A Grid Based Particle Method for Solving Partial Differential Equations on Evolving Surfaces and Modeling High Order Geometrical Motion

Shingyu Leung*,[a], John Lowengrub[b], Hongkai Zhao[b]

[a] *Department of Mathematics, Hong Kong University of Science and Technology, Clear Water Bay, Hong Kong.*
[b] *Department of Mathematics, University of California at Irvine, Irvine, CA 92697-3875, USA.*

Abstract

We develop numerical methods for solving partial differential equations (PDE) defined on an evolving interface represented by the grid based particle method (GBPM) recently proposed in *S. Leung and H.K. Zhao, A Grid Based Particle Method for Moving Interface Problems, J. Comput. Phys., 228:7706-7728, 2009.* In particular, we develop implicit time discretization methods for the advection-diffusion equation where the time step is restricted solely by the advection part of the equation. We also generalize the GBPM to solve high order geometrical flows including surface diffusion and Willmore-type flows. The resulting algorithm can be easily implemented since the method is based on meshless particles quasi-uniformly sampled on the interface. Furthermore, without any computational mesh or triangulation defined on the interface, we do not require remeshing or reparameterization in the case of highly distorted motion or when there are topological changes. As an interesting application, we study locally inextensible flows governed by energy minimization. We introduce tension force via a Lagrange multiplier determined by the solution to a Helmholtz equation defined on the evolving interface. Extensive numerical examples are also given to demonstrate the efficiency of the proposed approach.

Key words:

1. Introduction

The numerical solution of partial differential equations (PDEs) on evolving surfaces is a challenging problem as is simulating the motion of high order geometric evolution equations. The interface may stretch, deform and break apart. In the case of geometric motion, the velocity of the interface depends on high order derivatives of the interface position which poses challenges for accuracy and stability of the method. Such problems have application in the biological, physical and engineering sciences.

Various numerical methods have been proposed to solve PDEs on surfaces and geometric flows. One popular approach is based on explicit representation of surfaces by parametrizing or laying down a mesh on the surface (e.g., [13, 25, 26, 5, 2, 10, 9, 3]). When the surface is evolving, this type of approach is also called Lagrangian formulation since parametrization or meshes on the surface is tracked along the Lagrangian trajectory. One major issue of this approach is that it is difficult (especially in 3 dimensions), to maintain a smooth global parametrization or a quasi-uniform mesh for a moving interface with complicated geometry and dynamics involving large deformations or topological changes. As a result, the resulting algorithm

*Corresponding author.
Email addresses: masyleung@ust.hk (Shingyu Leung), lowengrb@math.uci.edu (John Lowengrub), zhao@math.uci.edu (Hongkai Zhao)

usually requires reparametrization/remeshing during the evolution, although a recent algorithm developed for geometric motion of surfaces has the potential to overcome this problem [3]. In addition, if the topology of the interface changes, then local interface surgery is required which can be highly problematic to perform in three dimensions. However, these Lagrangian type methods are relatively easy to implement and are efficient and accurate numerically.

An alternative approach is based on an implicit representation of surfaces using an auxilliary function on a fixed Eulerian mesh. For example, the level set method can be used for implicit representation while the PDEs on the surface, which is the zero level set of the level set function, are extended off the interface onto neighboring level sets (e.g., [4, 1, 44, 15, 43, 11]). There are three main advantages using an implicit formulation. First, there is no need to parametrize or triangulate the surface. Second, it is easy to handle topological changes. Third, the surface PDEs is extended off the surface to a corresponding PDE in a neighborhood of the zero level set on a fixed Eulerian mesh for which many standard numerical methods for PDEs can be applied. Examples of other recent Eulerian approaches include [31, 23], which are based on the closest point method, [17], which is based on volume of fluid method, [32] which is based on a grid-free particle level set method, and [28, 12, 37], which are based on diffuse interface method. Typically the Eulerian formulation has an increased computation cost compared to Lagrangian formulation due to embedding of the equation into one higher dimension. Also it is difficult to deal with open surfaces. For applications of Eulerian methods to geometric evolution equations, we refer the reader to the recent review [22] for references.

