A Grid Based Particle Method for Moving Interface Problems

Shingyu Leung* Hongkai Zhao†

February 29, 2008

Abstract

We propose a novel algorithm for modeling interface motions. The interface is represented and is tracked using quasi-uniform meshless particles. These particles are 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 the particles on the interface. It 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. Moreover, adaptive sampling of the interface can be achieved easily through local grid refinement with simple quad/oct-tree data structure. Extensive numerical examples are presented to demonstrate the capability of our new algorithm.

Keywords: Interfacial flow; Interface tracking; Eulerian underlining mesh; Lagrangian sampling particles.
MSC: 65D17, 76M25, 78A05.

*Corresponding author. Tel.: (949) 824-3087. Fax: (949)-824-7993. Department of Mathematics, University of California at Irvine, Irvine, CA 92697-3875. Email: **syleung@math.uci.edu**

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

1 Notation

$\mathbf{x} = (x_1, \cdots, x_d) \in \mathbb{R}^d$	Coordinates of a grid point.
γ	Local tube radius.
Γ	Set of all active grids.
$\mathbf{p} = (p_1, \cdots, p_d)$	Coordinates of an active grid point, i.e. $\mathbf{p} \in \Gamma$.
$\mathbf{y} = (y_1, \cdots, y_d)$	Coordinates of a foot-points on the interface.
$\mathbf{u}(\mathbf{y}) = (u_1(\mathbf{y}), \cdots, u_d(\mathbf{y}))$	Velocity field at the foot-point located at $\mathbf{y}$.
$\mathbf{n}(\mathbf{y}) = (n_1(\mathbf{y}), \cdots, n_d(\mathbf{y}))$	Normal at the foot-point located at $\mathbf{y}$.
$\kappa(\mathbf{y})$	Curvature at the foot-point located at $\mathbf{y}$.
h	Grid size of the cell centered at $\mathbf{x}$ at the coarsest level.
$\mathbf{f}(s)$	A local approximation of the interface.
$g(\mathbf{y})$	A global parametrization of the interface.
m	Number of points used in the local reconstruction.
n	Degree of the polynomial used in the local reconstruction.
$\Omega(\mathbf{p})$	A small neighborhood of $\mathbf{p}$.
L	Level of adaptivity.
$h_{i,j}$	Local grid size.

2 Introduction

Modeling motion of interface and free boundary is an important task in many multiphase problems in science and engineering. Typically, there are two main numerical challenges: (1) representing and tracking a manifold with complicated geometry and dynamics that can develop large deformation, singularities, and even topological changes, (2) coupling between the moving interface and the global dynamics. In this paper we will focus on the first issue and develop a new grid based particle method for representing and tracking a moving interface. The second issue will be reported in our later work.

In general we can classify existing numerical methods for moving interface problems into two categories. The first category is Lagrangian tracking methods, in which the interface is explicitly represented by Lagrangian particles (markers) and its dynamics is tracked by the motion of these particles. Usually these markers are ordered or are connected through either some parametrization or meshes of the interface. For examples, boundary integral methods, boundary element methods, front tracking method and etc belong to this type. Main advantages of the tracking methods are (1) efficient and accurate representation of the interface, (2) simplicity in tracking the motion of particles. However it is in general difficult (especially in high dimensions) to maintain a nice explicit parametrization/surface meshes of the interface with complicated geometry and dynamics involving large deformations or

topological changes, such as merging or splitting of surface. The resulting algorithm usually requires reparametrization/remeshing constantly during the evolution as well as complicated reconnection through surgery when topology changes.

The second category 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. For examples, level set methods [13], phase field method [4, 1], volume of fluid method [9], and etc belong to this type. Main advantages of capturing methods are (1) a geometric problem is turned into a partial differential equation (PDE) problem (no parametrization or surface mesh is needed) on a fixed grid with simple data structure, (2) topological changes, such as merging or splitting, can be handled easily in the viscosity sense. However, there are also a few disadvantages for these Eulerian approaches. For examples, due to the implicit representation, capturing methods are usually less accurate and less efficient than tracking method in terms of both interface representation and evolution. Excessive numerical dissipation might be introduced in the computations. Extra effort is needed to determine the interface explicitly. Also solving PDEs based on Cartesian grid makes grid refinement and adaptivity complicated.

Another difficulty for most of the above numerical methods is how to control topological change according to the physics. For example, when two wave fronts meet, they cross each other without interfering. On the other hand, when two burning fronts meet or when two bubbles collide, they merge. Lagrangian tracking methods can easily model the motion of interface passing through each other since particles on the interface are connected locally through parametrization or surface meshes. When two different parts of the interface get close in physical space, they do not feel each other since they are far apart in the parameter space or in the surface meshes. These two segments move independently and pass through each other. However, merging or splitting is a major difficulty for the Lagrangian tracking method as explained above. On the contrary, Eulerian capturing methods embeds the interface in a single valued scalar function. So topological changes such as merging or splitting can be handled easily while interface crossing is difficult to deal with for most capturing methods.

In this paper we propose a new framework to model interface motion which naturally combines and takes advantages of the Lagrangian and the Eulerian 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. Here are a few key features for our method.

1. The interface is represented by particles without mesh or parametrization. This feature allows one to easily add or delete particles, which is important for maintaining a consistent resolution of the interface as well as dealing with topological changes.

2. The sampling of the particles has a one-to-one reference to the underlying grid points which are in the neighborhood of the interface. This Eulerian reference provides both a

quasi-uniform sampling of the interface and neighborhood information among meshless particles. These information is useful for local construction of the interface and collision detection of different parts of the interface. The reference is updated with the evolution with no PDE involved on the underlying grid.

3. Adaptive sampling of the interface can be achieved easily through local grid refinement of the underlying grid since no PDE is solved on the grid.

4. Topological change can be controlled using both Lagragian and Eulerian information available.

In the past, various authors have proposed several ideas to combine Lagrangian and Eulerian approaches for moving interface problems. Grid based tracking method proposed in [7] tracks the interface using Lagrangian particles and monitors the topological changes using the underlying grid. Particle level set method proposed in [6] uses the level set method with auxiliary particles to capture as well as track the interface motion. Because these particles have better resolution of the interface, they are used to reconstruct the level set function after each time-step. All these methods are different from ours in which we use meshless particles to represent and track the interface, and we do not solve any PDE on the underlying grid.

We will describe our formulation in details in Section 3 and then give various examples in two-dimensions and three-dimensions in Section 4.

3 Grid Based Particle Method

3.1 Algorithm

In this subsection, we first outline the general algorithm. A detailed description of each step will be given in later subsections.

Algorithm:

1. Initialization [Figure 1 (a)]. Collect all grid points in a small neighborhood (computational tube) of the interface. From each of these grid points, compute the closest point on the interface. We call these grid points **active** and their corresponding particles on the interface **foot-point**.

2. Motion [Figure 1 (b)]. Move all foot-points according to a given motion law.

3. Re-Sampling [Figure 1 (c)]. For each active grid point, re-compute the closest point to the interface reconstructed locally by those particles after the motion in step 2.

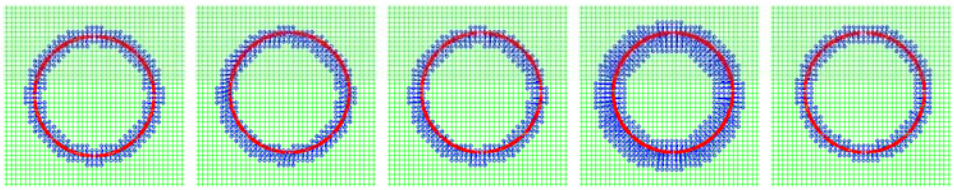

Figure 1: Grid Based Particle Method. From left to right: (a) Initialization, (b) after motion, (c) after re-sampling, (d) after activating new grid points with their foot-points, (e) after inactivating grid points with their foot points. Blue circles denote active grid points and red squares denote their corresponding foot-points on the interface.

4. Updating the computational tube [Figure 1 (d-e)]. Activate any grid point with an active neighboring grid point and find their corresponding foot-points. Then, inactivate grid points which are far away from the interface.

5. Adaptation (Optional). Locally refine the underlying grid cell if necessary.

6. Iteration. Repeat steps 2-5 until the final computational time.

3.2 Initialization and Motion

In this paper, we represent the interface by meshless particles which are associated to an underlying Eulerian mesh. In our current algorithm, 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 give each particle an Eulerian reference during the evolution. The closest point to a grid point, $\mathbf{x}$, and the corresponding shortest distance can be found in different ways depending on the form in which the interface is given.

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**, γ, 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 **foot-point** associated to this active grid point. This link between the active grid points and its foot-points is kept during the evolution. Furthermore, we can also compute and store certain Lagrangian information of the interface at the foot-points, including normal, curvature and parametrization, which will be useful in various applications.

As a result of the interface sampling, the density of particles on the interface will be roughly inversely proportional to the local grid size. This relation provides an easy adaptive

approach in the current grid based particle method. In some regions where one wants to resolve the interface better by putting more marker particles, one might simply locally refine the underlying Eulerian grid and add the new foot-points accordingly. We will further explain this process in details in Section 3.5.

This initial set-up is illustrated in figure 1. We plot the underlying mesh in solid line, all active grids using small circles and their associated foot-points using squares. On the left most sub-figure, we show the initial set-up after the initialization. To each grid point near the interface (blue circles), we associate a foot-point 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. One 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 which will be discussed in Section 3.3. The velocity may also depend on some global quantity, such as pressure, which has to be determined through a global PDE with an interface condition. This situation will be addressed in later work.

It should be noted that a foot-point $\mathbf{y}$ 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 foot-points to become unevenly distributed along the interface. This may introduce both stiffness, when particles are getting together, and large error, when particles are getting apart. To maintain a quasi-uniform distribution of particles, we need to resample the interface by recomputing the foot-points 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 foot-points on the interface. We will further discuss in details these procedures in the Section 3.3.

One important issue in time evolution is time step. The step size is constrained by the stability condition, which is governed by the underlying PDE. For hyperbolic type of problem for instance, the time step Δt should be of the order $O(h)$ where h is the local grid size. For motion by mean curvature, the PDE is of degenerate parabolic type. This means the time step Δt should be of the order $O(h^2)$. In our numerical algorithm, we implemented the second order total variation diminishing TVD-RK scheme [15].

3.3 Re-Sampling

As mentioned above, the interface has to be resampled during the evolution to maintain a well distributed set of sampling particles on the moving interface. The key step in this process is a least square approximation of the interface using polynomials at each particle in a local coordinate system. After the local reconstruction, we find the new foot-point associated to each active grid point and compute any needed geometric and Lagrangian information, such as normal, curvature, and also possibly an updated parametrization of the interface at this new foot-point. We will describe these processes in details in the following sections.

One subtle issue is how often one should resample the interface. As mentioned before, the criteria relies on how well the current particles resolve/sample the interface. This depends on the motion of the interface. For the simplest case, e.g., the interface is convected in a given external velocity field, the motion of each particle is decoupled from the others. In this scenario, we can simply follow each particle's Lagrangian trajectory (by solving an ODE) without resampling step in theory. However, in order to have a good sampling rate and quasi-uniform distribution of the particles one still needs to perform the resampling procedure once in a while unless the velocity field is uniform, such as a rigid motion. For more general motions, one may monitor the distribution of the particles and determine when resampling is needed. For simplicity, we resample after each time step in our current implementation. Together with the time constraint mentioned above, it is also guaranteed that the interface will not move out of the current computational tube after each time step.

3.3.1 A Local Reconstruction based on Least Square Fitting

To resample the interface, i.e., to find the new foot-point of each active grid point after motion of the interface, we reconstruct the interface locally near each active grid point. The reconstruction is based on least square fitting a polynomial in a local coordinated system. At each grid point, we first collect a set of neighboring particles on the interface using the available Eulerian reference. Then we select a fixed m particles and a local coordinate for the least square fitting. Here is the detailed procedure.

