{P}_\Sigma(\mathbf{y}_{\Delta x})-\mathbf{y}_{\Delta x}|,$$

where $\mathbf{y}_{\Delta x}$ is the numerical approximation from the local least squares fitting and $\mathcal{P}_\Sigma(\mathbf{y}_{\Delta x})$ is the L^2-projection of the point $\mathbf{y}_{\Delta x}$ onto the exact interface Σ, as shown in Figure 12.

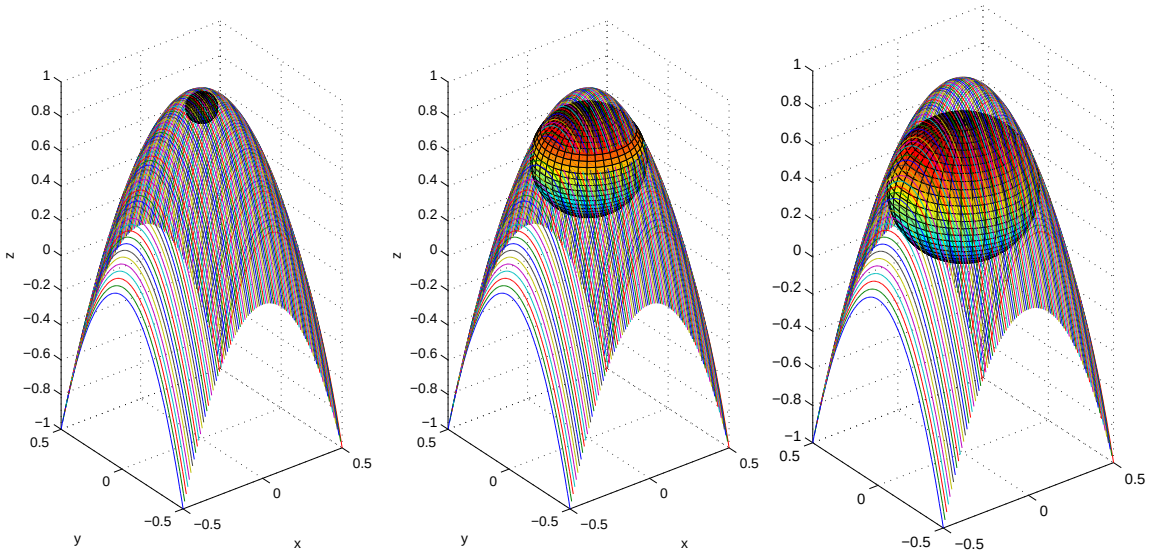


Figure 10: The geometric fitting for the test case with (left) $k=0.01$ and $h=0.23$; (middle) $k=0.03$ and $h=0.11$; (right) $k=0.25$ and $h=0.25$.

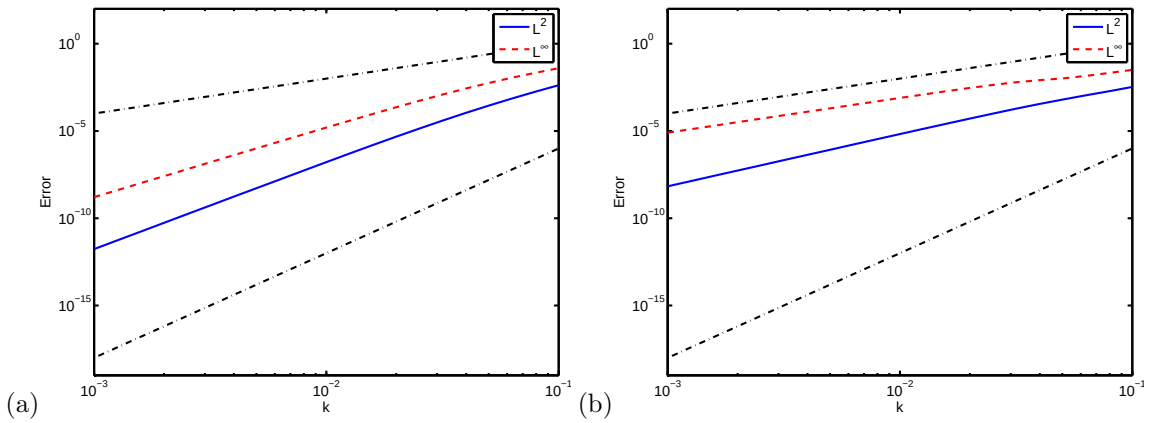


Figure 11: The L^2- and the L^∞-errors as a function of the size of the fitting region (in terms of k) in the example from Figure 10. The two black dot-dash lines on the top and on the bottom are reference curves for $O(k^2)$ and $O(k^6)$, respectively. (a) $h_x=h_y=0.25$ and (b) $h_x=0.25$ and $h_y=10$.

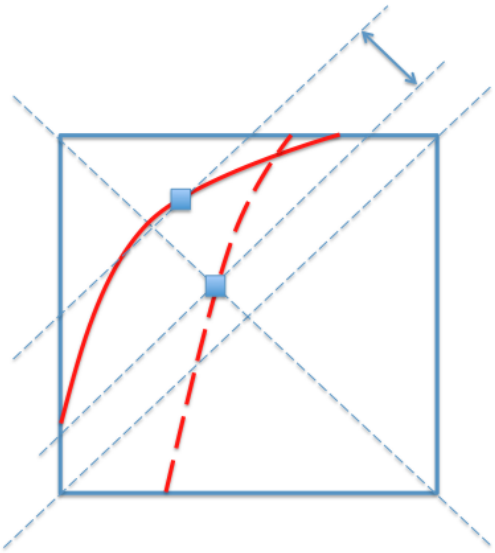


Figure 12: Definition of the measurement in error. The thick solid line and the thick dashed line within the cell represent the exact interface and the local approximation, respectively. The solid square dot along the diagonal is the computed footpoint of this cell. The other solid square dot on the exact interface is the L^2-projection of the computed footpoint onto the interface. The error is the L^2-distance between these two points.

Figure 13 (a) (log-log plot) shows the convergence as the underlying grid refines. The solid line shows log-log plot of the least squares fitting of the error in the form

$$E_{\Delta x} = c_1(\Delta x)^{c_2},$$

whose slope gives an approximation to c_2 and the rate of convergence. The result show that the error made in the local reconstruction converges to zero like $O(\Delta x^3)$.

We repeat a similar example in three dimensions by considering a sphere perturbed by the real spherical harmonic Y_4^4

$$r = r_0 + 0.05 \times \frac{3}{4}\sqrt{\frac{35}{\pi}}\cos(4\phi)\sin^4\theta,$$

where $r_0 = 0.15$ and $\mathbf{x} = (x_0, y_0, z_0) = (0.5, 0.5, 0.5)$. The results in Figure 13 (b) shows that the rate of convergence of the L^2-error is approximately 2.

Figure 13 (c) shows the total number of the number of footpoints on the interface versus the number of underlying cells in each direction, denoted by N. As expected, the number of sampling particles grows only linearly with the number of grid.

5.2.2 Arc-length

Considering the same interface defined in equation (5.7), we have also studied the convergence in computing the arc-length of an interface. We define the following error $E_{\Delta x} = |I_{exact} - I_{\Delta x}|$, where I represents the arc-length of a closed interface. The method computes the surface area with error converges in approximately $O(\Delta x^3)$, Figure 13 (d).

5.2.3 Local curvature

The final convergence study without evolution shown in Figure 13 (e) is the curvature on interface. Similarly, we consider the same interface defined in equation (5.7) and define the following L^2-error by

$$E_{\Delta x}^2 = \left[\int_\Sigma |\kappa_{\mathcal{P}_\Sigma(\mathbf{y}_{\Delta x})} - \kappa_{\mathbf{y}_{\Delta x}}|^2 ds\right]^{1/2},$$