In [20], we have recently proposed a novel approach, the Grid Based Particle Method (GBPM), to represent and model an interface and its motion. The idea is to sample the interface by meshless and non-parametrized Lagrangian particles according to an underlying uniform or adaptive Eulerian mesh. This results in a quasi-uniform sampling of the interface. As the interface moves, we continuously update the location of these particles by solving ordinary differential equations (ODEs), rather than partial differential equations (PDEs). Using extra Lagrangian information defined on these sampling points, we can naturally capture topological changes such as merging or breakup of surfaces.

Unlike usual Lagrangian methods depending on surface parametrizations or meshes, the GBPM does not require any connectivity information among the particles. This feature allows one to easily add or delete particles, which is important for maintaining a consistent resolution of the surface as well as dealing with topological changes. Moreover, 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. This is useful for local construction of the surface and detection of interface collision or self-intersection. The Eulerian reference is continuously updated without using a PDEs approach. Even though it is not demonstrated in the current paper, an adaptive sampling of the interface can be achieved easily through local grid refinement of the underlying grid taking advantage of the fact that no PDE is solved on the grid. For a more detailed description, we again refer interested readers to some recent publications [20, 19].

We have successfully applied this technique to various velocity models in [20], including motion by mean curvature. We have also demonstrated that the method can be used to capture the viscosity solution or the multivalued solution when two interfaces come across each other. Numerous test examples have been shown in [20] that demonstrate the flexibility of the approach. We can also deal with curves/surfaces with high codimension as well as open curves/surfaces easily with this method [19, 21].

In this paper, we extend our previous work and use the GBPM to solve advection-diffusion equations on surfaces. In particular, we develop implicit methods that remove time step restrictions due to diffusion. We then further extend these techniques and consider high-order geometric flows including surface diffusion and Willmore flow [42]. As an interesting application of these techniques, we further impose a local inextensibility condition on an evolving interface. This constraint is important in many fields including lipid vesicle modeling in mathematical biology (e.g., [30, 8, 36, 40, 39, 35]), flexible fiber interactions (e.g., [38]) and robotics (e.g., [6]). To enforce local inextensibility, we incorporate a Lagrange multiplier which acts as the local tension of the evolving interface. This Lagrange multiplier satisfies a surface Helmholtz equation relating the geometry of the evolving interface with the tangential and the normal velocities of the unconstrained flow.

The paper is organized as follows. In Section 2, we first summarize the grid based particle method we have introduced in [20]. We explain how we generalize the method for solving PDE's on evolving surface in Section 3.1 to Section 3.3. With these techniques, we then model higher order motions of the interface including the motion by surface diffusion and the Willmore flow in Sections 3.4 to 3.5. In Section 4, we apply the techniques we have developed to simulate local inextensible flows. Section 5 shows examples to demonstrate the performance of our method.

2. Grid based particle method (GBPM)

In this section, we give a brief review of the grid based particle method. For a complete description of the algorithm, we refer interested readers to [20]. The interface is represented 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 gives 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 etc, which may 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 makes it easy to incorporate adaptivity 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.

This initial set-up is illustrated in figure 1 (a). We plot the underlying mesh using solid lines, all active grids are plotted using small circles and their associated foot-points are plotted using squares. 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. In practice, we do not necessarily use such a thick computational tube. Since the thicker the tube, the more the interface is over-sampled. Usually, we use $\gamma = 1.1h$ which already gives a relatively good sampling.

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 a 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 a local interface reconstruction as described in [20]. Here, we extend this approach to solve PDEs on surfaces and higher order geometric equations.