For each active grid $\mathbf{p}$ with the associated particle $\mathbf{y}$ (not necessary the closest point anymore after motion) on the interface, we first collect at least m active grid points (including $\mathbf{p}$), $\mathbf{p}_0, \mathbf{p}_1, \cdots$, in a small neighborhood of $\mathbf{p}$, whose associated particles on the interface are $\mathbf{y}_0, \mathbf{y}_1, \ldots$ (including $\mathbf{y}$). Carried with each of these particles is the old normal vector, denoted by $\mathbf{n}_0', \mathbf{n}_1', \cdots$, at the previous location of $\mathbf{y}_i, i = 0, 1, \ldots$ before the motion. We sort $\mathbf{y}_i$ in an ascending order according to its distance to $\mathbf{p}$. We pick the first m particles $\mathbf{y}_1, \cdots, \mathbf{y}_m$ according to some criterion described in the next section. Denote $\mathbf{y}_0$ to be the one that is closest to $\mathbf{p}$ among all the $\mathbf{y}_i$. Our least square fitting is constructed in the local coordinate system $\{(\mathbf{n}_0')^\perp, \mathbf{n}_0'\}$ with $\mathbf{y}_0$ as the origin. The key point is that in this local coordinate system the interface can be represented as a graph function locally if the underlying grid can resolve the geometry of the interface. The new coordinates, $\tilde{\mathbf{y}}_i$ for $i = 0, \cdots, m-1$, of

the collected m particles on the interface is given by

$$\tilde{\mathbf{y}}_i = \begin{pmatrix} \tilde{x}_i \\ \tilde{y}_i \end{pmatrix} = \begin{pmatrix} n_2 & -n_1 \\ n_1 & n_2 \end{pmatrix} (\mathbf{y}_i - \mathbf{y}_0) \,, \tag{1}$$

where $\mathbf{n}_0' = (n_1, n_2)$. Then we approximate the interface locally by a polynomial of degree $n(< m)$ using least square fitting. As an example, to fit a quadratic polynomial, i.e. $n = 2$, $f(\tilde{x}) = a_0 + a_1\tilde{x} + a_2\tilde{x}^2$, we minimize the L^2-difference between $f(\tilde{x}_i)$ and $\tilde{y}_i$. For instance, given m data points $(\tilde{x}_i, \tilde{y}_i)$, with $\tilde{x}_i \in [\tilde{x}_{\min}, \tilde{x}_{\max}]$, for $i = 1, \cdots, m$, we have the normal equation

$$\begin{pmatrix} m & \sum \tilde{x}_i & \sum \tilde{x}_i^2 \\ \sum \tilde{x}_i & \sum \tilde{x}_i^2 & \sum \tilde{x}_i^3 \\ \sum \tilde{x}_i^2 & \sum \tilde{x}_i^3 & \sum \tilde{x}_i^4 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \end{pmatrix} = \begin{pmatrix} \sum \tilde{y}_i \\ \sum \tilde{x}_i\tilde{y}_i \\ \sum \tilde{x}_i^2\tilde{y}_i \end{pmatrix} . \tag{2}$$

Solving this system of linear equations will give a local explicit representation of the interface. This strategy can be generalized to higher dimensions easily. For example, in three-dimension, we first construct the local coordinate system, $\{\mathbf{n}_1'^{\perp}, \mathbf{n}_2'^{\perp}, \mathbf{n}'\}$ based on the unique household reflector matrix that maps $\mathbf{n}'$ to $(0, 0, 1)$. Then using a polynomial, e.g., of the quadratic form

$$f(\tilde{x}, \tilde{y}) = a_{0,0} + a_{1,0}\tilde{x} + a_{0,1}\tilde{y} + a_{2,0}\tilde{x}^2 + a_{1,1}\tilde{x}\tilde{y} + a_{0,2}\tilde{y}^2 \,. \tag{3}$$

to approximate the local interface.

As mentioned before, the communication among particles along the interface during the evolution is embedded in the local reconstruction step, e.g., the least square fitting procedure in our algorithm. In our implementation we used the standard least square fitting procedure which works well in our extensive numerical tests. It seems that the least square fitting strikes a good balance between accuracy and stability from the over determined data. However, there are various potential modifications that can be incorporated to improve accuracy and/or stability. For examples, one way is to introduce appropriate weights for neighboring particles in the fitting energy according to the importance or uncertainty, e.g., based on the distance to the center point. Another possibility is to introduce some regularization term to penalize sharp features.

3.3.2 Important Issues for Choosing Particles for Local Reconstruction

The selection of m neighboring particles for local reconstruction needs some care. One condition is that they have to be distinct. In some extreme cases several foot-points may collapse to one location. This is possible when $\mathbf{p}$ is near where the interface is singular, such as corners of a square. If these m particles correspond to a single point, or are very close to each other compare to the underlying grid size, the local reconstruction of the interface will be inaccurate, unstable or even impossible. Another possible consequence for choosing very close particles for reconstruction is causing strict CFL condition for time step.

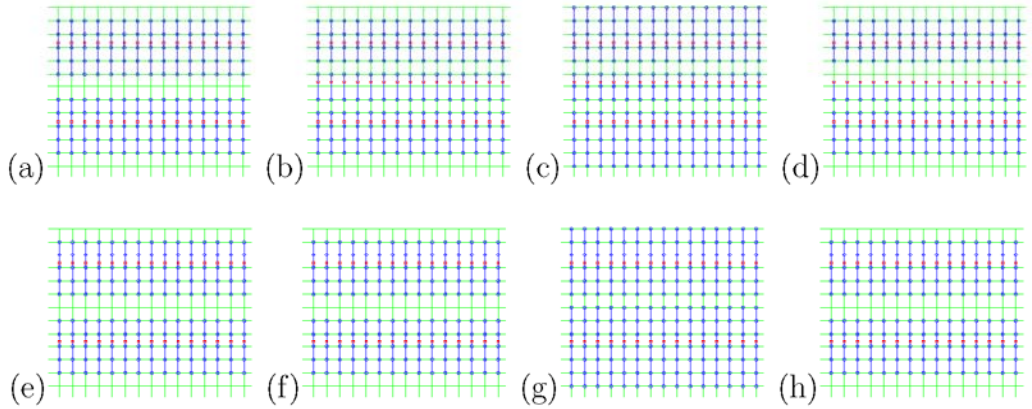

Figure 2: First row uses no restriction on picking points for local reconstruction. An artificial interface, the middle one, is created. Second row shows results with normal information incorporated. From left to right: after motion, after re-sampling, after activating new grid points with their foot-points and after inactivating grid points with their foot points. One obtains a similar result if a global parametrization is used instead.

As mentioned in the previous section, at a grid point $\mathbf{p}$ we first collect potential neighboring particles, which are associated to those active grid points in a neighborhood of $\mathbf{p}$, and form a list in ascending order according to its distance to $\mathbf{p}$. The base point is chosen to be the closest one. Once the base point is fixed we add particles one by one from the sorted list in the following way until we get m particles. When we try to add a particle, we compare that particle with all particles that we have accepted. If this new particle is too close to any points we collected, we will not pick this particle and continue down the list. In case that there are not enough particles in the initial collected set we can enlarge the neighborhood of $\mathbf{p}$ which will include more neighboring grid points and hence their associated particles on the interface. The enlargement of neighborhood is more likely to happen for grid points that (1) are near singularities such as corners, (2) are at the tube boundary, (3) have different segments of the interface in the neighborhood (see discussion below). This strategy makes sure we do have m distinct foot-points for local reconstruction, but the computational complexity is $O(m^2)$. A less expansive constraint is to relax the above the condition by requiring only the newly added point is not too close to the base point. The complexity of this strategy is only $O(m)$ but numerically we found that the error introduced will be larger in the resampling step. The closeness is defined on the order of local grid size, e.g. $0.25h$, where h is the local grid size.

Another important issue is that we want these m particles belong to the same segment of the interface. For example, when two segments of the interface are close to each other, e.g., when interface crossing or merging happens, special care has to be taken. In this case, these m neighboring particles, which are determined by neighborhood relation through the underlying reference grid, should be chosen consistently to avoid using conflict information

in local reconstruction. In Figure 2, we demonstrate the effect of mixing information from different segments. Figure 2(a) shows the initial configuration of moving two horizonal lines vertically upward. As before, blue circles are active grid points and red squares are their associated foot-points on the interface. If we arbitrarily collect particles associated to the grid points in a neighborhood of $\mathbf{p}$, we may mix particles from both segments which will cause trouble in the local reconstruction. For the example shown in Figure 2, the problem will happen for active grid points near the tube boundary $\partial\Gamma$. Using particles from **both** segments, one will reconstruct a completely wrong interface in the middle and generate wrong foot-points, as shown in Figure 2(b). This example implies that one has to incorporate extra information in this reconstruction step to distinguish particles from different segments.

The key observation for resolving this issue is that one should incorporate both Lagrangian information and Eulerian reference, which we have in our proposed method. For examples, the Lagrangian information can be the normal direction and/or some parametrization. If two different segments of the interface come close to each other, it means one can detect inconsistent Lagrangian information in particles that are close in physical space based on Eulerian reference. When this situation is detected, one can use consistency of Lagrangian information to classify neighboring particles into different segments accordingly. Depending on applications, we propose the following two strategies to decide whether those m particles collected above belong to the same segment. The first method uses the normal information. For instance, we classify these particles into groups so that each group has normal vectors approximately pointing in a similar direction before reconstruction. Of course, those normal-like vectors $\mathbf{n}'$ are not really normals at the current step in general. However, if the time step size is small enough, these vectors already provide enough information for us to tell if these particles belong to the same segment.

The second way is to classify those particles based on some global parametrization. For each particles on the interface, we assign a parameter. It is important to note that our parametrization is only for the purpose to provide neighborhood information among the particles in a parameter space, which can be used to distinguish different segments of the interface. No differentiation is taken with respect to the parametrization in our numerical algorithm. So we do not have to pose strict conditions on the parametrization such as close to the arclength. However, we will show a natural reparametrization procedure similar to the reconstruction step that can assign a new parametrization to the resampled foot points in the Section 3.3.3.

We now summarize these two approaches here.

1. **Using normal information.** In this case, we check if $\mathbf{n}'_0 \cdot \mathbf{n}'_j > \cos\theta_{\min}$ for all $j = 1, \cdots, m-1$, where $\mathbf{n}'_0$ is the normal at the base point $\mathbf{y}_0$. If there exists j such that this condition is not satisfied, we conclude that this set of particles consist of segments from different parts of the interface.

2. **Using a global parametrization.** A global parametrization, $g(\mathbf{y})$, is another strategy one might use to classify particles into various segments. If $|g(\mathbf{y}_0) - g(\mathbf{y}_j)| > \epsilon$

for some $j = 1, \cdots, m-1$, we conclude that these particles consist of segments from different parts of the interface.

If these particles have split into groups, we locally reconstruct several interfaces according to each group of these particles. Then for each of these reconstructed interfaces, we apply the method described below in the Section 3.3.3 to determine the closest point $\mathbf{y}$ from this active grid point $\mathbf{p}$. Then among all of these potential foot-points, we associate to the one which has the smallest distance to $\mathbf{p}$ as the new foot-point. This mechanism allows the foot-point for a particular grid point switches from one segment to another segment, which is necessary and important when different segments of the interface are getting close to each other. In this case the neighborhood region of a grid point may have to be enlarged to collect enough particles for the local reconstruction of multiple segments.