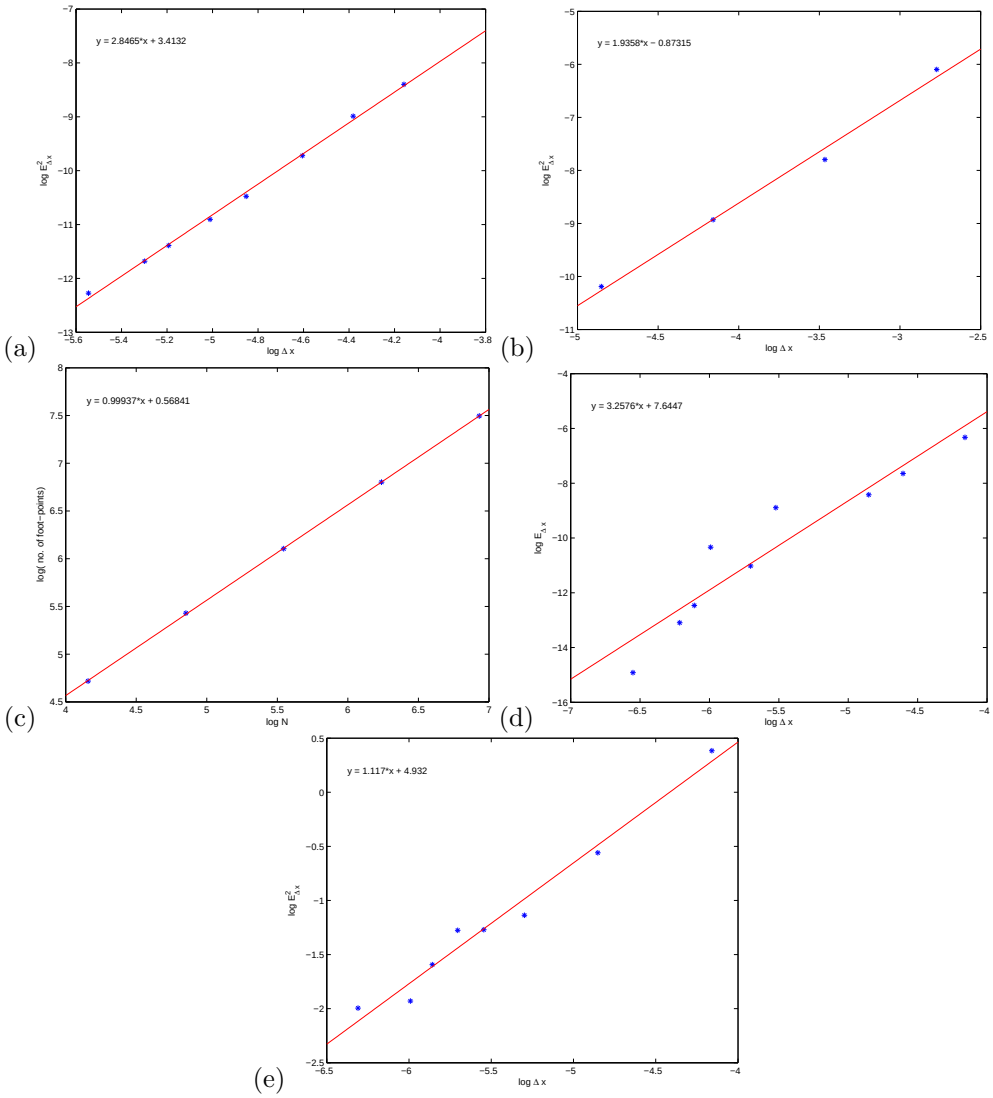


Figure 13: Convergence study for local geometric fitting of a star-shaped curve (in the two dimensions) or surface (in the three dimensions). (a) L^2-error in the footpoints for the two dimensional case (rate is approximately 3). (b) L^2-error in the footpoints for the three dimensional case (rate is approximately 2). (c) The total number of active cell-points without motions for the three dimensional case (rate is approximately 1). (d) Error in the arc-length for the two dimensional case (rate is approximately 3). (e) Error in the local curvature for the two dimensional case (rate is approximately 1).

where $\kappa_{\mathbf{y}_{\Delta x}}$ is the numerical approximation of local curvature and $\kappa_{\mathcal{P}_\Sigma(\mathbf{y}_{\Delta x})}$ is the local curvature of the L^2-projection of the point $\mathbf{y}_{\Delta x}$ onto the exact interface Σ. The result shows the error converges in approximately $O(\Delta x)$.

5.2.4 Evolution under the rigid body rotation

The above tests show only the approximation errors in the reconstruction step without any evolution. Next we start with the same interface defined in equation (5.7) and consider the evolution under the rigid body rotation, $\mathbf{u}=(u,v)$ where

$$u = 1-2y,$$
$$v = 2x-1.$$

At $t=\pi$, the interface rotates exactly one revolution and it is the same as the initial profile. Numerically we pick $\Delta t=(\pi/10)\Delta x$ and use RK2 in time discretization. The convergence results are shown in Figure 14 (a) which shows that the rate of convergence of the L^2-error and L^∞-error are both approximately 2. In Figure 14 (b) and (c), we study also the convergence (measured in the L^2- and the L^∞- norms) rates by looking at the number of footpoints used in sampling the interface. Since the number of sampling points is inversely proportional to Δx, we expect to obtain the same order in the convergence and it has been confirmed numerically in these figures. One interesting thing is to compare this with that by the GBPM. We have plotted the result by the GBPM in these plots. These two plots show that the CBPM and GBPM tend to have similar convergence rate in terms of Δx. Even though the errors in these CBPM solutions are smaller than those by the GBPM, we concentrate on the convergence rates but do not claim that CBPM will always produce more accurate solutions than the GBPM. This is because the accuracy in these solutions depend on several factors such as the number of footpoints used in the local reconstruction and the radius of the computational tube in the GBPM.

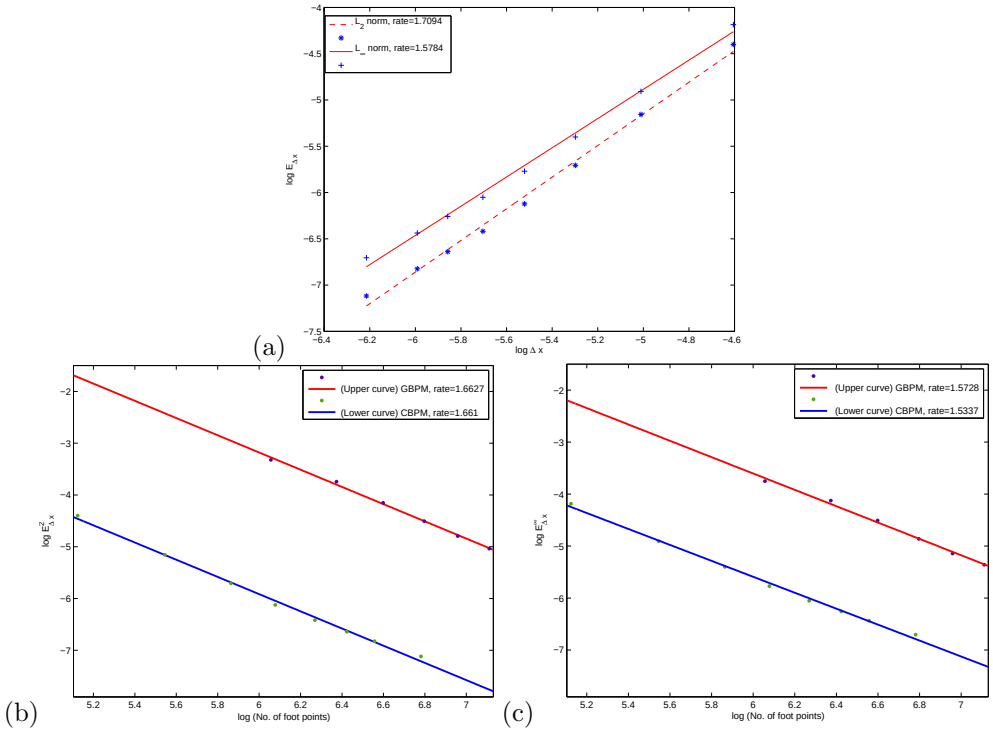


Figure 14: (a) Convergence analysis on the star-shaped interface after rotation with the error measured in the L^2-norm (rate is approximately 1.71) and the L^∞-norm (rate is approximately 1.58). (b) The L^2 error and (c) the L^∞ error against the logarithm of the numbers of foot-points using both CBPM and GBPM. The rates in the last two figures are approximately 1.66 and 1.55, respectively.

5.3 Simple Motions

In this section, we will test our algorithm on several simple motions including flow under a single vortex, motion in the normal direction, and motion by mean curvature.

We first consider the vortex flow as discussed in [6, 12, 25] for inspecting whether a numerical method can be able to resolve very thin interfaces. The initial interface used here is a circle centered at (0.5, 0.75) with radius 0.15. The velocity field $\mathbf{u}=(u,v)$ is given as follows

$$\begin{aligned} u &= -\sin(\pi x)^2 \sin(2\pi y), \\ v &= \sin(\pi y)^2 \sin(2\pi x). \end{aligned} \tag{5.8}$$