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 become clustered, and large error, when particles become far 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 (See figure 1 c-e). During

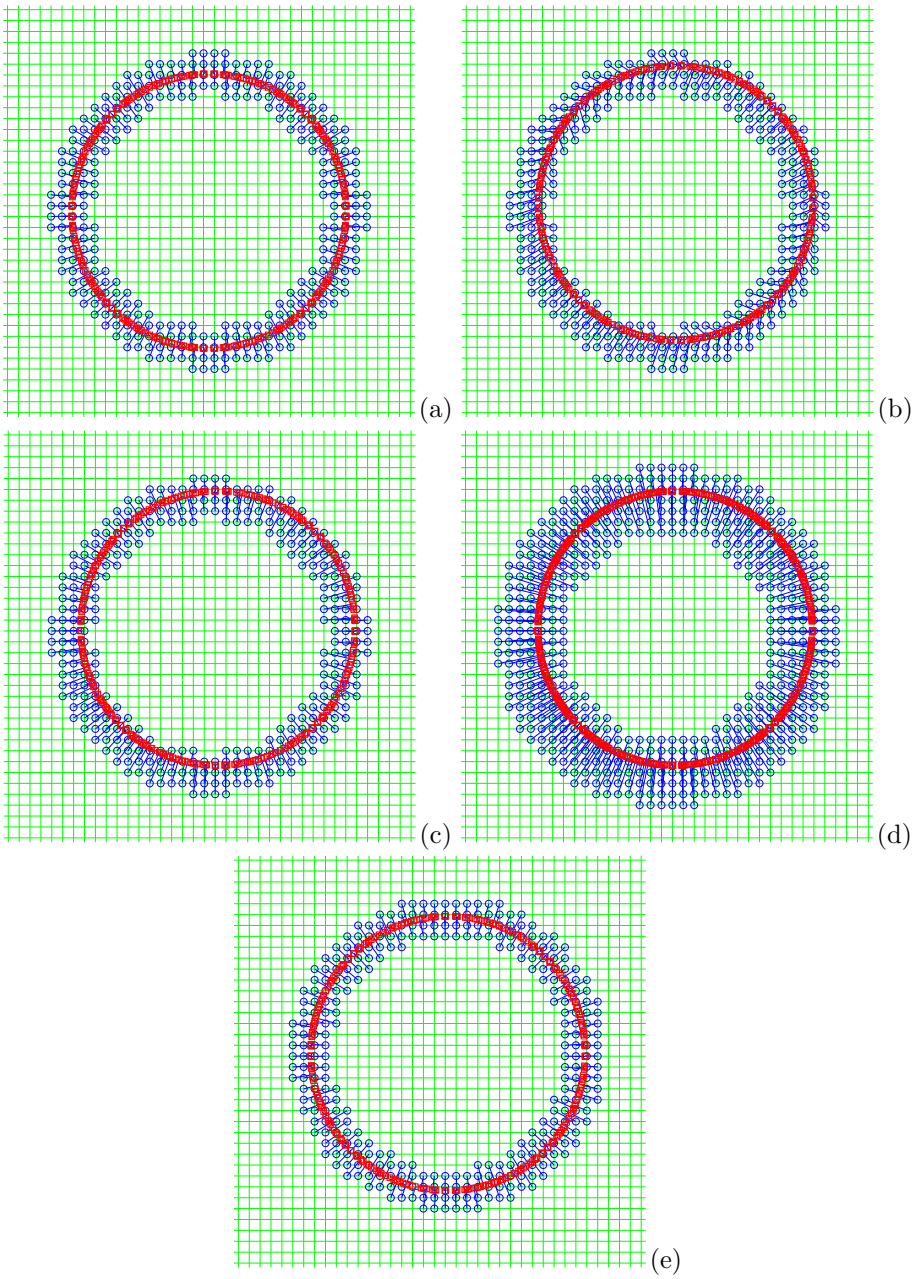

Figure 1: Grid Based Particle Method. From left to right: (a) Initialization, (b) after motion, (c) after re-sampling, (d) after activating new grid points with their foot-points, (e) after inactivating grid points with their foot points that are too far from the interface.