3.3.3 New Foot-Points and Theirs Associated Normal, Curvature and Global Parametrization

Now, to determine the new foot-point associated to $\mathbf{p}$ on this polynomial $\mathbf{f}(\tilde{\mathbf{x}})$, we minimize the function $d(\tilde{\mathbf{x}}; \mathbf{p})$ with respect to $\tilde{\mathbf{x}}$ defined by

$$d(\tilde{x}; \mathbf{p}) = \frac{1}{2}\|\mathbf{f}(\tilde{\mathbf{x}}) - \mathbf{p}\|^2. \tag{4}$$

Here we only consider in details two dimensional problems where the interface is a curve. Higher dimensional generalization is relatively straight-forward. The minimizer to (4) for a degree two polynomial $f(\tilde{x}) = a_0 + a_1\tilde{x} + a_2\tilde{x}^2$ can be found explicitly. In particular, we solve the following cubic equation for real root $\tilde{x}$

$$\tilde{x}^3 + \alpha_2\tilde{x}^2 + \alpha_1\tilde{x} + \alpha_0 = 0\,, \tag{5}$$

with

$$\alpha_2 = \frac{3a_1}{2a_2}\,,\ \alpha_1 = \frac{a_1^2 + 2a_0a_2 + 1}{2a_2^2}\ \text{ and }\ \alpha_0 = \frac{a_0a_1}{2a_2^2}\,. \tag{6}$$

If there is more than one real root to this cubic equation, we choose the one which gives the smallest function value of $d(\tilde{x}; \mathbf{p})$.

For higher order local reconstruction or higher dimensional problems, there is no explicit expression for the minimizer. In these cases, one has to implement an iterative solver to minimize the function (4). Since we have a pretty good initial guess from the previous foot-point, the iteration converges very fast.

There are situations where the minimizer $\tilde{x}^* = \text{argmin}\ d(\tilde{x}, \mathbf{p})$ lies outside the interval $[\tilde{x}_{\min}, \tilde{x}_{\max}]$. This implies that we are extrapolating the information. Numerically, this should be avoided due to stability and accuracy concern. If $\tilde{x}^* \notin [\tilde{x}_{\min}, \tilde{x}_{\max}]$, we simply inactivate the corresponding grid point in this re-sampling phase.

Other information on the interface can also be approximated from the reconstruction. For example, in the two dimensional case, the curvature on the new foot-point can be computed using

$$\kappa(\tilde{x}^*) = \left.\frac{f''(\tilde{x})}{\{1+[f'(\tilde{x})]^2\}^{3/2}}\right|_{\tilde{x}=\tilde{x}^*} = \frac{2a_2}{[1+(a_1+2a_2\tilde{x}^*)^2]^{3/2}}\,. \tag{7}$$

Higher dimensional problems can be easily done using a local reconstruction $f(s_1, s_2)$. We first compute the first and the second fundamental forms of this local reconstruction, and then we approximate both the mean curvature and the Gaussian curvature at the foot-point on the interface.

The normal vector at the foot-point can be approximated by the normal vector of the local reconstructed polynomial at the desired location, or be computed by the expression

$$\mathbf{n}(\tilde{x}^*) = \pm\frac{\mathbf{y}(\tilde{x}^*)-\mathbf{p}}{|\mathbf{y}(\tilde{x}^*)-\mathbf{p}|}\,, \tag{8}$$

where the $\pm$-sign is determined in such a way that the new normal vector is consistent to the one before the local reconstruction step. For example, one can use

$$\mathbf{n}(\tilde{x}^*) = \mathrm{sgn}[\tilde{\mathbf{n}}(\tilde{x}^*)\cdot\mathbf{n}']\tilde{\mathbf{n}}(\tilde{x}^*) \tag{9}$$

where $\mathbf{n}'$ is the normal at the base particle where we used to define local coordinate, and

$$\tilde{\mathbf{n}}(\tilde{x}^*) = \frac{\mathbf{y}(\tilde{x}^*)-\mathbf{p}}{|\mathbf{y}(\tilde{x}^*)-\mathbf{p}|}\,. \tag{10}$$

Note that the sign can distinguish grid points on each side of the interface and the change of the sign indicates that a grid point is crossing the interface after the motion. In the case where there are multiple segments of the interface in the neighborhood, there could be multiple foot points on different segments. We choose the closest one and its normal is determined in consistency to that particular segment.

In addition to those geometric quantities, such as the normal direction, curvature, etc, which can be easily computed after local interface reconstruction, one can also track/monitor other Lagrangian quantity, such as some material distribution or parametrization, on the moving interface. In the simplest case, the Lagrangian quantity just follows the particle trajectory. If this is the case, one can simply track the Lagrangian quantity by least square fitting and interpolation during the resampling step. For example, suppose that $g(\mathbf{y})$ is the Lagrangian quantity defined on the interface. After one time step of evolution of the interface, during the resampling step, we locally approximate $g(\mathbf{y})$ by least square fitting $g(\mathbf{y}_i)$, where $\mathbf{y}_i$ are the neighboring particles, in the local coordinate system as what we did for the local reconstruction of the interface. Then we interpolate the value of g at the newly sampled foot-point. In general the Lagrangian quantity may also evolve on the interface according to certain laws, e.g., governed by some PDE. We will discuss how to solve PDE(s) on moving interface using grid based particle method in a latter work.

In some of our examples, we use global parametrization to distinguish different segments of an interface, e.g., for detection of collision, consistent local reconstruction of multiple segments, and control of topological changes. Suppose that we have an initial parametrization, we need to maintain it during the evolution. One way is to treat it as a Lagrangian quantity and to follow the particle trajectory by least square fitting and interpolation during the resampling step as described above. However, parametrization is not unique and we only need a crude one since we do not need an arclength parametrization or differentiation with respect to the parameters, we use the following simple averaging strategy for the newly sampled foot-points.

Denote $g(\mathbf{y})$ to be a parametrization of the interface. If a global parametrization $g(\mathbf{y})$ is needed for the new foot-point at $\mathbf{y}$, we simply perform a weighted averaging of $g(\mathbf{y}_i)$. Because we require only *a* global parametrization, the choice of this averaging can be non-unique. For example, in two dimension, we parameterize the curve using a global parametrization $\theta \in [-\pi, \pi)$. This essentially means that we look for a transformation which distributes all foot-points on a unit circle. We can use the following weighted averaging

$$g(\mathbf{y}(\tilde{x}^*)) = \mathrm{Arg}(c + s\sqrt{-1})\,, \tag{11}$$

where

$$c = \frac{\sum_i w_i \cos\gamma_i}{\sum_i w_i} \;\text{ and }\; s = \frac{\sum_i w_i \sin\gamma_i}{\sum_i w_i}\,, \tag{12}$$

with the weights given by

$$w_i = \exp\left(-\frac{\tilde{x}_i^2}{\max(\tilde{x}_{\min}^2, \tilde{x}_{\max}^2)}\right)\,. \tag{13}$$

Since the parametrization is only used for the purpose of distinguish particles that are far away along the interface in our method, we need to avoid the following cases from happening: particles that are far apart along the interface have close parametrization and vice versa. So we use simple reparametrization techniques to maintain the parametrization of the quasi-uniformly sampled foot-points evenly distributed in the parameter space. In 2D we sort θ_i for all i on the interface and also relabel them such that $\theta_{i+1} - \theta_i = 2\pi/N = \Delta\theta$ where N is the total number of foot-points. Hence, neighboring foot-points (sampled according to an underlying grid) on the interface are separated on the order of $O(\Delta\theta)$ in the parameter space.

The situation in higher dimensions is a little different since this re-parametrization technique cannot be generalized in a straight-forward manner. First of all, unlike the simple unit circle as in the previous case, there is no unique representation of all points on a sphere. More importantly, it is not clear how one can *evenly distribute* all points on a sphere. In fact, a nice global parametrization of the sphere itself is still an open problem named the Thomson problem. The original problem is to determine a stable configuration of classical electrons

on the surface of a sphere where these electrons are repelling each other by electrostatic repulsion. Mathematically, one determines a configuration $(\mathbf{r}_1, \cdots, \mathbf{r}_N)$ which minimizes

$$\sum_i \sum_{j \neq i} \frac{1}{|\mathbf{r}_i - \mathbf{r}_j|} \tag{14}$$

with constrains $|\mathbf{r}_i| = 1$.

Suppose we use the polar coordinate on a sphere $(\theta(\mathbf{y}), \phi(\mathbf{y}))$ for the global parametrization. We use the same averaging technique as in 2D for θ and ϕ independently at a newly sampled foot-point. For re-parametrization, we try to minimize the (14) using the method of steepest descent with the initial configuration given by the averaging procedure. In practice we do not need to find the exact minimizer and also since each particle does not move too much after each time step, for every foot-point, $\mathbf{y}_i$, with a global parametrization given by $(\theta(\mathbf{y}_i), \phi(\mathbf{y}_i))$, we minimize the energy of $\mathbf{r}_i$ involving only neighboring particles's $\mathbf{r}_j$ by projecting these points on the sphere to the tangent plane at $\mathbf{r}_i$ and perform several steps of relaxation following the steepest descent. This provides a good enough global parametrization for our purpose to distinguish points from different segments of the interface.

Last but not least, if the signed distance function is necessary, we use

$$\text{sgn}[\mathbf{n}(\tilde{x}^*) \cdot \tilde{\mathbf{n}}(\tilde{x}^*)]|\mathbf{y}(\tilde{x}^*) - \mathbf{p}|\,. \tag{15}$$

3.4 Updating the Computational Tube

To ensure that the computational tube, Γ, is following the moving interface with a fixed width, the computational tube of active grid points has to be updated by adding and/or deleting active grids together with their associated foot-points. This process consists of an activation phase and an deactivation phase. After resampling procedure is done for grid points in current tube, we first activate those inactive grid points $\mathbf{x}$ which has at least one current active neighboring grid point whose distance is smaller than the tube width. Next, we de-activate any active grid points whose distance to the interface is more than the tube width, γ.

To activate new grid points and to find their corresponding foot-point on the interface and maybe other information at these foot-points, we follow the same local reconstruction procedure described in Section 3.3. The only difference is that the newly activated grid point does not contribute an associated particle to the set of neighboring particles used for the local reconstruction of the interface.

After each activation step, we simply check the distance to the interface of each active grid point. If the distance is larger than γ, we deactivate the corresponding grid point, meaning that we remove $\mathbf{p}$ from Γ and also ignore its corresponding foot-point $\mathbf{y}$.

There are other situations where we deactivate certain grid points. For example, if the local geometry of the interface can not be resolved by the finest resolution of the underlying

grid, e.g., curvature κ is too large. This procedure can also be regarded as some kind of numerical regularization or filtering for our method. Of course, to represent the interface more accurately and efficiently, one needs to have a local refinement strategy for the underlying grid and hence the sampling density can adjust to the local geometry of the interface. This will be explained in the next section.

In general the tube width γ depends on local mesh size, e.g., a constant times local mesh size. So the computation cost is proportional to the total number of particles on the interface or the number of active grid points in the computational tube, which is proportional to the tube width, the size and dimension of the interface, and is inverse proportional to the grid resolution.

3.5 Adaptivity

To resolve both the dynamics and the geometry of a complicated moving interface efficiently adaptivity is very important. For example, high sampling rate should be used when there is fast dynamical and/or fine geometrical feature. In the grid based particle method, the sampling rate is determined by the underlying mesh, i.e., the density of the sampling particles on the interface is inverse proportional to the local mesh size. So it is natural to control the adaptivity of sampling particles through local mesh refinement. For simplicity, we describe some details of adaptive procedure based on local mesh refinement for Cartesian grid in the paper. Standard mesh refinement can also be used on triangular mesh.