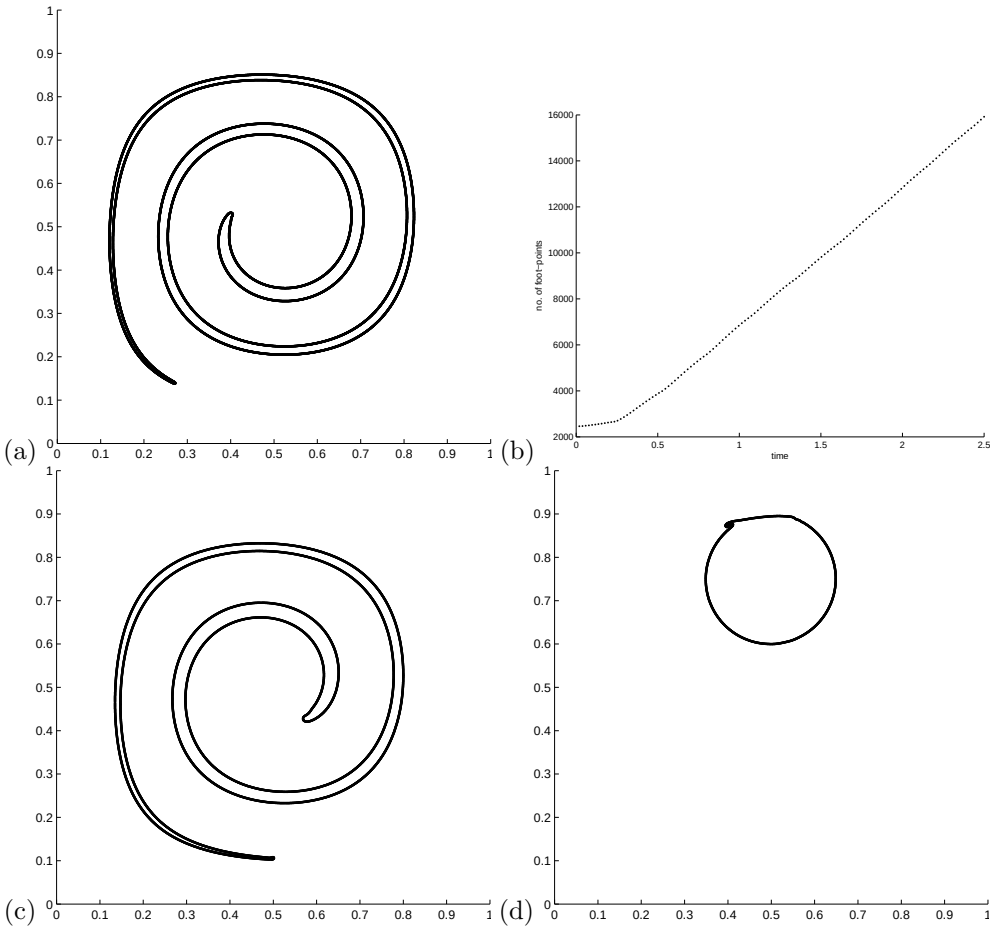
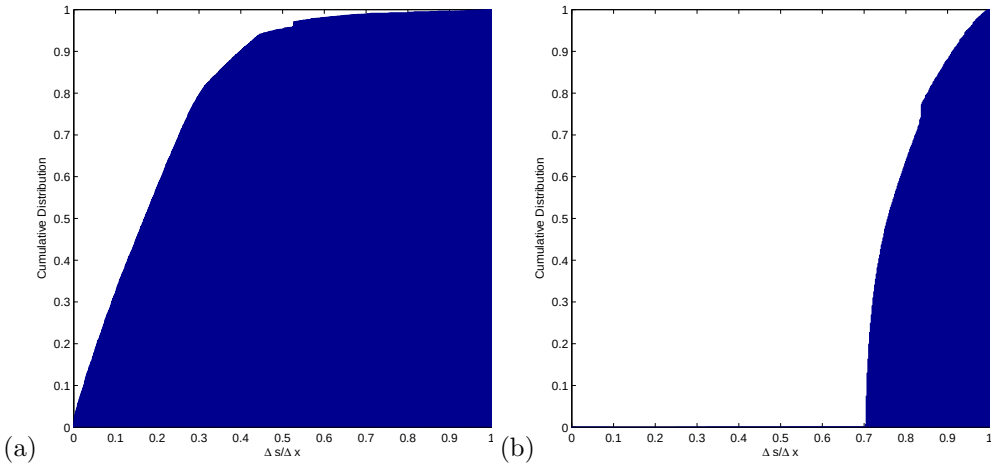


Figure 15: (a) Motion under a single vortex flow (5.8) of a circle at $t=3.0$ with the resolution of 2048^2. (b) The change in the number of sampling points for the flow given by equation (5.8). (c-d) Motion under a single vortex flow with rewind motion of a circle using the resolution 1024^2. Solutions at (c) $t=4$ and $T=8$ and (d) $t=T=8$.

In Figure 15 (a), we plot the interface at time $t=3.0$ with a uniform underlying grid of the resolution of 2048^2. The time step Δt is $0.8\Delta x$. Roughly speaking, the GBPM samples an interface from both sides of the interface while the CBPM gives a representation point only when a cell intersects with the interface. Therefore we expect that the number of footpoints sampled on the interface under our algorithm is much less than that in the GBPM. In this numerical example, we use a finer mesh for the CBPM in order to better resolve the interface at $t=3.0$ under the defined vortex motion so we can have a slightly fairer comparison with the solution by the GBPM. In particular, we have obtained approximately 1.6×10^4 footpoints with the mesh resolution of 2048^2, Figure 15 (b), while there are around 2×10^4 footpoints in the original GBPM with the mesh resolution of 1025^2, as demonstrated in [21]. At the final time, the tail has developed large curvature. The uniform underlying grid may not be able to capture the interface with large curvature and this explains why the tail is not sharp enough and is a bit rounded. Adaptivity can be implemented to remedy this problem. For example, when the curvature detected is large or when

two segments of the interface are getting close, one can locally refine the cells to ensure the ability to capture the detailed features of the solution.

To test mass conservation, we reverse the velocity field by multiplying it by $\cos(\pi t/T)$. The exact location of the interface at $t=T$ should be the same as in the initial condition. Figure 15 (c-d) show the results for different T. In Figure 15 (d), there is a tail near the top of the circle which shows some numerical artifacts. This is due to the small error made in the solution at $t=T/2$ and it will be enormously amplified over time. One might compare our solutions with those in [6, 12, 25].



(a) (b)

Figure 16: Cumulative distributions of the distances between two adjacent particles Δs_i in Figure 15 (a). Solutions obtained by (a) the GBPM with $\Delta x=1/1024$ and in (b) the CBPM with $\Delta x=1/2048$.

In Figure 16, we also compare the distribution of distance between particles of Figure 15 in the CBPM with that in the GBPM. Again, the results show that CBPM tends to have a better sampling distribution.

Now we examine our method by considering motion in a specified normal velocity. As the problem becomes non-linear, we are interested in finding the viscosity solution of the evolution. We will also show that our method has the ability to control the change in the interface topology and can obtain the correct viscosity solution for a star-shaped interface with motions in normal direction.

The interface we used is again the six-folded star shape defined in equation (5.7). As the outward normal speed is equal to one, the interface will grow outward uniformly in time. We follow the CFL condition and numerically choose $\Delta t = 0.5\Delta x$. The interface will self-intersect when the it grows in the normal direction, which corresponds to the shock formation in hyperbolic conservation law. The numerical results are shown in Figure 17 (a-c). In contrast to the usual Lagrangian particle methods where adjacent interfaces will cross each other, we deactivate any active cell points when we detect a conflict in the normal direction in its neighborhood. In this example, we do not need any global parametrization. The normal direction is the only Lagrangian information that we use to enforce the deactivation of cell points. To further examine the self-intersecting parts of the interface, we zoom in the solution and compare our solution with the multivalued solution from typical high order ray tracing method in Figure 17 (d-e).

As a final remark, the multi-valued solution can also be obtained in our algorithm. We can use a global parametrization of the interface as described in Section 3.5. In Figure 18, we showed a case that involves two circular interfaces under motion in the outward normal direction. For this case, we have two disjoint closed interfaces, we parameterize them using two disjoint intervals. When we collect neighboring particles for local reconstruction, we check if their corresponding parameterizations come from the same interval of parametrization. If not, we will split the set of footpoints into groups and locally reconstruct each segment individually. As a result, each circle expands independently without any interference from the other interface.

We also test our algorithm using motion by mean curvature which is widely used in dynamic interface problems. Two types of initial interfaces are given to illustrate the result. The two initial interfaces, both centered at (0.5, 0.5), are a circle with radius 0.15 and the six-folded star-shaped in equation (5.7). Under the motion by mean curvature, interfaces shrink into a circle asymptotically and eventually disappear. In Figure 19 (a), we plot the change in the radius of the circle. The radius of the circle can be analytically