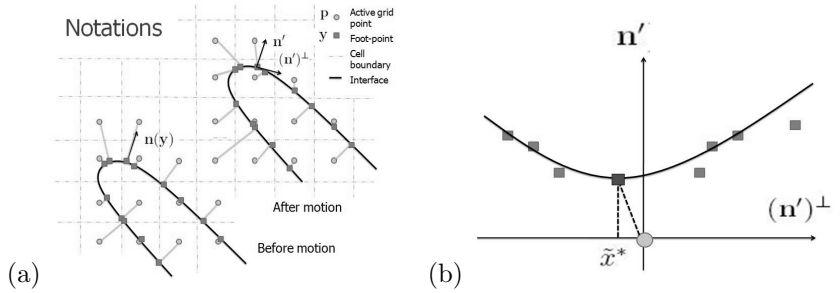

Figure 2: (a) Definition of a local coordinates and (b) the way how we determine the new foot-point using a local least squares reconstruction of the interface.

the resampling process, we locally reconstruct the interface, which involves communication among different particles on the interface. This local reconstruction also provides geometric and Lagrangian information at the recomputed foot-points 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, see figure 2 (a). Using this local reconstruction, we find the closest point from this active grid point to the local approximation of the interface, figure 2 (b). This gives the new foot-point 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 foot-point. For a detailed description, we refer interested readers to [20].

To end this section, we summarize the algorithm here.

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.
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. Time and spatial derivatives

In the first part of this section, we discuss how we apply the GBPM to solve a PDE on an evolving surface. Then in the second part we apply the GBPM to compute high order flow including motion by surface diffusion and Willmore flow.

3.1. Advection equation

In this section, we consider $f : \Sigma(t) \to \mathbb{R}$ defined on the surface $\Sigma(t)$ represented by quasi-uniformly distributed but meshless particles. Assuming that the function f is advected by the flow, we have

$$\frac{Df}{Dt} = 0\,, \tag{1}$$

where $D/Dt = \partial/\partial t + \mathbf{u}\cdot\nabla$ is the material derivative along any trajectory of a particle on the interface.

Solving this equation is relatively straightforward in our formulation. Since we are representing the interface $\Sigma(t)$ using particles, the corresponding f at each of these particles is constant during the motion. Denote $\mathbf{y}_i(t^n)$ the location of the foot-point associated to an active grid point $\mathbf{x}_i$ at the time $t = t^n$. According to the motion $\mathbf{u} = \mathbf{u}(\mathbf{y}_i)$, we solve the ODE

$$\frac{d\mathbf{y}(t)}{dt} = \mathbf{u}(\mathbf{y}(t))\,, \tag{2}$$

for $\mathbf{y}(t^{n+1})$ with the initial condition $\mathbf{y}(t^n) = \mathbf{y}_i(t^n)$. Denoting this location by $\mathbf{y}_i^*(t^n)$, we obtain the function value f after this motion phase

$$f(\mathbf{y}_i^*(t^n)) = f(\mathbf{y}_i(t^n))\,. \tag{3}$$

The more important step is the resampling phase. In this step, we locally reconstruct the interface and then recompute the foot-point associated to a particular activated grid point. Thus, we have to update the function value at the new foot-point location $\mathbf{y}_i(t^{n+1})$ using the function values at $\mathbf{y}_j^*(t^n)$. This is an approximation problem. Given $f(\mathbf{y}_j^*(t^n))$ for $j = 1, \cdots, m$, one approximates $f(\mathbf{y}_i(t^{n+1}))$. In this paper, we consider the local coordinates at $\mathbf{y}_i^*(t^n)$ and then the approximation can be easily done on the tangent plane at the point $\mathbf{y}_i^*(t^n)$. Here we discuss the two dimensional formulation. Extension to higher dimensions is straightforward. In the local coordinate $\{(\mathbf{n}')^\perp, \mathbf{n}'\}$ with $\mathbf{n}'$ the normal vector associated to the foot-point $\mathbf{y}_i(t^n)$, we express the function f in terms of this local coordinates,

$$f(x, y) = f(\tilde{x}, \tilde{y}(\tilde{x})) = f(\tilde{x}) \tag{4}$$

where $\tilde{y}(\tilde{x})$ is the least squares polynomial we obtained to locally approximate the interface. Let $\tilde{x}^*$ be the minimizer which gives the location for the new foot-point, we have the following approximation problem. Given m data points $(\tilde{x}_j, f(\tilde{x}_j))$ for $j = 1, \cdots, m$, we want to approximate $f(\tilde{x}^*)$. Various methods could be used. In the current formulation, we again use the least squares technique and determine a polynomial $\tilde{f}$ approximating the function f according to the data points. Then the function value at the new foot-point is given by $\tilde{f}(\tilde{x}^*)$, i.e.