An important advantage for our grid based particle method is that the underlying grid is used to provide (1) unconnected sampling particles with an Eulerian reference, (2) neighborhood information for those unconnected particles on the interface. No PDE is solved on the underlying grid. This allows us to use simple diadic subdivision of grid cells based on quad-tree (2D) or oct-tree (3D) data structure for the local mesh refinement. An appropriate monitor function and refinement criterion is used to indicate when and where refinement is needed. In our implementations we use (principal) curvature as the monitor function. We check that at each active grid point $\mathbf{p}$ with foot-point $\mathbf{y}$ if the following condition is satisfied,

$$|\kappa| < \frac{c}{h_{i,j}}, \tag{16}$$

where κ is the principal curvature with largest magnitude at $\mathbf{y}$, $h_{i,j}$ is the local mesh size at $\mathbf{p}$, and c is some thresholding constant one can choose. Numerically, c means the least ratio between the radius of the osculating circle and the local mesh size. If this condition is violated, we either deactivate the grid point $\mathbf{p}$ and remove its foot-point $\mathbf{y}$ if no local mesh refinement is going to be carried out, i.e., we remove fine features that can not be resolved by the underlying mesh, or we perform the following local mesh refinement until the condition (16) is satisfied. In our implementation, we use a staggered grid which means each grid point is the center of a cell. Once the condition (16) is violated at the foot-point of an active grid point $\mathbf{p}$ after resampling, we simply subdivide the cell centered at $\mathbf{p}$ and

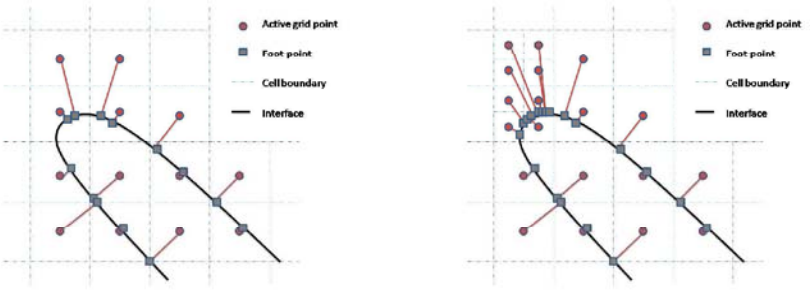

Figure 3: When high curvature region is detected, we will refine the corresponding cell and will assign new associating foot-points.

include the center of each sub-cell into the list of active grid points, i.e. the computational tube Γ. The foot-points of these new grid points are computed just like adding new grid point when updating computational tube described in Section 3.3 and 3.4. For example in 2D, we first divide the original cell into 4 sub-cells, with each centered at $(x_{i\pm 1/4}, y_{j\pm 1/4})$. Next we activate these four new grid points and also de-activate the original cell center. We demonstrate this procedure in figure 3. If this cell needs to be further refined, we then activate all sub-cells centered at $(x_{i\pm 1/4\pm 1/8}, y_{j\pm 1/4\pm 1/8})$ and remove any grid points from the previous level of refinement. This implies that the total number of grid points originated from the cell centered at $\mathbf{p}$ is at most $4^{(L-1)}$ where L is the level of local refinement with $L = 1$ corresponding to the grid in the coarsest level. Of course, this is not the only way how one might use for local refinement. For example, in the previous quad-tree structure approach, one may keep all grid points from the previous level of refinement. This implies that in two dimension there are totally $(4^L - 1)/3$ grid points stemming from a cell in the coarsest level.

As the interface evolve, certain part may become smoother. Therefore, we do not need a very refined mesh to represent that part of the interface. In this case, we will lower the level of the adaptivity. In the current implementation, we use a similar criterion and procedure as the refinement to locally coarsen the underlying grid when this oversampling is detected. The curvature used in the criterion is the average of curvature at those foot-points associated to the centers of the subcells in the coarse cell.

With local grid refinement, the computational tube width and the neighborhood of a grid point are all defined in terms of local grid size. This general local mesh refinement strategy can also be applied in other situations, for example when two interfaces are getting close as discussed in the next Section.

3.6 Control of topology

One of the most challenging task for moving interface problem is how to control and handle topological changes according to the physics. When two pieces of interface get close, two scenarios can happen:

I. The two pieces interact with each other. They collide and merge, e.g., two bubbles merging, one bubble splitting into two bubbles.

II. The two pieces cross each other, e.g., two wave fronts passing each other.

We explain how we can easily handle both these situations using grid based particle method. The first key step is the detection of two pieces of interface getting close, which is equivalent to detection of inconsistent Lagrangian information in particles that are close in the physical space based on the Eulerian reference, as described in Section 3.3.2. The Lagrangian information could be normal direction or some parametrization. If an active grid point's associated particle contains Lagrangian information inconsistent with at least one of those particles that are associated to those active grid points in a neighborhood of the grid point, it indicates two pieces of interface are getting close. Once this is detected we can apply the following procedures to handle either one of the above two scenarios. For case I, we just deactivate the grid point, whose neighborhood contains inconsistent Lagrangian information, and its associated particle. It is easy to add or delete particles on the interface since the interface is represented by meshless particles, which does not require complicated reconnection or reparametrization procedure. If the computational tube radius is γ, the radius of the neighborhood should be at least 2γ to take into account the tube width. For case II, no special operation needs to be done since the resampling procedure (see Section 3.3.2) can (1) distinguish and reconstruct different pieces consistently, (2) switch an grid point to the closest piece automatically. As a result two interfaces can cross each other freely. These two cases are demonstrated in examples from Section 4.4.

Remark 1: Local grid refinement described in Section 3.5 can be implemented when two interfaces are getting close. It can help to resolve both interfaces and their interactions better.

Remark 2: We can deal with two interfaces crossing each other since the interface is sampled by closed-by grid points on both side which essentially means that our representation of the interface has some redundancy. This redundancy allows us to reconstruct two crossing interface separately and switch the association of a grid point from one interface to the other easily. However, if there are too many pieces of an interface collapsing to a point, e.g., a wave front collapses to a focal point, then one has to introduce another dimension to parametrize all these different pieces. This is why we turn a geometric optics problem in the physical space into a phase space formulation in Section 4.6. As the consequence, an interface with codimension two in the phase space has to be used. The example also shows that the grid based particle method can be easily extended to manifolds with codimension higher than one.

4 Examples

In this section we apply the grid-based particle method to various numerical examples. We will first study some convergence properties of our method in Section 4.1. Next we will show examples in two dimensions in details from Section 4.2 to Section 4.5. Then in Section 4.7, we will generalize the algorithm to deal with three dimensional motions where the interface is a closed surface.

Unless otherwise specified, we will be using the tube with radius $\gamma = 1.1h$, where h is the local grid size. We use quadratic polynomials for the least square fitting in local reconstruction, which uses 6 particles in two dimensions and 10 particles three dimensions. The time integration is done using the TVD-RK2 scheme with time step equals $0.75h_{\min}$, where $h_{\min}$ is the smallest grid size of the underlying mesh. In most examples below, we do not have any global parametrization of the interface and we mainly look only at the normals to check if there is any conflict in the Lagrangian information.

4.1 Convergence Analysis

One of the most important step in the algorithm is the local reconstruction, which determines the approximation error of the interface during the resampling process. The least square fitting algorithm, which only uses neighborhood information among particles, gives us a quite flexible way of balancing accuracy and robustness. However, since there is no pre-determined structure on the sampling particles, it makes rigorous analysis of the approximation error a challenging task. Here we use numerical tests to demonstrate the accuracy of the reconstruction step.

For each activated grid point $\mathbf{p}$ near a circle of radius $r = 0.25$ centered at $\mathbf{x}_c$, we first assign its associated foot-point the closest location on the interface given by

$$\mathbf{y} = \mathbf{x}_c + r\frac{\mathbf{p} - \mathbf{x}_c}{|\mathbf{p} - \mathbf{x}_c|}. \tag{17}$$

Then, without any motion, we apply our re-sampling algorithm described in Section 3.3. In most situations, the local reconstruction step using the least square approximation will perturb the locations of these foot-points. In our test we will show how the approximation error converges to zero when the underlying grid is refined uniformly, since the resolution of sampling particles on the interface is proportional to the resolution of the underlying grid in the grid based particle method.

We do not have a theoretical estimates on the order of convergence, but numerical experiments show that the error we made in the local reconstruction converges to zero like $O(\Delta x^4)$. Figure 4 (a) shows (log-plot) the convergence as we refine the underlying grid. For each fixed Δx, we define the L^2 error made at all foot-points by

$$E_{\Delta x} = \left[\int_{-\pi}^{\pi} \left(|\mathbf{x}_c - \mathbf{y}_{\Delta x}| - r\right)^2 d\theta\right]^{1/2}. \tag{18}$$

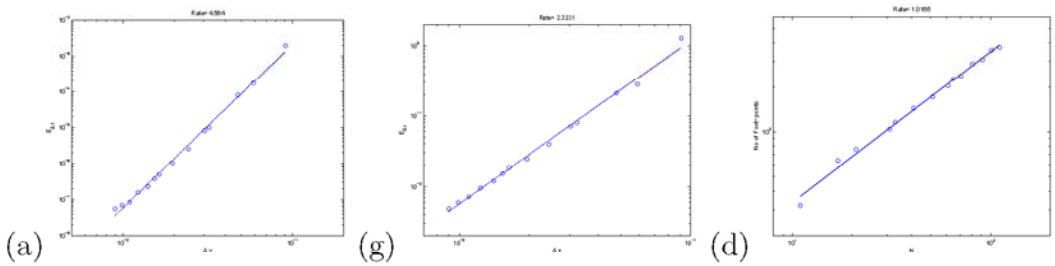

Figure 4: Convergence analysis with only local reconstruction for (a) locations of the foot-points, (b) their corresponding curvature, and (c) the total number of active grid points.

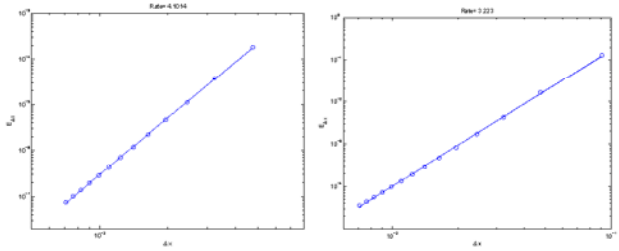

Figure 5: Convergence analysis for the (left) reconstruction and (right) constant motion of a sphere.

Numerically to approximate this integral, we first compute both the error made for each foot-point $\mathbf{y}_{\Delta x}$ and also the angle θ made between the x-axis and the vector $(\mathbf{x}_c - \mathbf{y}_{\Delta x})$. Then we sum all these errors with the weight according to the distribution of these θ's.

The solid line shows log-plot of the least square fitting of the error in the form

$$E_{\Delta x} = c_1(\Delta x)^{c_2}\,, \tag{19}$$

whose slope gives an approximation to c_2 (4.564). We have repeated a similar test example in figure 5 for a sphere in three dimensions. The corresponding rate of convergence is also approximately four (4.1014).

Another important geometric quantity which is very often used in moving interface problems is the curvature. In figure 4 (b), we repeat the above procedure but also compute the local curvature from the local reconstruction. Since determining the curvature requires differentiating the local reconstruction twice, we found that the rate is dropped by approximately two, to 2.3231.

Figure 4 (c) shows the total number of the number of foot-points on the circle vs. the number, N, of underlying grid point in each direction. It shows that the number of sampling particles grows approximately linearly (1.0166) with the number of grid in each dimension.