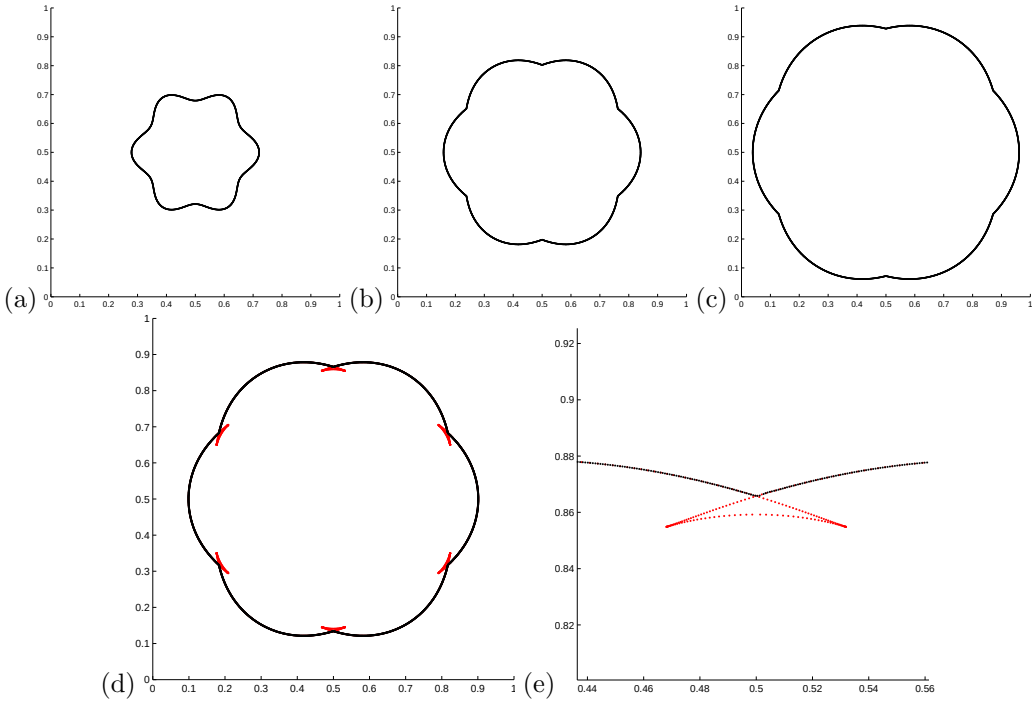


Figure 17: (a-c) Motion in the outward normal direction of a star-shaped interface with the resolution 1024^2 up to t=2.4. (d-e) Comparison between viscosity solution and multivalued solution in the inward normal direction of a star-shaped interface with the resolution 1024^2. The multivalued solution by typical Lagrangian ray tracing is represented by red dots.

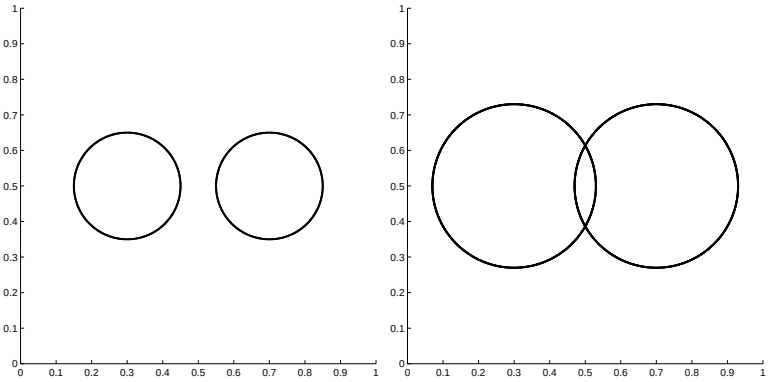


Figure 18: The multivalued solution in the outward normal direction of two circular interfaces with the resolution 512^2.

computed, given by $r(t)=\sqrt{0.15^2-2t}$. We show that the radius computed matches the exact solution (represented by the solid line) well with a relatively coarse mesh resolution 256^2. Using the time step restriction $\Delta t=0.5\Delta x^2$, we then show the evolutions of the star-shaped interface up to $t=1500\Delta t$ in Figure 19 (b) with resolution of 256^2.

In these experiments, we used a time step $\Delta t=O(\Delta x^2)$ for the ease of implementation. However, the constraint in determining Δt is not straightly due to the CFL-stability condition for solving the corresponding partial differential equation in the level set method, but because we require the footpoints staying *near* its original active cell. With the same motion by mean curvature, we also study the size of time step Δt allowed in the CBPM. In figure 20, we show the error in the final solution at $t=0.01125$ with grid size $\Delta x=1/256$ and $1/512$. As we pick a larger time step, error increases and eventually the solution has shown large instability. More importantly, it is found that such instability can be suppressed by increasing the number of sampling points incorporated in the local reconstruction (denoted by m). Even though the overall computational *time* might depend on this parameter, we

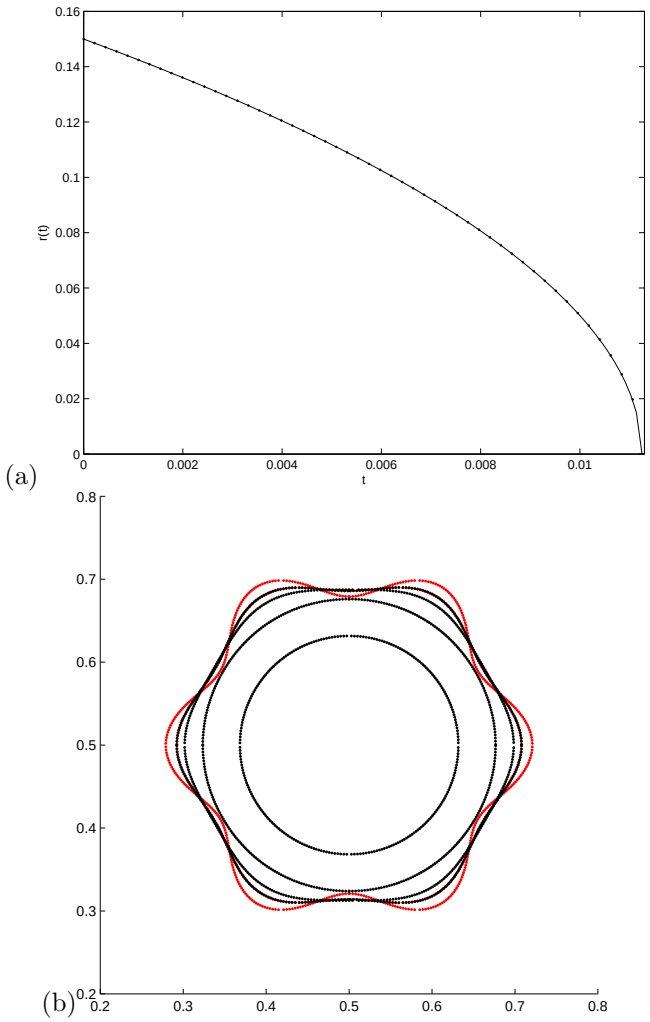


Figure 19: (a) The change in the radius of the circle under motion by mean curvature with the resolution 256^2. The exact solution is represented by the solid line. (b) Evolution of an initial six-folded symmetric interface with $r_0=0.2$ and $\epsilon=\pi/150$ under the motion by mean curvature with resolution 256^2.

remark that the overall computational *complexity* for obtaining the final solution will still be given by $O(N/\Delta t)=O((\Delta x\Delta t)^{-1})=O(\Delta x^{-3})$ where $N=O(\Delta x^{-1})$ is the number of footpoints on the one dimensional interface.

5.4 Motion by surface diffusion and Willmore flow

In this section, we consider some higher order geometric motions in two dimensions including the motion by the surface diffusion and also the Willmore flow. To compute those high order geometric terms, such as the surface Laplacian of the curvature in the surface diffusion flow, we first compute the curvature at footpoints using the geometric basis. Then we approximate the local curvature as a function defined on the interface using least squares fitting. The surface Laplacian is then approximated by differentiating this local polynomial. Figure 21 shows a case where a six-folded star-shaped interface evolves by surface diffusion. We impose a restrictive time step constraint $\Delta t=O(\Delta x^4)$, numerically we picked $\Delta t=0.5\Delta x^4$. Indeed, such a restrictive time step constraint will be impractical for real applications. To relax such a stability condition, one might follow those implicit schemes as demonstrated in [17]. Another example in this section is a circle with radius 0.15 evolving under the Willmore flow given by $v_n=\Delta_\Sigma\kappa-\frac{1}{2}\kappa^3$. Therefore, the circle grows and the analytical solution of its radius is given by $r(t)=(2t+0.15^2)^{1/4}$. In Figure 22, we plot the change in the radius of the circle and showed that the radius computed matches the exact solution extremely well using only a relatively coarse mesh of resolution 128^2.

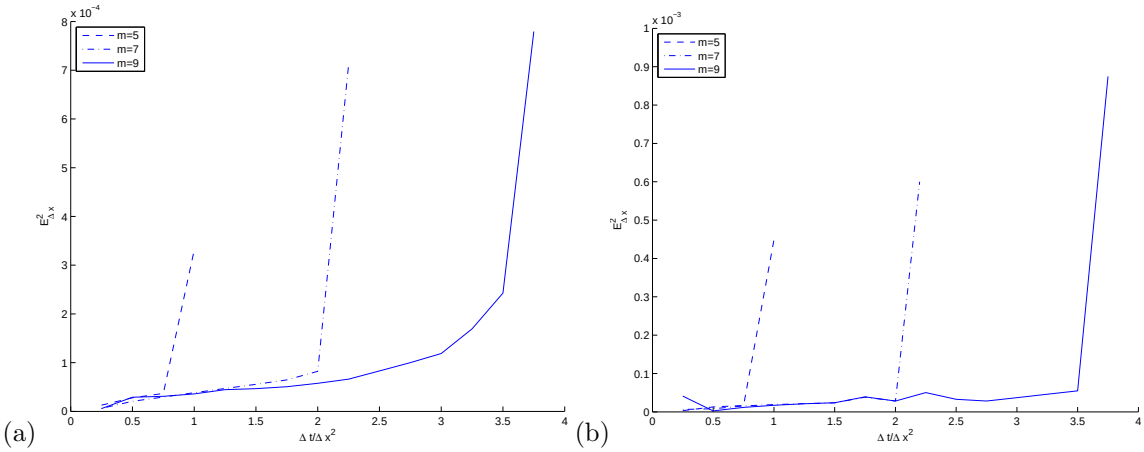


Figure 20: Error study on the star-shaped interface after motion by mean curvature at time $t=0.01125$ with (a) $\Delta x=1/256$ and (b) $\Delta x=1/512$ and m is the number of neighboring particles used in the local reconstruction.