$$f(\mathbf{y}_i(t^{n+1})) = \tilde{f}(\tilde{x}^*)\,. \tag{5}$$

Although this is not conservative, it is found to give very good results.

To simplify the notation in the following sections, we define the following operators for solving the advection equation. Let $f(\Sigma^n, \mathbf{y}^n, t^n)$ be the function evaluated at the Lagrangian particles $\mathbf{y}^n$ which samples the surface Σ^n at t^n. We define the advection operator $\mathcal{A}_{\Delta t}$ by

$$\mathcal{A}_{\Delta t} f(\Sigma^n, \mathbf{y}^n, t^n) = f(\Sigma^{n+1}, \mathbf{y}^*, t^*)\,. \tag{6}$$

The superscript $(*)$ accounts for the fact that the sampling particles $\mathbf{y}^*$ are locations of the foot-points $\mathbf{y}^n$ right after motion. These points are in general **not** the corresponding closest points from the underlying grid anymore. But since these points $\mathbf{y}^*$ still sample the interface at $t = t^{n+1}$, we denote the surface by Σ^{n+1}. To restore the quasi-uniform sampling of the interface, we apply the resampling step as described in Section 2. The function value at these new sampling points $\mathbf{y}^{n+1}$ are obtained by interpolation, denoted by the operator $\mathcal{I}$,

$$f(\Sigma^{n+1}, \mathbf{y}^{n+1}, t^{n+1}) = \mathcal{I} f(\Sigma^{n+1}, \mathbf{y}^*, t^*)\,. \tag{7}$$

To sum up, the solution of the advection equation on the evolving surface is

$$f(\Sigma^{n+1}, \mathbf{y}^{n+1}, t^{n+1}) = \mathcal{I}\mathcal{A}_{\Delta t} f(\Sigma^n, \mathbf{y}^n, t^n)\,. \tag{8}$$

3.2. Surface Laplacian

Consider a surface Σ parametrized by (s_1, s_2). The surface Laplacian of a function $f : \Sigma \to \mathbb{R}$ is given by

$$\Delta_\Sigma f = \sum_{i,j=1}^{2} \frac{1}{\sqrt{g}} \frac{\partial}{\partial s_i} \left(\sqrt{g} g^{ij} \frac{\partial f}{\partial s_j} \right) \tag{9}$$

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

$$g_{ij} = \frac{\partial \mathbf{X}}{\partial s_i} \frac{\partial \mathbf{X}}{\partial s_j} , \tag{10}$$

with the surface Σ is given by $\mathbf{X}(s_1, s_2)$.

To locally parametrize the surface in our formulation, we use again the local coordinate system of the tangent plane. At each particle, we translate and rotate the coordinate system according to its normal vector $\mathbf{n}$ so that the foot-point is now at the origin of the new coordinate. The local parametrization of the surface is then given by the tangent plane $\{\mathbf{t}_1, \mathbf{t}_2\}$ where $\mathbf{n}$, $\mathbf{t}_1$ and $\mathbf{t}_2$ are orthogonal. If we use a second order polynomial to locally approximate the interface, i.e.