The above test only shows the approximation error in the reconstruction step. Next we

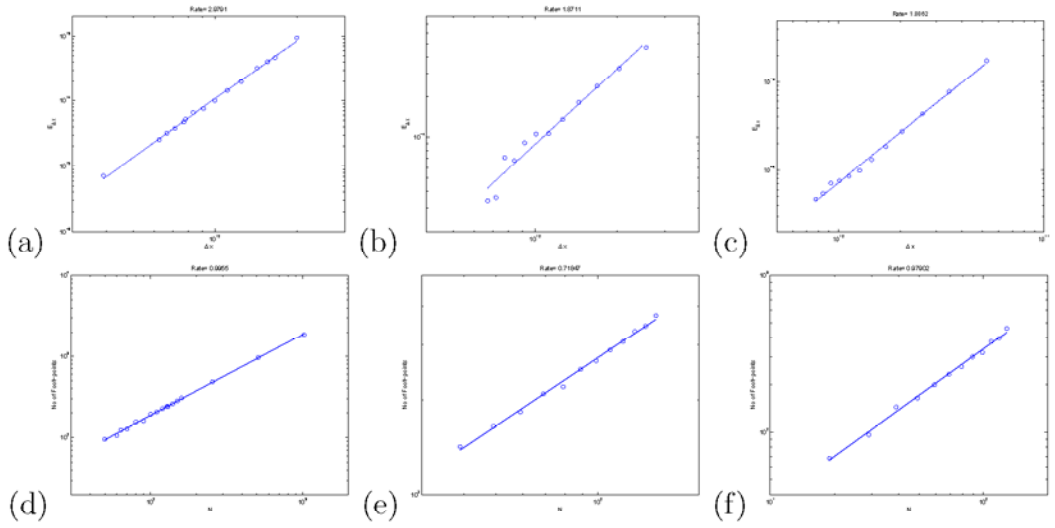

Figure 6: (First row) Convergence analysis and (second row) change in the number of foot-points for the (a,d) constant motion, (b,e) rotation and (c,f) the motion by mean curvature of a circle.

show convergence with time evolution by testing the motion of a circle in a constant velocity field, e.g., a simple rigid body rotation, and the motion by mean curvature.

In figure 6 (a), we compute the error we made in moving a circle of radius 0.2 initially centered at (0.25,0.25) under the velocity $\mathbf{u} = (1, 1)$ for $t = 0.5$. Figure 6 (d) shows the corresponding change in the number of foot-points in the simulation. For this simple linear convection problem, we use $\Delta t = 0.95\Delta x$. Since the velocity is constant, RK2 is exact in time discretization. Hence the error at a fixed time is the accumulation of approximation error in each resampling step, which should behave like $O(\Delta x^4 \cdot \Delta x^{-1}) = O(\Delta x^3)$ which matches with the numerical results (2.9791). The corresponding order for three dimensions is (3.223).

For the rigid body motion, we consider the following rigid body motion

$$\begin{aligned} u &= 1 - 2y \\ v &= 2x - 1\,. \end{aligned} \tag{20}$$

The initial circle is centered at $(0.5, 0.75)$ with radius 0.25. We rotate the circle for one revolution in time $t = \pi$. Figure 6 (b) shows the error in the final location of the interface. The corresponding rate is of $O(\Delta x^2)$ (1.8711), which is due to using RK2 in time discretization. We also plot the change in the total number of foot-points in figure 6 (e).

In figure 6 (c) and (f), we show the error we made in computing the mean curvature motion of a circle with initial radius $r = 0.4$ and also the corresponding variation of the number of foot-points. We solve the interface location at a fixed time $t = 0.05$. Since the

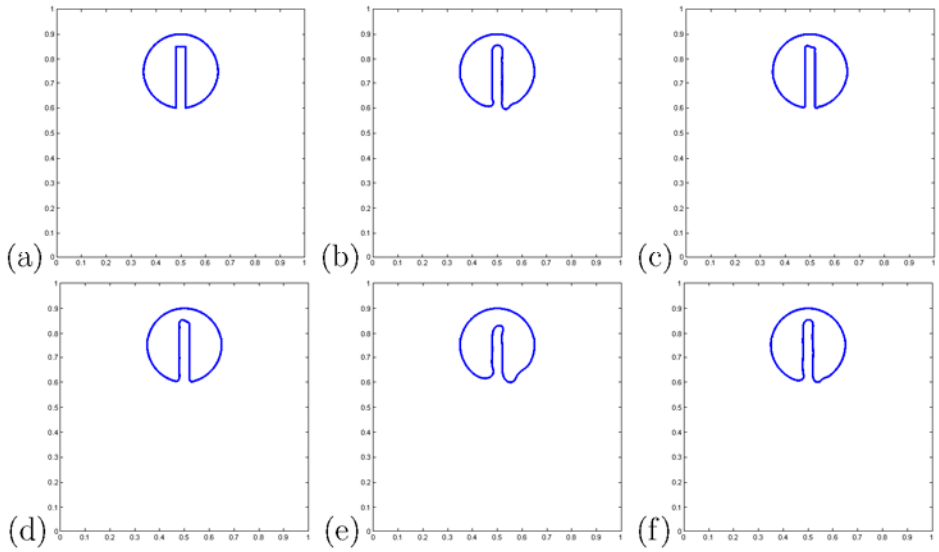

Figure 7: Rigid body rotation of (a) the Zalesak's disk after one revolution with the uniform resolution of (b) 257^2 and (c) 1025^2 and (d) with the coarsest resolution of 257^2 and the finest resolution of 1025^2. Solution after 10 revolutions using (e) a uniform resolution of 257^2 and (f) the same underlying adaptive mesh as in (d).

underlying PDE is nonlinear parabolic, which imposes a more restrictive CFL condition of $\Delta t \sim O(\Delta x^2)$. Numerically we pick $\Delta t = 0.5\Delta x^2$. Since the approximation of curvature is second order accurate, the rate of convergence rate is $O(\Delta x^2)$ (1.8862).

4.2 Rigid Body Rotation of a Zalesak Disk

In this example, we rotate a Zalesak disk, as shown in figure 7 (a), by the same rigid body flow introduced in (20). The disk will rotate around the point (0.5,0.5) for one revolution in time $t = \pi$. The main challenge for this example is the singularities on the interface, i.e. those four sharp corners. For this simple rigid motion, we know exactly where the singularities are during the evolution. So particle method is usually better since each particle on the interface just follows a simple ODE without coupling with other particles. Singularities do not harm anything. However, we do not know when and where singularities may develop in general. Moreover, we may have to approximate/reconstruct the interface near singularities to compute geometric quantities in many applications. In our test we do not explicitly track the location of singularities. Least square fitting is used for local reconstruction during the resampling step at all places. We show that the adaptive algorithm described in Section 3.5 will make automatic local refinement near those corners. Figure 7 (b-c) show the solution of one revolution using a relative coarse grid of resolution of 257^2 and a very fine grid of resolution of 1025^2, Figure 7 (d) shows the result of using an adaptive grid with the coarsest

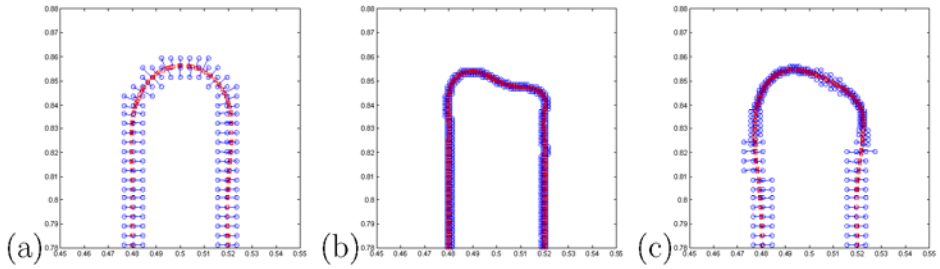

Figure 8: A zoom in of the active grid points with their associated foot-points of the Zalesak's disk after one revolution with the uniform resolution of (a) 257^2 and (b) 1025^2 and (c) with the coarsest resolution of 257^2 and the finest resolution of 1025^2.

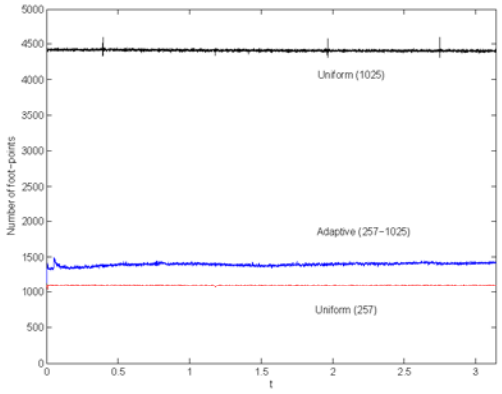

Figure 9: Changes in the total number of foot-points for the Zalesak's disk rotation for one revolution.

resolution of 257^2 and the finest resolution of 1025^2. A zoom-in of these figures are shown in figure 8. We see how the local grid and the correspond foot-points are refined near the corners. In figure 9, on the other hand, we plot the change in the total number of foot-points in time on different grid. It shows that the total number of foot-points for the case using adaptive grid is much lower than that using a uniform fine grid. Since the computational cost is proportional to the total number of foot-points the adaptivity we introduced improve the efficiency dramatically. We also show the solution after 10 revolutions ($t = 10\pi$) using this adaptive underlying mesh in figure 7 (f). It is interesting to note that although the corners are smeared out to some extent due to least square fitting in the local reconstruction, which is done at every time step, the numerical diffusion is effectively controlled by the grid resolution and does not deteriorate much in time (even for the uniform coarse grid).

4.3 Motions under Single Vortex Flow

The following example on vortex flow has been widely used [6, 8, 11] as an important test case for various numerical methods. It was originally proposed by Bell et al. [2] to test if a numerical method is able to resolve very thin filaments. The initial profile of the interface is a circle centered at (0.5,0.75) with radius 0.15. The velocity field is defined by the following stream function

$$\Psi = \frac{1}{\pi} \sin^2(\pi x) \sin^2(\pi y) \,. \tag{21}$$

In this example, we use a global parametrization to distinguish different segments on the interface. As the interface rolls up, different segments of the interface will get closer and closer to each other. The most important issue here is to use consistent information i.e., particles from the same segment, for local reconstruction of the interface during the resampling step. Using normals may become too sensitive when fine structure develops for this example. The initial parametrization for the circle is updated during the resampling step using the simple averaging scheme described in Section 3.3.3.

On the first row of figure 10, we plot our solutions at different time $t = 0.6, 1.2, \cdots, 3.0$ using a uniform underlying grid with the resolution of 1025^2, i.e. $h = 1/1024$. On the bottom two rows, we show the corresponding results using an adaptive grid with the coarsest resolution of 129^2 and the finest resolution of 1025^2. The mesh is locally refined if (1) large curvature is detected, i.e., criterion (16) is violated, (2) when two segments are getting close, i.e., once different segments are detected in local reconstruction on current mesh the mesh is locally refined once. The second row uses a more restrictive criterion, i.e., a smaller c in equation (16), which is enough to capture as detailed features as using a uniform fine grid. While the third row uses a larger c and hence it looses some part of the long thin tail.