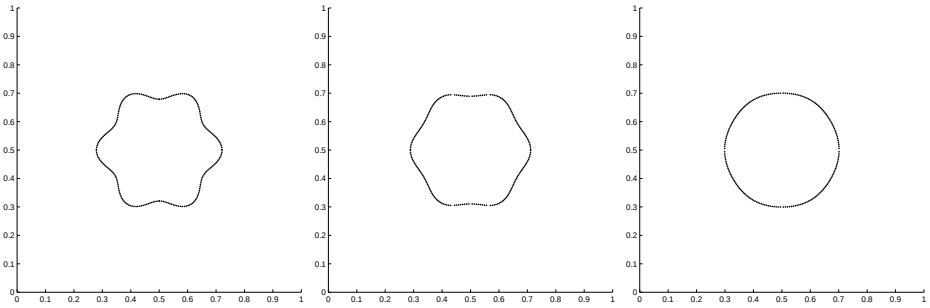


Figure 21: The solution of a six-folded star-shaped interface by the surface diffusion with the resolution 128^2.

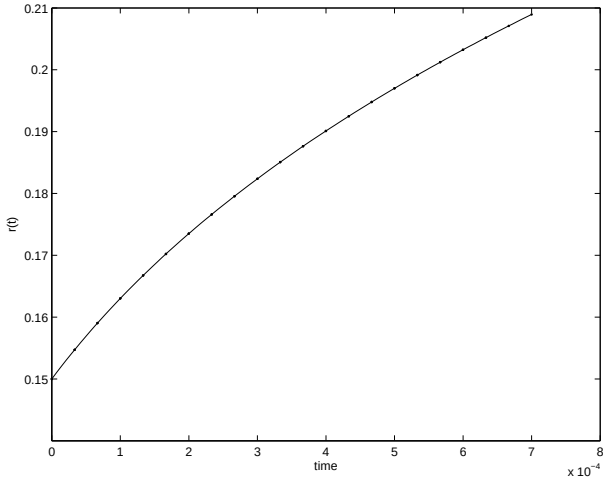


Figure 22: The change in the radius of the circle by the Willmore flow with the resolution 128^2. The exact solution is represented by the solid line.

5.5 Solving PDE on surfaces represented by the CBPM

As discussed in the previous section, we are also able to solve partial differential equations on evolving surfaces. We first consider imposing the local inextensible constraint following the paper [17]. To preserve

the local surface area to the first order, one obtains [16, 29]

$$(u_t)_s - \kappa u_n = 0,$$

where s is the arc-length parametrization, κ is the signed curvature and u_n and u_t are the normal and the tangential velocities, respectively. As in [35], to impose this constraint in the flow induced by the original energy $E^o(\Sigma)$, we introduced a locally defined Lagrange multiplier Λ to obtain

$$E^n(\Sigma) = E^0(\Sigma) + \int_\Sigma \Lambda \cdot (s_\Sigma - s_0) d\Sigma.$$

Minimizing this new energy, one solves the following Helmholtz equation for the Lagrange multiplier Λ

$$-\Lambda_{ss} + \kappa^2 \Lambda = (u_t^o)_s - \kappa u_n^o. \tag{5.9}$$

The numerical approach to solve this equation is similar to the one presented in [17]. To solve equation (5.9), we locally approximate Λ using a local square quadratic polynomials, namely $\Lambda(\theta) = a_0 + a_1\theta + a_2\theta^2$. Given the sampling points $(\theta_i, \Lambda(\theta_i) = \Lambda_i)$ for $i = 1, \cdots, m$, a_2 can be determined by finding the least squares solution of the following system using the SVD

$$\begin{pmatrix} 1 & \theta_1 & \theta_1^2 \\ 1 & \theta_2 & \theta_2^2 \\ \vdots & \vdots & \vdots \\ 1 & \theta_m & \theta_m^2 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \end{pmatrix} = \begin{pmatrix} \Lambda_1 \\ \Lambda_2 \\ \vdots \\ \Lambda_m \end{pmatrix} \tag{5.10}$$

Also, as mentioned in Section 4.5, we can compute the surface Laplacian in the polar form centered at the least square circle as $\Delta_\Sigma \Lambda = \frac{2a_2}{r_c^2}$. Therefore, equation (5.9) can be approximated as

$$-\frac{2a_2}{r_c^2} + \kappa^2 \Lambda = (u_t^o)_s - \kappa u_n^o. \tag{5.11}$$

By considering (5.10) with all footpoints on the interface, we can solve the linear system arising from (5.11) using the SVD. Once we have the solution, we update the interface using

$$u_n = u_n^o + \Lambda\kappa.$$

Lets consider a circle with radius 0.15 evolving under the motion by mean curvature. In this case, we have $u_n^0 = -\kappa$ and $u_t^0 = 0$, so the arc length growing with time is given by $2\pi\sqrt{0.15^2 - 2t}$. Solving the Helmholtz equations, we obtain $\Lambda = 1$ and then $\mathbf{u} = \mathbf{0}$. Therefore, the interface should remain unchanged as a circle. In Figure 23(a), we showed the computed arc-length of the circle with and without the local in-extensible constraint and the result follows.

Next, we consider the case in which the same circle is evolved by motion in the normal direction with $u_n^0 = 1$ and $u_t^0 = 0$. The arc length grows linearly with time and is given by $2\pi(0.15 + t)$. In Figure 23(b), we show the computed arc-length of the circle with and without the local in-extensible constraint. Similar to the previous case, the circle remains unchanged with the in-extensible constraint.

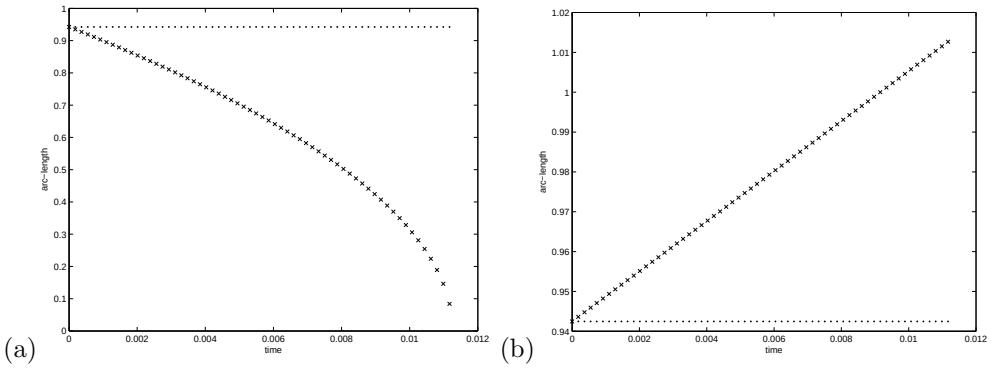
Next, we consider a four-folded interface initial profile

$$r = 0.25 + 0.075\cos 4\theta. \tag{5.12}$$

It evolves by Willmore flow with the local in-extensible constraint. In Figure 24 (a) and (b), we plot the evolution of the interface without and with the constraint, respectively.

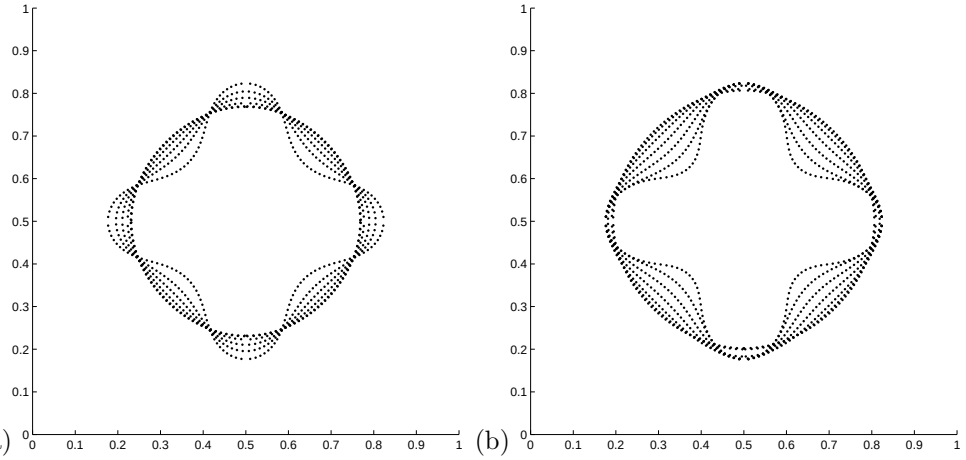
5.6 Surface integrals

As discussed, one highlight of our proposed algorithm is the natural decomposition of the interface so that surface integrals can be easily approximated. In this section, we demonstrate various numerical examples involving integrals on surfaces. In particular, we follow the discussion in Section 3.7 to impose a volume preserving constraint in various flows. In two dimensions, the enclosed area of a close interface will remain unchanged with the area preserving constraint imposed. We consider again a six-folded star-shaped defined in equation (5.7) interface evolving under the motion by mean curvature. Here, we now further impose the volume preserving constraint by computing two integrals, as discussed in Section 3.4.