$$\mathbf{X}(s_1, s_2) = \sum_{i=0}^{2} \sum_{0 \le i+j \le 2} a_{i,j} s_1^i s_2^j , \tag{11}$$

the metric tensor $[g_{ij}]$ in this coordinate system can be found via

$$[g_{ij}] = \begin{pmatrix} 1 + \left(\frac{\partial \mathbf{X}}{\partial s_1} \right)^2 & \frac{\partial \mathbf{X}}{\partial s_1} \frac{\partial \mathbf{X}}{\partial s_2} \\ \frac{\partial \mathbf{X}}{\partial s_1} \frac{\partial \mathbf{X}}{\partial s_2} & 1 + \left(\frac{\partial \mathbf{X}}{\partial s_2} \right)^2 \end{pmatrix} , \tag{12}$$

where

$$\begin{aligned} \frac{\partial \mathbf{X}}{\partial s_1} &= a_{1,0} + 2a_{2,0} s_1 + a_{1,1} s_2 , \\ \frac{\partial \mathbf{X}}{\partial s_2} &= a_{0,1} + 2a_{0,2} s_2 + a_{1,1} s_1 . \end{aligned} \tag{13}$$

Next, since the function f is defined only on the foot-points $\mathbf{y}$, we also fit f using a local least squares quadratic polynomial

$$f(s_1, s_2) = \sum_{i=0}^{2} \sum_{0 \le i+j \le 2} b_{i,j} s_1^i s_2^j . \tag{14}$$

The corresponding derivatives in equation (9) can then be approximated by the derivatives of this local least squares approximation.

A similar approximation can be found in [41] where the authors proposed an approach to find the surface Laplacian using local polynomial estimation. However, their approach is applied on a fixed surface which is not evolving. Moreover, all calculations are done on the tangent plane at each of the sampling points, i.e. the local metric $[g_{ij}]$ is flat, which makes the overall approximation of low order.

3.3. Advection-Diffusion equation

In this section, we apply the techniques for computing the surface Laplacian to solve the advection-diffusion equation. We develop explicit and implicit schemes. Numerical details are also discussed.

3.3.1. An explicit scheme

The easiest way to solve the advection-diffusion equation is to apply operator splitting techniques and to compute each operator explicitly. We first consider the advection part given by

$$\frac{Df}{Dt} = 0\,, \tag{15}$$

with the initial condition $f(\mathbf{y}_i(t^n))$ defined on the interface $\Sigma(t^n)$. This can be solved by the techniques described in Section 3.1. Once we have obtained the solution $f(\mathbf{y}_i(t^{n+1}))$ defined on the surface $\Sigma(t^{n+1})$, we solve the following diffusion equation

$$\frac{\partial f}{\partial t} = \Delta_{\Sigma(t^{n+1})} f\,, \tag{16}$$

where the surface $\Sigma(t^{n+1})$ remains unchanged in this step. Using the forward Euler method, we have

$$f^{n+1} = [I + \Delta t \Delta_{\Sigma(t^{n+1})}] f^n\,. \tag{17}$$

We denote the solution from this diffusion equation on the fixed surface Σ^{n+1} by

$$f(\Sigma^{n+1}, \mathbf{y}^{n+1}, t^{n+1}) = \mathcal{D}^{n+1}_{+,\Delta t} \mathcal{I} \mathcal{A}_{\Delta t} f(\Sigma^n, \mathbf{y}^n, t^n)\,. \tag{18}$$

The operators $\mathcal{I}$ and $\mathcal{A}_{\Delta t}$ are defined in the previous section. The superscript $(n+1)$ in the operator $\mathcal{D}$ represents the surface Σ^{n+1}. The subscript $+$ reflects the plus sign in the operator $[I + \Delta t \Delta_{\Sigma(t^{n+1})}]$, i.e.

$$\mathcal{D}^{n+1}_{+,\Delta t} = [I + \Delta t \Delta_{\Sigma(t^{n+1})}]\,. \tag{19}$$

This explicit scheme can be easily implemented but the corresponding CFL condition is restrictive: the time step $\Delta t = O(h^2)$, where h is the underlying fixed grid size (not the interparticle distance).

3.3.2. Implicit schemes

To relax the time step restriction, we will discuss several implicit schemes which will result in a CFL condition of $\Delta t = O(h)$ which solely depends on the advection part.