Figure 11 shows the change in the total number of particles as a function of time. The left one corresponds to a uniform fine grid, the right twos correspond to an adaptive grid with more restrictive criterion and less restrictive criterion respectively. As we can see, when finer and finer structure develops more and more particles are used to sample the interface in

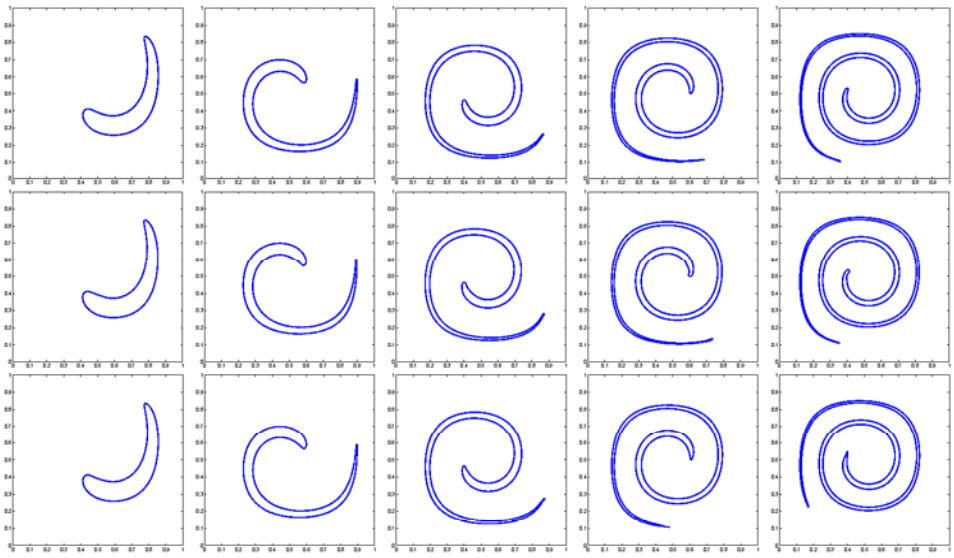

Figure 10: Motion under a single vortex flow of a circle at $t = 0.6, 1.2, \cdots, 3.0$ with the resolution of 1025^2 (upper row) and with the coarsest resolution of 129^2 and the finest resolution of 1025^2 using a smaller $c = 0.01$ (middle row) and a larger $c = 0.02$ (bottom row) in the equation (16).

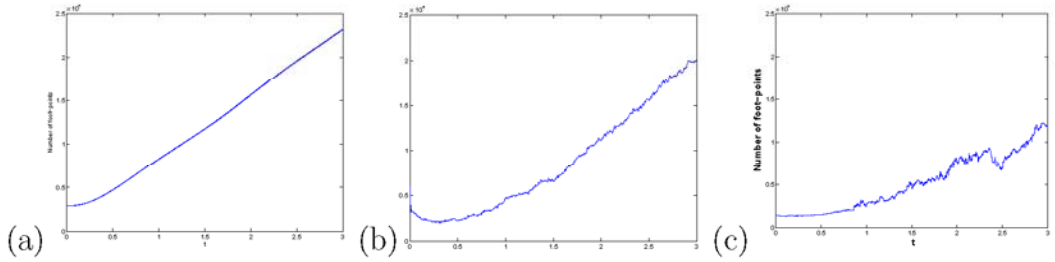

Figure 11: Motion under a single vortex flow of a circle. Total number of foot-points, or the number of active grid points, with (a) a uniform resolution of 1025^2 and the coarsest resolution of 129^2 and the finest resolution of 1025^2 using (b) $c = 0.01$ and (c) $c = 0.02$ in the equation (16).

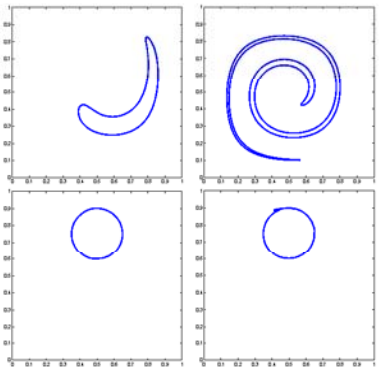

Figure 12: Motion under a single vortex flow with rewind motion of a circle at (top row) $t = T/2$ and (bottom row) $t = T$ using the coarse resolution of 129^2 and the finest resolution of 1025^2 with (left column) $T = 2$ and (right column) $T = 8$.

all cases. However, the adaptive grid uses fewer particles. For this particular example, there are two interesting phenomena. First, on the adaptive grid the total number of particles decrease initially (in particular the middle one) because the initial small circle stretches out smoothly at the beginning before sharp tip and long thin tail develops. Thus the mesh may be coarsened at the beginning. Second, once sharp tip and long thin tail develops the grid is refined almost everywhere along the interface. Hence the total number of particles on both grids are getting close. We are using approximately 2.4×10^4 Lagrangian particles to sample the interface at $t = 3$ using a uniform underlying grid, while approximately 2×10^4 particles to sample the interface at the same value of t using an adaptive grid with the coarsest resolution of 129^2 and the finest resolution of 1025^2. These numbers are much less than the particle level set method [6, 8]. The method in [6] requires approximately 5.9×10^4 particles at $t = 0$ to sample the initial circle with grid resolution of only 256^2 in the particle level set method [6], and at least 5.5×10^4 particles at $t = 0$ with grid resolution of 1000^2 using the lagrangian particle level set method [8]. [6] did not report any numbers on the Lagrangian particles used, but we expect more and more particles were required to fix the interface as it rolls-up.

To test mass conservation we follow [10] and 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 12 shows the results for different T. The mean distances from the point $(0.5, 0.75)$ for the cases $T = 2$ and 8 are 0.1501 and 0.1512, respectively, with the corresponding standard deviations 7.0549×10^{-4} and 2.9991×10^{-3}.

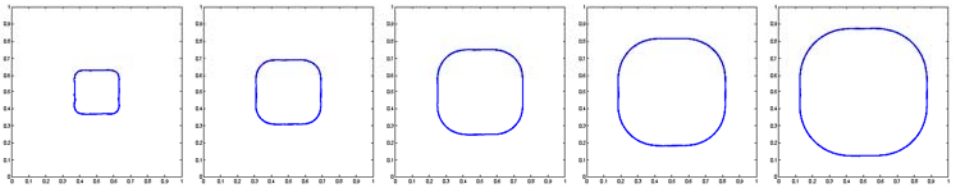

Figure 13: Motion in the outward normal direction of a square with the coarsest resolution 129^2 and the finest resolution 1025^2.

4.4 Motions in the Normal Direction

We now consider examples of interface motion with given normal velocity. Since normal direction is involved in the motion law the problem becomes non-linear.

The first example is motion of a square centered at $(0.5, 0.5)$ with a length of 0.2 for each side with constant outward normal velocity equal to one, which means that interface will grow outward in time at a uniform speed. We also test on the motion with a constant speed one in the inward normal direction. Initial interface is a square with sides of length 0.8. In our test we solve each of these motions with different initial interfaces up to $t = 0.3$.

Since the problem is nonlinear, we are interested in finding the right weak solution, the viscosity solution. For the motion in the outward normal direction, although the interface has singularities initially, i.e., at the four corners the normal direction is discontinuous, the corners will become rounded and the normal direction becomes continuous immediately. This situation corresponds to a rarefaction wave phenomenon in hyperbolic conservation law. The numerical results are shown in figure 13. For the inward normal motion, the viscosity solution will shrink while maintaining a perfect square, i.e., the corners correspond to shocks in hyperbolic conservation law. Figure 14 shows that our algorithm can compute the right viscosity and handle the sharp corner nicely. Unlike usual Lagrangian particle methods where adjacent sides will cross each other at the corners, we enforce the deactivation of active grid points and removal of their associated points when Lagrangian information contradicts in neighboring particles as described in Section 3.6. Here we use the normal direction as the Lagrangian information and we do not need any global parametrization in this example. To clearly see how those foot-points, grid points and their corresponding normal vectors interact, we also zoom in at the top right corner of the solution at $t = 0.3$, figure 15. On figure 15(a), we plot the foot-points in squares and their corresponding normal in vectors. On (b), we have each active grid point in circle and we also link to their corresponding foot-point. Since we use adaptive grid the grid points and the associated particles have finer resolution near the corners.

In the next example, we show that the our method can control the topology when two interfaces get close, i.e., one can choose either the viscosity solution or the multi-valued solution. In figure 16, we use two separated circles as the initial condition. These two circles both have radius 0.1 centered at (0.4,0.4) and (0.6,0.6), respectively. Both of these circles

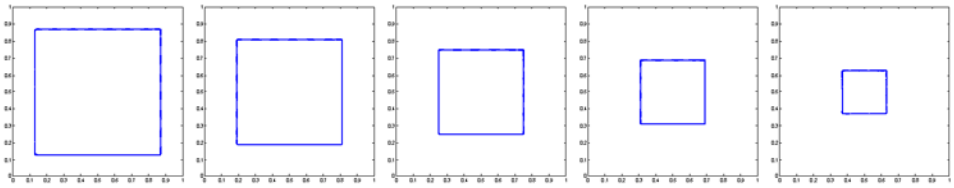

Figure 14: Motion in the inward normal direction of a square with the coarsest resolution 129^2 and the finest resolution 1025^2.

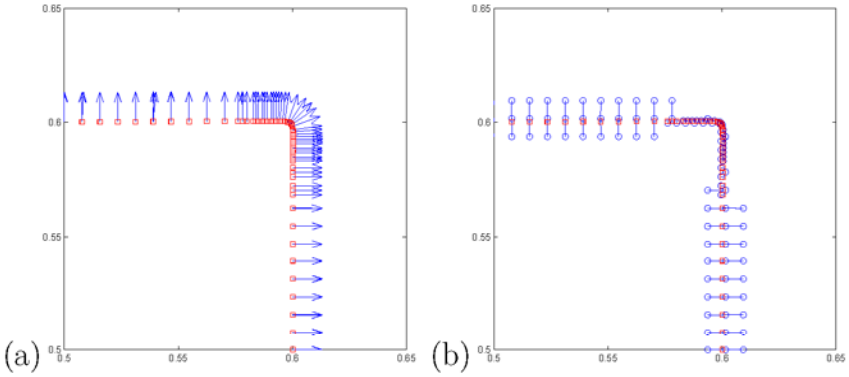

Figure 15: Motion in the inward normal direction of a circle with the coarsest resolution 129^2 and the finest resolution 1025^2. (a) Normal vectors and (b) their corresponding active grid-point locations at the final time step.

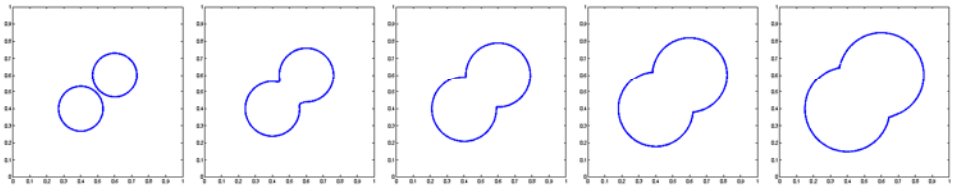

Figure 16: Merging of two circles with motion in the outward normal direction with the coarsest resolution 129^2 and the finest resolution 1025^2 at $t = 0.04, 0.08, \cdots, 0.4$.

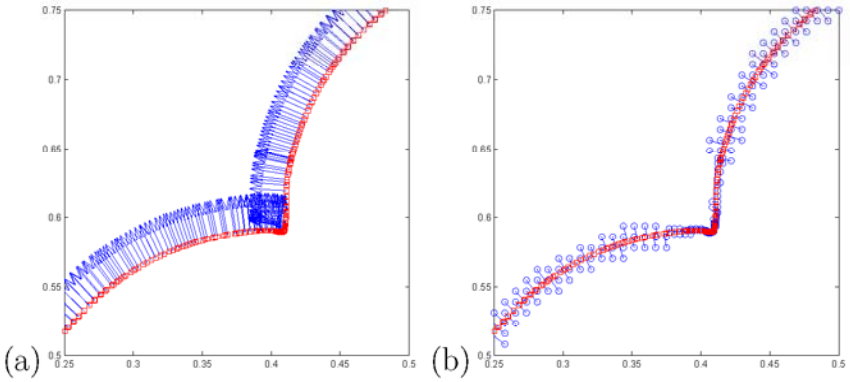

Figure 17: Merging of two circles with motion in the outward normal direction with the coarsest resolution 129^2 and the finest resolution 1025^2. (Left) Normals and (right) their corresponding active grid-point locations at the final time step.

will grow at a uniform speed in the outward normal direction. Since the radius of the circle will grow linear in time, they will touch each other at $t = 0.1(\sqrt{2} - 1)$. In figure 16 we show the evolution of two circles in the viscosity sense, i.e., they merge when they collide, at $t = 0.04, 0.08, \cdots, 0.4$. Numerically, we just have to deactivate any grid point and removal of its associated particle when collision is detected, e.g., if their normal vectors conflict with each other in a small neighborhood centered at the active grid (see Section 3.6). We also show the active grid points, foot-points and their corresponding normal vectors in figure 17. Local grid refinement is automatically employed when two interfaces are getting close and when singularity develops. One can clearly see that the interface is nicely sampled at the singularities using local grid refinement.