(a) (b)

Figure 23: The change in the arc-length of the circle evolving under different flows and the in-extensible analogues with resolution 256^2. (a) Motion by mean curvature and (b) motion in the normal direction. Crosses represent the arc-length without constraint while the dots represent those with constraint.



(a) (b)

Figure 24: Evolution of an initial four-folded symmetric interface with $r_0=0.25$ and $\epsilon=0.30$ under the Willmore flow (a) without and (b) with the local inextensible constraint with resolution 128^2.

In this first example, we consider a circle centered at $(x_0,y_0)=(0.5,0.5)$ with radius $r_0=0.15$ under the shear flow $\mathbf{u}=(u_x,u_y)$ given as follow

$$\begin{aligned} u_x &= x-x_0+\gamma(y-y_0) \\ u_y &= y-y_0\,, \end{aligned}$$

where $\gamma=4$. Numerically we picked $\Delta t=0.2\Delta x$ and the evolution stops at $t=0.3906$. Figure 25 (a) shows the evolution of the interface under the flow without the volume preserving constraint and Figure 25 (b) shows the interfaces with and without the constraint imposed at the final time with the same resolution; Figure 25 (c) finally shows the change of enclosed area against time with and without the constraint. To compute the enclosed volume, we apply the divergence theorem

$$\int_V (\nabla\cdot\mathbf{F})dV=\oint_\Sigma (\mathbf{F}\cdot\mathbf{n})ds.$$

Let $\mathbf{F}=(x,y)$ in the two dimensional case, we can have the following approximation to the enclosed volume

$$\frac{1}{2}\oint_\Sigma (\mathbf{F}\cdot\mathbf{n})ds\approx\frac{1}{2}\sum(xn^1+yn^2)\Delta s,$$

where the normal vectors and arc length segments are all defined on the cell i. The approach can also be generalized to three dimensional cases.

In the next example, we consider the same six-folded star-shaped interface defined in equation (5.7) under the motion in the normal direction. As shown in Figure 17 (a-c), the enclosed area of the interface

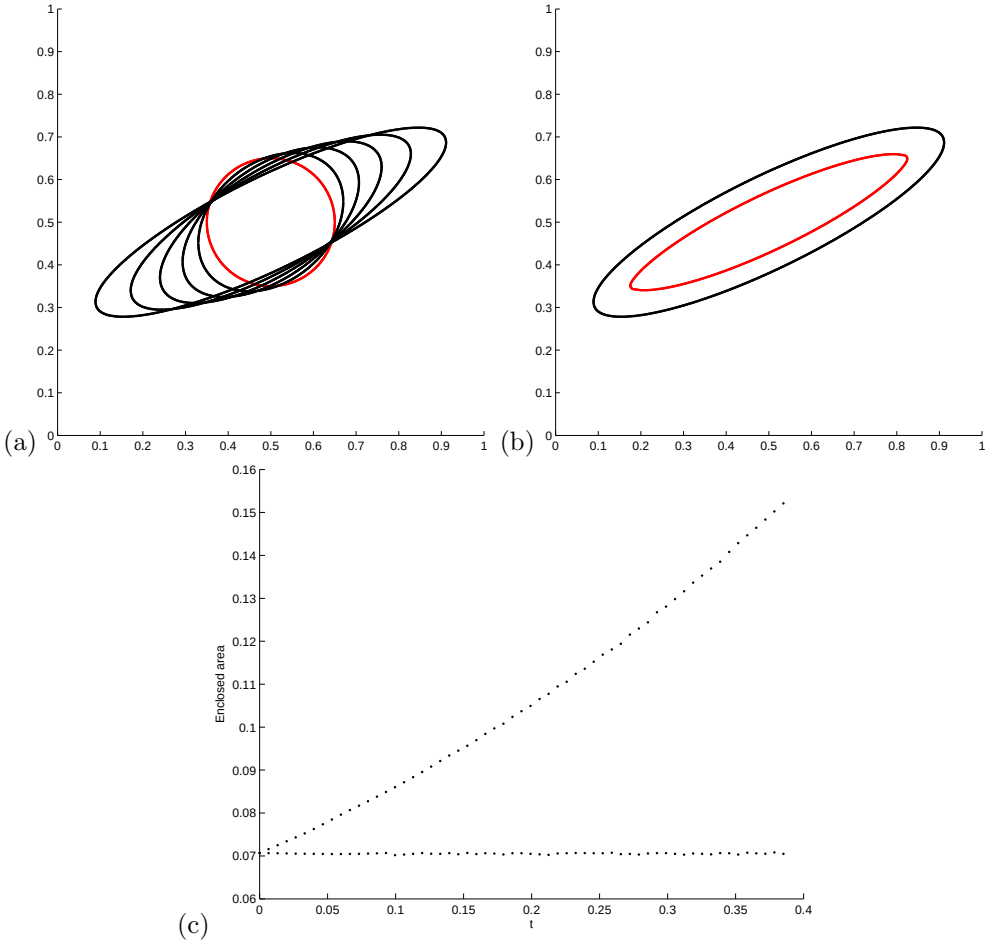


Figure 25: (a) Evolution of an initial circle centered at $(0.5,0.5)$ with $r_0=0.2$ under the shear flow without the volume preserving constraint with resolution 512^2. (b) Evolution of the same initial circle under the same shear flow with the volume preserving constraint (inner curve). The solution without the constraint is also plotted for comparison (outer curve). (c) Enclosed area against time time with and without the constraint imposed with the same resolution.

grows with time and the velocity field does not satisfy the volume preserving constraint. In Figure 26, we showed the change of the enclosed area of the interface over time with and without the constraint. Again, the results showed that the enclosed area of the interface is well-preserved over time with the constraint imposed.

Finally, in Figure 27, we consider the evolution of a six-folded star-shape in equation (5.7) under the motion by mean curvature. Unlike the case without such constraint, Figure 19 (b), the interface with the constraint does not shrink asymptotically into a circle and disappear. Instead, the interface evolves into a circle with a constant enclosed area and remains unchanged in time.

5.7 Three dimensional examples

In this section, we consider several examples in three dimensions. The first example is a sphere centered $(x_0,y_0,z_0)=(0.35,0.35,0.35)$ with radius 0.15 under a single vortex flow $\mathbf{u}=(u_1,u_2,u_3)$. The flow is given by

$$\begin{aligned} u_1(x,y,z) &= 2\sin^2(\pi x)\sin(2\pi y)\sin(2\pi z),\\ u_2(x,y,z) &= -\sin(2\pi x)\sin^2(\pi y)\sin(2\pi z),\\ u_3(x,y,z) &= -\sin(2\pi x)\sin(2\pi y)\sin^2(\pi z). \end{aligned}$$

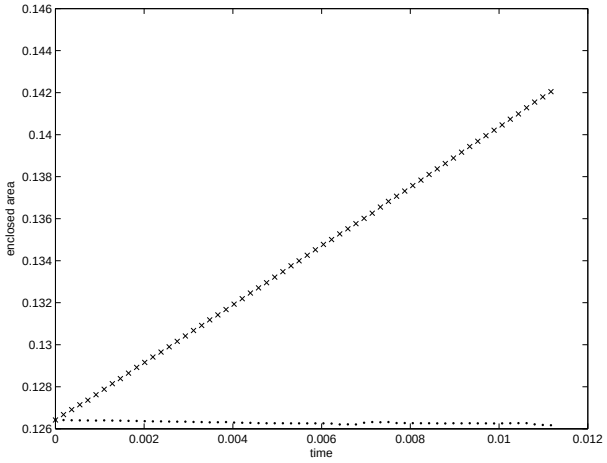


Figure 26: Change of the enclosed area of a six-folded star-shaped interface under motion in normal direction with volume preserving constraint with the resolution 256^2. Crosses represent the arc-length without constraint while the dots represent those with constraint.

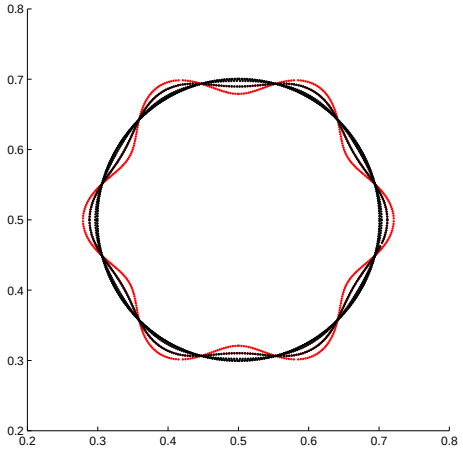


Figure 27: Evolution of an initial six-folded symmetric interface with $r_0=0.2$ and $\epsilon=\pi/150$ under the motion by mean curvature with the volume preserving constraint with resolution 256^2.