We consider again the numerical discretization of the surface Laplacian we described in Section 3.2. For simplicity, we consider here only the two dimensional case, it is relatively straightforward to generalize the approach to higher dimensions. For a given surface Σ^n, the surface Laplacian (9) can be written as a linear combination of derivatives on the surface

$$\Delta_{\Sigma^n} f = \alpha_1 f_s + \alpha_2 f_{ss} \tag{20}$$

for some α_i's depending only on the coefficients of the local quadratic reconstruction

$$\mathbf{f}(s) = a_0 + a_1 s + a_2 s^2\,. \tag{21}$$

As described in Section 3.2, associated to each active grid point, we locally approximate the function f by a local least squares quadratic polynomial

$$f(s) = b_0 + b_1 s + b_2 s^2\,. \tag{22}$$

In particular, given the data (s_i, f_i) for $i = 1, \cdots, m$, these coefficients b_i's in the local reconstruction are determined by solving the normal equation

$$\begin{pmatrix} m & \sum s_i & \sum s_i^2 \\ \sum s_i & \sum s_i^2 & \sum s_i^3 \\ \sum s_i^2 & \sum s_i^3 & \sum s_i^4 \end{pmatrix} \begin{pmatrix} b_0 \\ b_1 \\ b_2 \end{pmatrix} = \begin{pmatrix} \sum f_i \\ \sum s_i f_i \\ \sum s_i^2 f_i \end{pmatrix}\,, \tag{23}$$

which gives

$$\begin{pmatrix} b_0 \\ b_1 \\ b_2 \end{pmatrix} = \begin{pmatrix} \beta_{1,1} & \beta_{1,2} & \beta_{1,3} \\ \beta_{2,1} & \beta_{2,2} & \beta_{2,3} \\ \beta_{3,1} & \beta_{3,2} & \beta_{3,3} \end{pmatrix} \begin{pmatrix} \sum_i f_i \\ \sum_i s_i f_i \\ \sum_i s_i^2 f_i \end{pmatrix} . \tag{24}$$

This implies that coefficients b_i depend linearly on the function values f_i. Moreover, assuming that we are computing the surface Laplacian at $s = s^*$ with $s^* = s_i$ for some $i = 1, \cdots, m$, we can express the surface Laplacian (9) as a linear combination of f_i's

$$\Delta_{\Sigma^n} f = \sum_{i \neq *} \gamma_i f_i + \gamma^* f^* , \tag{25}$$

for some $\gamma^* = \gamma^*(s^*, \beta_{k_1,k_2}, \alpha_{k_3})$ and β_{k_1,k_2} depends on s_i, i.e., locations of footpoints.

This linear relationship is important for developing iterative methods for implicit schemes. We first consider the following implicit scheme based on a simple operator splitting. We follow the explicit scheme we have just developed in the previous section, $\mathcal{D}^{n+1}_{+,\Delta t}\mathcal{I}\mathcal{A}_{\Delta t}$, and replace the diffusion part by a backward Euler scheme. This implies

$$[I - \Delta t \Delta_{\Sigma(t^{n+1})}] f^{n+1} = f^n , \tag{26}$$

or

$$f^{n+1} = [I - \Delta t \Delta_{\Sigma(t^{n+1})}]^{-1} f^n . \tag{27}$$

In the operator form, we denote the resulting numerical scheme by

$$f(\Sigma^{n+1}, \mathbf{y}^{n+1}, t^{n+1}) = (\mathcal{D}^{n+1}_{-,\Delta t})^{-1} \mathcal{I}\mathcal{A}_{\Delta t} f(\Sigma^n, \mathbf{y}^n, t^n) . \tag{28}$$

Unfortunately, discretizing equation (26) does not result in a diagonally dominant system of linear equations for arbitrary Δt. We demonstrate this using a very simple case where the surface is flat and the grid points are evenly distributed with uniform spacing of $\Delta x = 1$. We first approximate the function $f(x)$ locally using the least squares fitting, this gives $f(x) = b_0 + b_1 x + b_2 x^2$ where

$$b_2 = 0.14286 f_{-2} - 0.07143 f_{-1} - 0.14286 f_0 - 0.07143 f_1 + 0.14286 f_2 . \tag{29}$$

For sufficiently small Δt, we have $1 + 2 \cdot 0.14286 \Delta t > 4(0.14286 + 0.07143)\Delta t$. This implies that the matrix we use to approximate the operator $[I - \Delta t \Delta_{\Sigma(t^{n+1})}]$ is diagonally dominant only conditionally in Δt. The resulting system of linear equations can still be inverted using simple Jacobi/Gauss-Seidel iteration for sufficiently small Δt. In general, however, the condition will not be guaranteed for arbitrary Δt or local geometry.