On the other hand, we can also compute the multi-valued solution, namely allow two interfaces crossing each other as long as we construct multiple pieces consistently and allow a grid point to switch its closest point in the resampling step as explained in Section 3.3.2 and Section 3.6. Since normal direction alone is not good enough to distinguish two different interfaces when they overlap substantially (their normals near the crossing point become similar), we use a global parametrization of the interface as described in Section 3.3. For

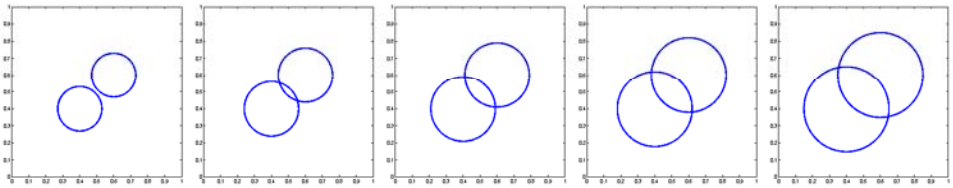

Figure 18: Crossing of two circles with motion in the outward normal direction with the coarsest resolution 129^2 and the finest resolution 1025^2 at $t = 0.04, 0.08, \cdots, 0.20$.

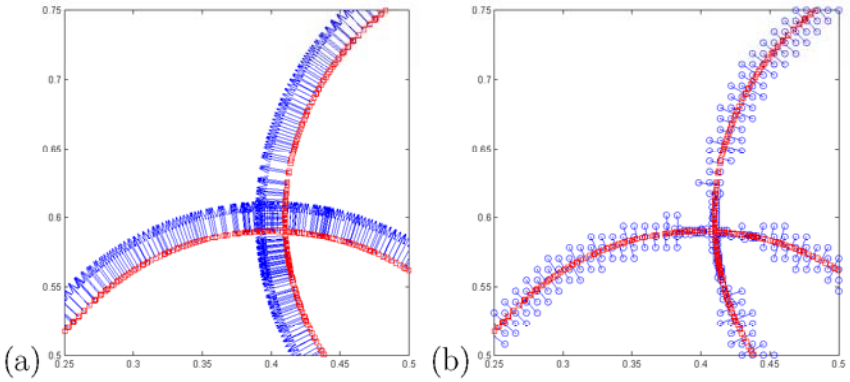

Figure 19: Crossing of two circles with motion in the outward normal direction with the coarsest resolution 129^2 and the finest resolution 1025^2. (a) Normals and (b) their corresponding active grid-point locations at the final time step.

this test case where we have two separated closed interfaces, we parameterize them using two disjoint intervals. For example, for one circle we use $\theta \in [-\pi, \pi)$ to parameterize all particles, and for the other interface we use $\theta \in [3\pi, 5\pi)$. The averaging algorithm for updating the parametrization described in Section 3.3.3 will maintain this condition. When we collect neighboring particles for local reconstruction, we check if their corresponding parameterizations come from the same interval of θ. If not, we will split the set of foot-points into groups and locally reconstruct each segment individually. Local mesh refinement is used in the region where two interfaces are close. Figure 18 shows the evolution of the interfaces at various time. As seen clearly, each circle expands independently without any interference from the other interface. Similar to the viscosity solution, we also plot the active grid points, foot-points and their corresponding normal vectors, in figure 19.

4.5 Motions by Mean Curvature

Curvature is an important second order geometric quantity that appears in many moving interface problems. Here we use the standard motion by mean curvature to test our algo-

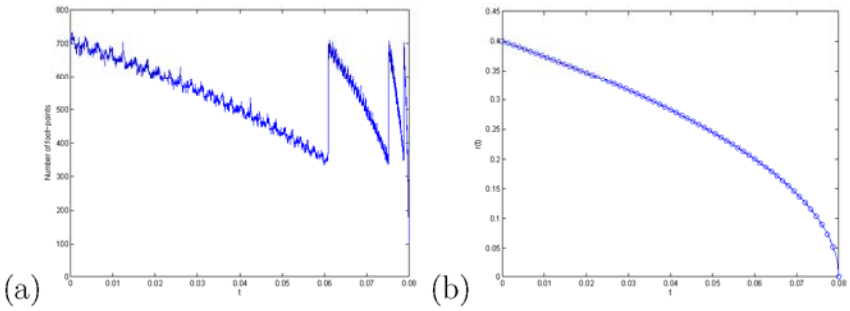

Figure 20: Motion by mean curvature of a circle with the coarsest resolution 129^2 and the finest resolution 1025^2. (a) The number of foot-points and (b) the change in the radius of the circle.

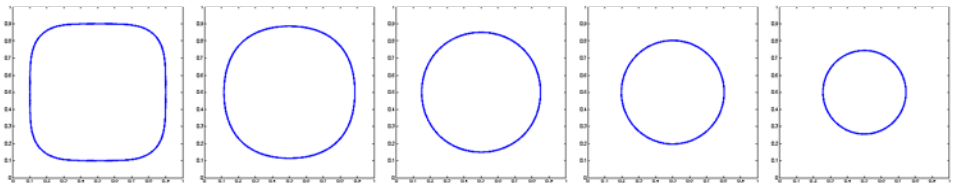

Figure 21: Motion by mean curvature of a square with the coarsest resolution 129^2 and the finest resolution 1025^2.

rithm. Two initial profiles, a circle and a square, are chosen. The circle is initially centered at $(0.5, 0.5)$ with radius 0.4. The square is centered at $(0.5, 0.5)$ with side 0.8. We solve the evolutions of these two interfaces up to $t = 0.08$ using the time step restriction $\Delta t = 0.5\Delta x^2$.

The radius of the circle can be analytically calculated and it is given by $r(t) = \sqrt{0.4^2 - 2t}$. At $t = 0.08$, the circle will theoretically disappear and our algorithm also gives the same phenomena. In figure 20, we plot the change in both the total number of particles and also the mean radius of the circle. The computed mean radius (the circles) matches with the exact solution (solid line) very well. As the circle shrinks, the curvature of the circle will increase and the adaptivity in the underlying grid will refine at various times. This explains the sudden change in the total number of particles in figure 20(a).

In figure 21, we model the evolution of a square by motion of mean curvature at $t = 0.008, 0.024, \cdots, 0.072$. The corners of the square are smoothed out immediately and it asymptotically becomes a circle and then disappear.

4.6 Motion of a Codimension-Two Object

Geometric optics is an important class of asymptotic approximation to wave propagation. In the high frequency regime, one can approximate the phase function by the eikonal equation

$$|\nabla T| = \frac{1}{c}\,. \tag{22}$$

To obtain the multivalued traveltime solution of this equation in two dimensions, one way is to solve the eikonal equation using the method of characteristics, a Lagrangian formulation [5], and reformulate the problem in the phase space. This type of methods inherits the intrinsic shortcoming of the Lagrangian method, in which we have non-uniform resolution in the desire computational domain. To overcome this drawback, a new field the so-called Eulerian geometrical optics [3] has been introduced. The idea is to represent the wavefront using a codimension-two object in the phase space $x, y, \theta \in \mathbb{R}^3$. The propagation of the wavefront is then translated to the motion of this curve in the three dimensional space governed by

$$\begin{aligned}
\frac{dx}{dt} &= c(x,y)\cos\theta\,,\\
\frac{dy}{dt} &= c(x,y)\sin\theta\,,\\
\frac{d\theta}{dt} &= c_x(x,y)\sin\theta - c_y(x,y)\cos\theta\,.
\end{aligned} \tag{23}$$

For example, the level set method has been used to simulate the wavefront evolution in the geometrical optic regime [12, 14]. However, in order to represent a codimension-two object, two level set functions defined in the phase space are used in [12], which can be quite expensive. Our grid based particle method can deal with this case easily. The algorithm is the same as before. Moreover, the total number of particles is proportional to the dimension of the interface, which is a one dimensional curve in this case.

In this example, we compute the collapse of an elliptical wavefront with the semimajor axis and the semiminor axis given by 0.6π and 0.3π, respectively. The velocity field is given by $c(x, y) = 1$. On the top row in figure 22, we show the evolution of the wavefront in the phase (x, y, θ) at $t = 0.0, 0.1, \cdots, 0.4$. To visualize the wavefront in the physical space, we project these solutions to (x, y) and plot them on the bottom row in figure 22.

4.7 Three Dimensional Examples

We now show a few examples in three-dimensions where the interface is a closed surface.

The first example is the motion under a single vortex. This is a challenging test case. The velocity field of this flow is given by

$$u_1(x,y,z) = 2\sin^2(\pi x)\sin(2\pi y)\sin(2\pi z)$$

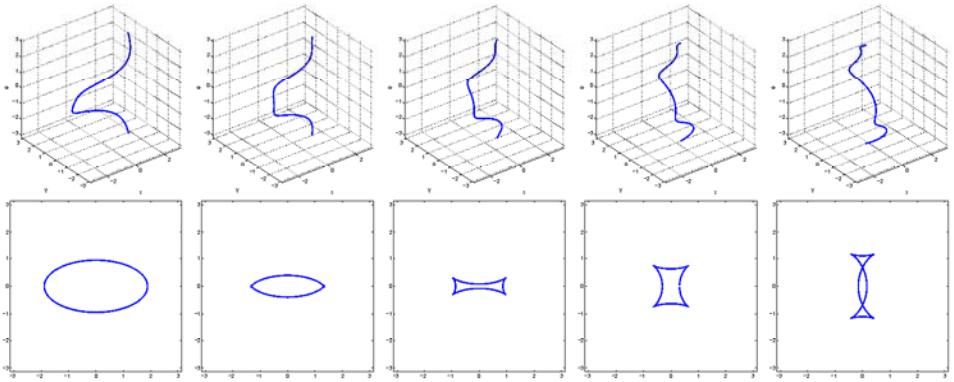

Figure 22: Geometrical Optics, $c(x, y) = 1$.

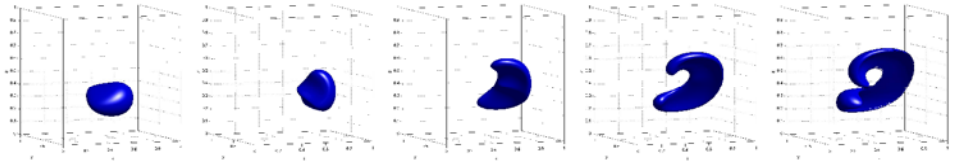

Figure 23: Motion under single vortex without the reversal motion using the resolution 181^3 at $t = 0.15, 0.30, \cdots, 0.75$.

$$\begin{aligned} u_2(x, y, z) &= -\sin(2\pi x)\sin^2(\pi y)\sin(2\pi z) \\ u_3(x, y, z) &= -\sin(2\pi x)\sin(2\pi y)\sin^2(\pi z)\,. \end{aligned} \tag{24}$$

The initial sphere is centered at $(0.35, 0.35, 0.35)$ with radius 0.15. In figure 23, we show our solution at $t = 0.15, \cdots, 0.75$ where the underlying grids are uniform with resolution 181^3, i.e. $\Delta x = 1/180$.