In Figure 28, we show the evolutions at $t=0.15,0.3$ and 0.45 with uniform resolution 128^3. Again, because of the less sampling rate of the CBPM comparing to GBPM, the later evolutions with high curvature segments can not be captured with resolution 128^3 and it can be improved by local adaptivity.

Another example is the shear flow of a perturbed sphere centered at $(x_0,y_0,z_0)=(0.5,0.5,0.5)$ with radius r perturbed by the real spherical harmonic $Y_4^2(\theta,\phi)$

$$r=r_0+0.1Y_4^2(\theta,\phi)=r_0+0.1\times\frac{3}{8}\sqrt{\frac{5}{\pi}}\cos(2\phi)\sin^2\theta(7\cos^2\theta-1),$$

with $r_0=0.3$. with shear flow $\mathbf{u}=(u_x,u_y,u_z)$ defined as follow

$$\begin{aligned} u_x &= x-x_0\,,\\ u_y &= y-y_0,,\\ u_z &= z-z_0+\gamma(y-y_0)\,, \end{aligned}$$

where $\gamma=4$.

Figure 29 shows the evolution of the perturbed sphere under the shear flow with resolution 256^3 while Figure 30 shows the change of enclosed volume with and without the volume preserving constraint imposed.

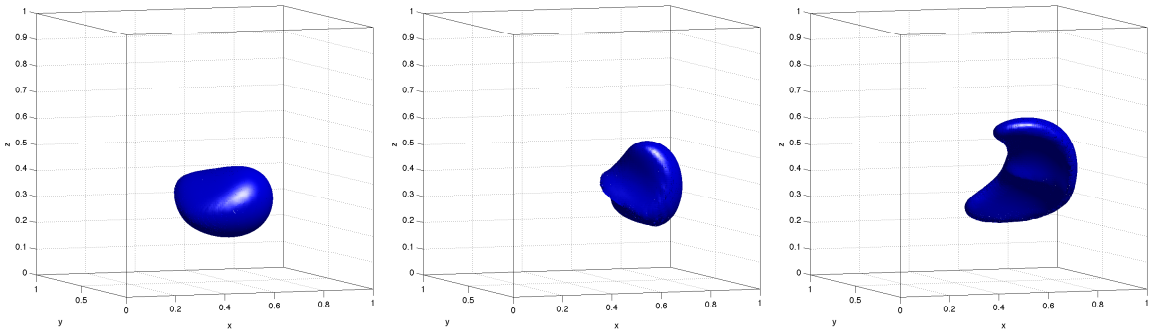


Figure 28: The evolution of a sphere under the vortex flow with resolution 128^3.

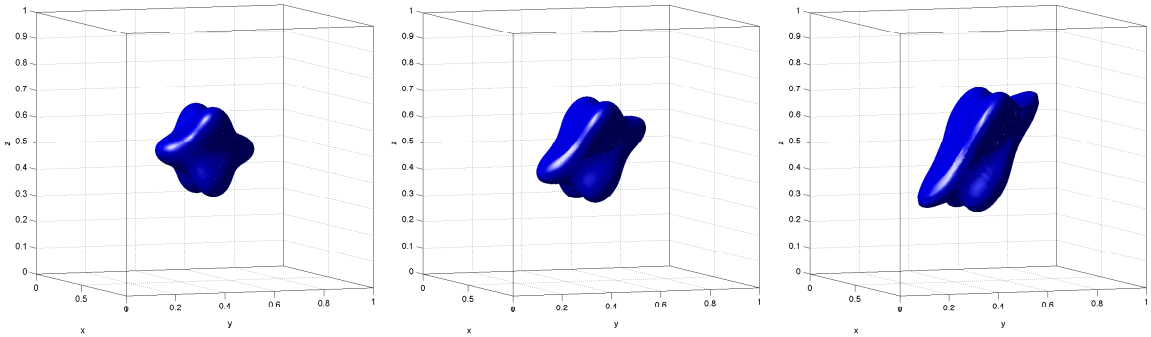


Figure 29: The evolution of the perturbed sphere under the shear flow with resolution 256^3.

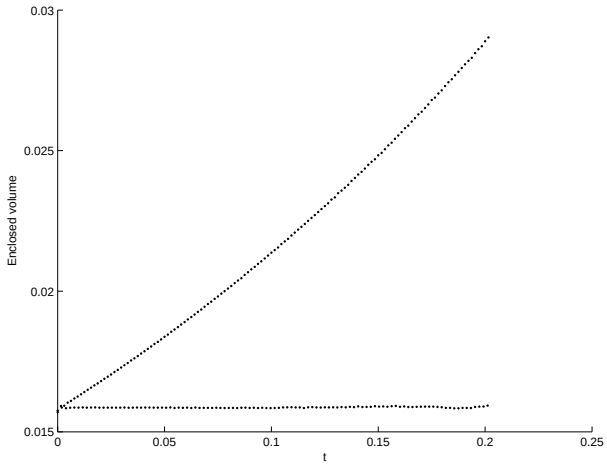


Figure 30: The change in the enclosed volume of the interface with and without the volume preserving constraint imposed of the same resolution.

We also consider the motion by mean curvature of the same perturbed sphere with $r_0 = 0.15$. Figures 31 (a) shows the evolution of the interface up to $t = 6.1035 \times 10^{-3}$, where the underlying grid has the resolution 256^3. The perturbed sphere gradually shrinks into a smaller sphere and disappear eventually. With the volume-preserved constraint imposed, the final solution is a sphere with a radius such that the enclosed volume is preserved, figures 31 (b). Figures 31 (c) shows the evolution of the same interface with constraint imposed and the change in the enclosed volume of the interface under the same flow. The interface becomes a sphere and remains nearly the same volume over time.

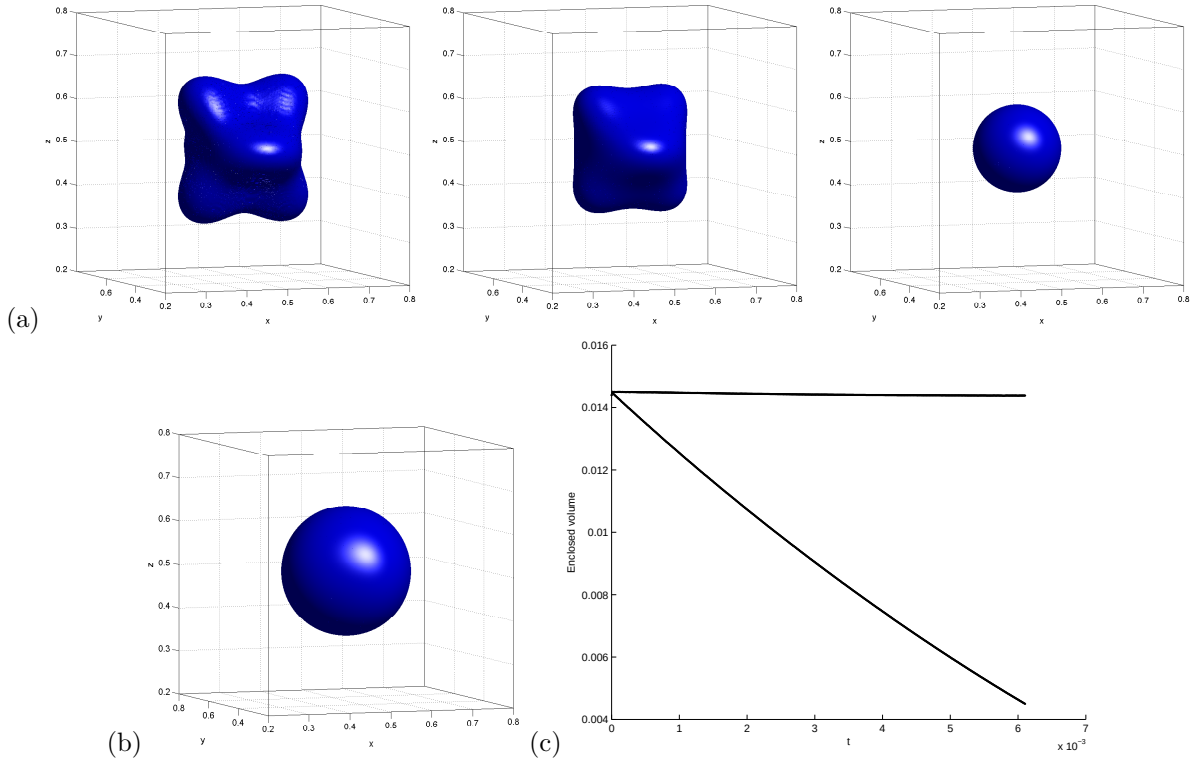


Figure 31: (a) Evolution of the sphere perturbed by a spherical harmonic under the motion by mean curvature with resolution 256^3. (b) Evolution of the same initial interface under the same shear flow at final time but with the volume preserving constraint. (c) The change in the enclosed volume with and without the volume preserving constraint.