To model the flow with a reversal motion, we have multiplied the above flow with a factor $\cos(\pi t/T)$ where T is the final time. On figure 24, we used 181 uniform mesh points on each dimension. The mean distance of the sphere at the final time from the point $(0.35, 0.35, 0.35)$ is 0.1509 with the standard deviation 6.4007×10^{-3}.

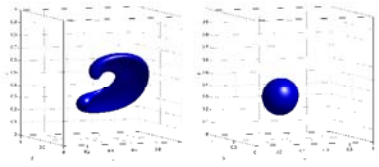

Figure 24: Motion under single vortex with the reversal motion using the resolution 181^3 at $t = 1.0$ and 2.0 with $T = 2.0$.

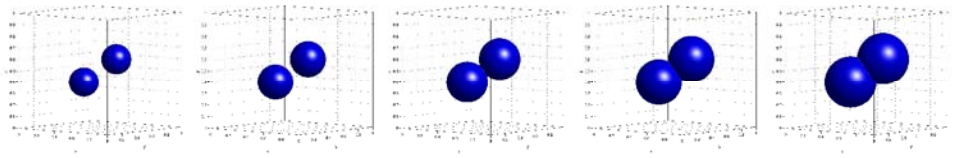

Figure 25: Motion of two spheres in the outward direction without topological change using the resolution 151^3.

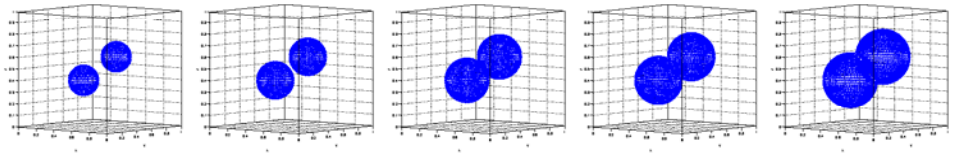

Figure 26: Motion of two spheres in the outward direction without topological change using the resolution 51^3.

As in the two dimensional case, our method can easily control topology when two interfaces meet allowing either multi-valued solutions or topological change. The next example generalizes the case in Section 4.4 where we have two spheres centered at (0.4,0.4,0.4) and (0.6,0.6,0.6), respectively, both with radii equal to 0.1. These two spheres expand in the normal direction with a uniform velocity in the normal direction and they meet at $t = 0.1(\sqrt{3} - 1) \simeq 0.0732$. Solutions at $t = 0.025, 0.05, \cdots, 0.125$ are shown in figures 25 to 28.

If the two interfaces pass through each other without any interference, as in the two-dimensional case this can be done easily using a global parametrization of the foot-points on the sphere. For foot-points from the sphere centered at (0.4,0.4,0.4), we parameterize them using the polar coordinates given by $(\theta, \phi) \in [-\pi/2, \pi/2) \times [-\pi, \pi)$. To distinguish from points from the second sphere, we parametrize foot-points from the second sphere in $(\theta, \phi) \in [3\pi/2, 5\pi/2) \times [-\pi, \pi)$. In the re-sampling step discussed in Section 3.3, one can check if neighboring particles have θ-parameterizations in the same interval. If not, we separate the particles into groups and locally reconstruct different interfaces individually. To show the multi-valued solution in this case, we plot all foot-points in figure 26. It can be seen clearly these foot-points hidden inside the spheres. As the time increases, each sphere expands with no interference from each others. Solutions using an underlying adaptive grid are shown on figure 27. Again the refinement is used to provide better resolution when two interfaces are getting close and crossing, which can be seen in figure 27.

Topological change in the sense of viscosity solution can be obtained too. One can check the normal directions in a small neighborhood centered at each grid point. If there is a conflict in the normal directions, we will remove the corresponding foot-points and de-activate the corresponding grid points. Figure 28 shows all the foot-points. Comparing to figure 26, there

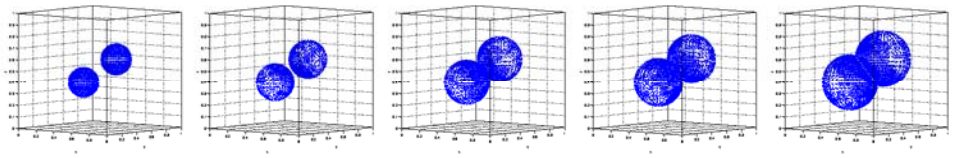

Figure 27: Motion of two spheres in the outward direction without topological change using the coarsest resolution 26^3 and finest resolution 51^3.

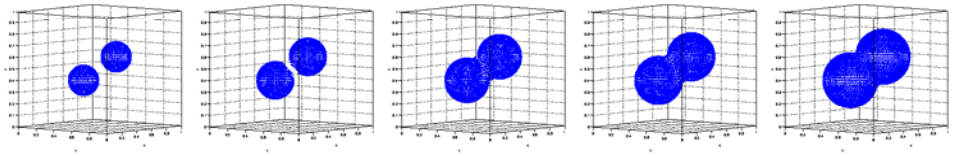

Figure 28: Motion of two spheres in the outward direction with topological change using the resolution 51^3.

is now no foot-points living inside the out-most interface. It should be emphasized that we do not require any parameterization in this example.

The last example is the motion by mean curvature. We first test on a sphere of initial radius 0.4 centered at (0.5,0.5,0.5). The underlying adaptive grid has the coarsest resolution of 33^3 and the finest resolution of 65^3. The comparison with the exact solution is shown in figure 29.

The more interesting example is the collapse of a dumbbell shape under the motion by mean curvature. We plot our solution in figure 30. Topological change in the handle region is nicely captured. We also show in figure 31 the locations of foot-points when topology changes.

5 Conclusion

We have proposed a novel grid-based particle method for interface problems which combines advantages from Lagrangian and Eulerian formulations. The method uses meshless particles to represent and track the moving interface. Using meshless particles avoids the difficulty of remeshing for usual tracking method. An underlying fixed mesh provides both Eulerian reference, neighborhood information, and sampling density for the meshless particles. No PDE is solved on the underlying mesh. Our method can naturally incorporate adaptivity and topology control. Future work includes extensions to motions of open surfaces and motions depending on global dynamics.

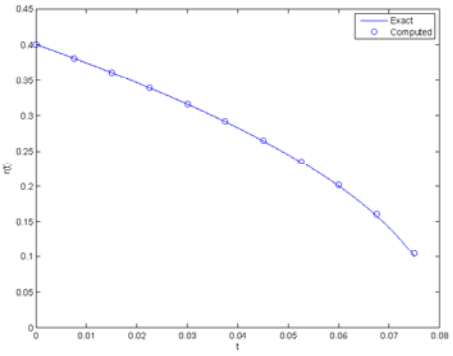

Figure 29: The motion by mean curvature of a sphere using an adaptive grid with the coarsest resolution 33^3 and finest resolution 65^3.

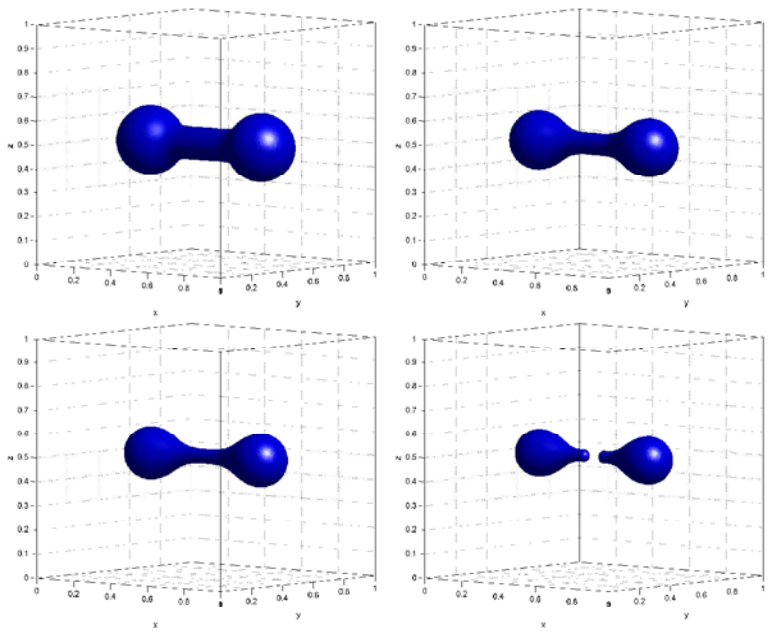

Figure 30: Motion by mean curvature of a dumbbell shape.

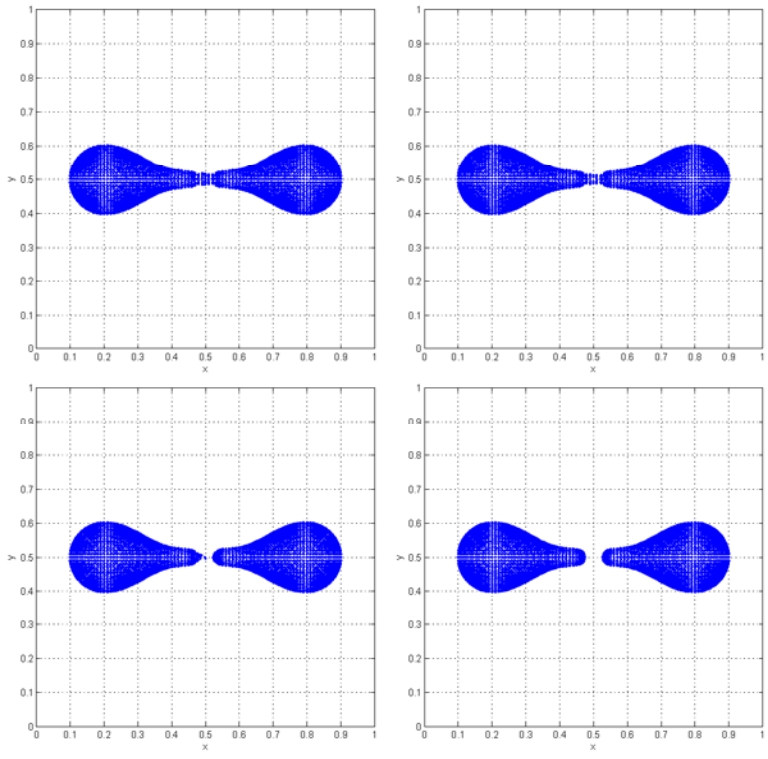

Figure 31: Motion by mean curvature of a dumbbell shape. Foot-point locations at the moment of topological change.

Acknowledgements

The work of Leung was supported in part by the ONR under Grant N00014-02-1-0090. The work of Zhao was supported in part by the NSF under Grand DMS-0513073, in part by the ONR under Grant N00014-02-1-0090 and also in part by DARPA under Grant N00014-02-1-0603.

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

[3] J. D. Benamou. An introduction to Eulerian geometrical optics (1992 - 2002). *J. Sci. Comp.*, 19:63–93, 2003.

[4] 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.

[5] V. Cerveny, I. A. Molotkov, and I. Psencik. *Ray method in seismology.* Univerzita Karlova press, 1977.

[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] 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.

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

[9] 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.

[10] R. LeVeque. High-resolution conservative algorithms for advection in incompressible flow. *SIAM J. Num. Anal.*, 33:627, 1996.

[11] C. Min and F. Gibou. A second order accurate level set method on non-graded adaptive cartesian grids. *J. Comput. Phys.*, In Press.

[12] S. Osher, L.-T. Cheng, M. Kang, H. Shim, and Y-H Tsai. Geometric optics in a phase space based level set and Eulerian framework. *J. Comput. Phys.*, 179:622–648, 2002.

[13] 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.

[14] J. Qian and S. Leung. A level set method for paraxial multivalued traveltimes. *J. Comput. Phys.*, 197:711–736, 2004.

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