6 Conclusion

We have proposed the cell based particle method (CBPM) which introduces several modifications to the original grid based particle method (GBPM) for moving interface problems [21]. They include

1. a change in the spatial discretization scheme from the grid-based to cell-based,
2. activating only those cells which intersects with the interface, and
3. a geometric basis for local reconstruction.

Similar to the GBPM, the interface in the CBPM is also represented and is tracked using quasi-uniform meshless particles. These particles are also sampled according to an underlying grid such that each particle is associated to a grid point which is in the neighborhood of the interface. The underlying grid provides an Eulerian reference and local sampling rate for particles on the interface. The CBPM also renders neighborhood information among the meshless particles for local reconstruction of the interface. The resulting algorithm, which is based on Lagrangian tracking using meshless particles with Eulerian reference grid, can naturally handle/control topological changes.

Comparing to the GBPM, the CBPM reduces the number points on the interface since GBPM has to sampling from both sides. Having said that, because the representation and the evolution of particles are

similar for these two approaches, these two algorithms have the same computational complexity given by $O(N/\Delta t)$ where N is the number of sampling points on the interface.

In addition to all advantages in the original GBPM, the CBPM has several new nice features. Because of the cell-based representation, the method can easily approximate surface integrals using segments. To determine if a cell intersects with the interface, we propose to replace the L^2-projection by the L^∞-projection. Numerically, we have demonstrated that this change also provides a better distribution of sampling particles on the surface. Even though one can still keep to use the local polynomial reconstruction as in the original GBPM, we have studied an alternative by the geometric basis which can further simplify the overall implementation of the algorithm.

Future work includes the modeling of Peach-Koehler force in dislocation dynamics based on some boundary integral formulation and simulating the dynamics of lipid vesicles. Moreover, even though the CBPM can easily approximate a surface integral, the computational complexity could be high if the evolution at each sampling point depends explicitly on a different surface integral. Since the complexity for evaluating a surface integral is $O(N)$ where N is the number of sampling points on the interface, the total computational complexity for evolving the interface for one time step will be $O(N^2)$. Such complexity will be extremely challenging for real applications in three dimensional space. One possible improvement is to incorporate the fast multipole method [11] for fast implementations.

Acknowledgment

Leung acknowledges the hospitality of the University of California, Irvine where preliminary work was performed. The work of Leung was supported in part by the Hong Kong RGC under grant 602210. Research of Zhao is partially supported by NSF grant DMS-1115698.

References

[1] S.M. Allen and J.W. Cahn. A microscope theory for antiphase boundary motion and its application to antiphase domain coarsening. *Acta Metall.*, 27:1085–1095, 1979.

[2] J. Bedrossian, J.H. von Brecht, S. Zhu, E. Sifakis, and J.M. Teran. A second order virtual node method for poisson interface problems on irregular domains. *Submitted to Journal of Computational Physics*, 2009.

[3] C.A. Brebbia and L.C. Wrobel. The boundary element method. *Computer methods in fluids (A 81-28303 11-23)*, pages 26–48, 1980.

[4] J. Cabrera and P. Meer. Unbiased estimation of ellipses by boorstrapping. *IEEE Trans. on Pattern Anal. Mach. Intell.*, 18(7):752–756, 1996.

[5] J.W. Cahn and J.E. Hilliard. Free energy of a nonuniform system. i. interfacial free energy. *Journal of Chemical Physics*, 28:258–267, 1958.

[6] 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.

[7] A. Fitzgibbon, M. Pilu, and R.B. Fisher. Direct least square fitting of ellipses. *IEEE Trans. on Pattern Anal. Mach. Intell.*, 21(5):476–480, 1999.

[8] W. Gander, G.H. Golub, and R. Strebel. Least squares fitting of circles and ellipses. *BIT Numerical Mathematics*, 34:558–578, 1994.

[9] J. Glimm, J. Grove, X.L. Li, and D.C. Tan. Robust computational algorithms for dynamic interface tracking in three dimensions. *SIAM J. Sci. Comput.*, 21:2240–2256, 2000.

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

[11] L. Greengard and V. Rokhlin. A fast algorithm for particle simulations. *J. Comp. Phys.*, 135(2):280–292, 1997.

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

[13] C.W. Hirt and B.D. Nichols. Volume of fluid /VOF/ method for the dynamics of free boundaries. *J. Comput. Phys.*, 39:201–225, 1981.

[14] T.Y. Hou, J.S. Lowengrub, and M.J. Shelley. Boundary integral methods for multicomponent fluids and multiphase materials. *J. Comput. Phys.*, 169:302–362, 2001.

[15] K. Kanatani. Statistical bias of conic fitting and renormalization. *IEEE Trans. on Pattern Anal. Mach. Intell.*, 16(3):320–326, 1994.

[16] D.Y. Kwon and F.C. Park. Evolution of inelastic plane curves. *Applied Mathematics Letters*, 12:115–119, 1999.

[17] S. Leung, J. Lowengrub, and H.K. Zhao. A grid based particle method for high order geometrical motions and local inextensible flows. *J. Comput. Phys.*, 230:2540–2561, 2011.

[18] S. Leung and J. Qian. Eulerian Gaussian beams for Schrödinger equations in the semi-classical regime. *J. Comput. Phys.*, 228:2951–2977, 2009.

[19] S. Leung, J. Qian, and R. Burridge. Eulerian Gaussian beams for high frequency wave propagation. *Geophysics*, 72:SM61–SM76, 2007.

[20] S. Leung and H.K. Zhao. A grid-based particle method for evolution of open curves and surfaces. *J. Comput. Phys.*, 228:7706–7728, 2009.

[21] S. Leung and H.K. Zhao. A grid based particle method for moving interface problems. *J. Comput. Phys.*, 228:2993–3024, 2009.

[22] S. Leung and H.K. Zhao. Gaussian beam summation for diffraction in inhomogeneous media based on the grid based particle method. *Communications in Computational Physics*, 8:758–796, 2010.

[23] J. Liu and S. Leung. A splitting algorithm for image segmentation on manifolds represented by the grid based particle method. *J. Sci. Comput.*, DOI 10.1007/s10915-012-9675-7, 2013.

[24] I. Markovsky, A. Kukkush, and S. Van Huffel. Consistent least squares fitting of ellipsoids. *Numer. Math.*, 98:177–194, 2004.

[25] C. Min and F. Gibou. A second order accurate level set method on non-graded adaptive cartesian grids. *J. Comput. Phys.*, 225:300–321, 2007.

[26] Y. Nievergelt. Hyperspheres and hyperplanes fitted seamlessly by algebraic constrained total least-squares. *Linear Algebra and its Applications*, 331:43–59, 2001.

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

[28] C. Pozrikidis. Interfacial dynamics for stokes flow. *J. Comput. Phys.*, 169:250–301, 2001.

[29] R. Rosso, A.M. Sonnet, and E.G. Virga. Evolution of vesicles subject to adhesion. *Proc. R. Soc. Lond. A*, 456:1523–1545, 2000.

[30] S.J. Ruuth, B. Merriman, and S. Osher. A fixed grid method for capturing the motion of self-intersecting wavefronts and related PDEs. *J. Comput. Phys.*, 163:1–21, 2000.

[31] D. Salac and M. Miksis. A level set projection model of lipid vesicles in general flows. *J. Comp. Phys.*, 230:8192–8215, 2011.

[32] S. Shin, S.I. Abedel-Khalik, V. Daru, and D. Juric. Accurate representation of surface tension using the level contour reconstruction method. *J. Comput. Phys.*, 203:493–516, 2005.

[33] S. Shin and D. Juric. Modeling three-dimensional multiphase flow using a level contour reconstruction method for front tracking without connectivity. *J. Comput. Phys.*, 180:427–470, 2002.

[34] K. Siddiqi, B.B. Kimia, and C.-W. Shu. Geometric shock-capturing ENO schemes for subpixel interpolation, computation and curve evolution. *Graphical Models and Image Processing*, 59:278–301, 1997.

[35] J.S. Sohn, Y.H. Tseng, S.W. Li, A. Voigt, and J. Lowengrub. Dynamics of multicomponent vesicles in a viscous fluid. *J. Comput. Phys.*, 229:119–144, 2010.

[36] J. Steinhoff, M. Fan, and L. Wang. A new Eulerian method for the computation of propagating short acoustic and electromagnetic pulses. *J. Comput. Phys.*, 157:683–706, 2000.

[37] 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.

[38] D. A. Turner, I. J. Anderson, J. C. Mason, and M. G. Cox. An algorithm for fitting an ellipsoid to data. *unpublished manuscript (available from http://citeseer.nj.nec.com/322908.html)*, 1999.

[39] S.K. Veerapaneni, D. Gueyffier, D. Zorin, and G. Biros. A boundary integral method for simulating the dynamics of inextensible vesicles suspended in a viscous fluid in 2d. *J. Comput. Phys.*, 228:2334–2353, 2009.

[40] G. Ventura, J.X. Xu, and T. Belytschoko. A vector level set method and new discontinuity approximations for crack growth by EFG. *International Journal for Numerical Methods in Engineering*, 54:923–944, 2002.

[41] D. Womble. A front-tracking method for multiphase free boundary problems. *SIAM J. Num. Anal.*, 26:380–396, 1989.