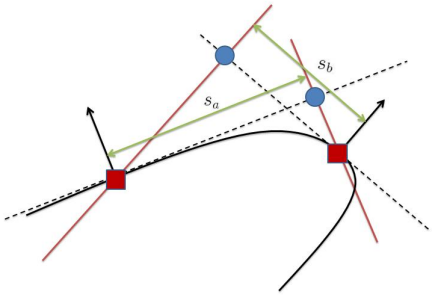

Figure 3: Discretization of the surface Laplacian gives an nonsymmetric matrix. See text for details.

Another numerical difficulty is that the above discretization gives a nonsymmetric matrix for general surfaces. Consider figure 3 where we have two foot-points (red square) on the interface (black solid line).

We plot the local coordinates used in the local reconstruction using an arrow and the dashed lines. We draw the projection of each foot-points to these two local coordinate systems using blue circle. Depending on the interface, s_a will generally be different from s_b. Since the coefficients of the local reconstruction depend on the values s_i in the stencil, the contribution from the function value at the left foot-point to the surface Laplacian at the right foot-point will be different from the vise-versa relation. This implies that the matrix we are inverting is in general nonsymmetric.

Numerically, we propose the following two strategies in solving for f^{n+1} in equation (26). As described above, for sufficiently small Δt, the operator $[I - \Delta t \Delta_{\Sigma(t^{n+1})}]$ is still diagonally dominant even though nonsymmetric. In this case, we can still invert it using simple Gauss-Seidel iteration. To simplify the notation, we introduce the variable $g = f^n$. For each foot-point (using the label *), we iterate the following expression for $f^{*,k}$

$$(1 - \gamma^* \Delta t) f^{*,k} = g^* + \Delta t \sum_{i \neq *} \gamma_i f_i^{**} , \tag{30}$$

for all foot-points and $k = 1, \cdots$. The superscript $**$ denotes the most-updated value for f_i, i.e. it represents either k or $k-1$. The initial guess of the iteration can be the solution from the forward Euler operator so that

$$f^{*,0} = \mathcal{D}^{n+1}_{+,\Delta t} \mathcal{I}\mathcal{A}_{\Delta t} f(\Sigma^n, \mathbf{y}^n, t^n) . \tag{31}$$

The iteration converges very rapidly and it usually takes less than 10 iterations.

For other cases where we do not have a good initial guess of the solution, or the Gauss-Seidel iteration does not converge, or we want an unconditionally stable scheme for the diffusion part, we simply solve the system of linear equations using the singular value decomposition (SVD). Iterative solvers including Quasi-Minimal Residual method (QMR) [14] might give faster convergence, but it is challenging to find a good preconditioner for such nonsymmetric matrix and so this approach will not be studied here.

This implicit scheme described above gives a CFL condition $\Delta t = O(h)$ but still is only first order accurate. To increase the accuracy in solving the diffusion equation, we can replace the backward Euler step by a Crank-Nicolson method

$$f^{n+1} = \left[I - \frac{\Delta t}{2} \Delta_{\Sigma(t^{n+1})} \right]^{-1} \left[I + \frac{\Delta t}{2} \Delta_{\Sigma(t^{n+1})} \right] f^n . \tag{32}$$

Numerically, the implementation of this iteration method is similar to (30) except that we replace Δt by $\Delta t/2$ and also define g by $\left[I + \frac{\Delta t}{2} \Delta_{\Sigma(t^{n+1})} \right] f^n$. Even though the Crank-Nicolson scheme is of higher order, the overall scheme

$$f(\Sigma^{n+1}, \mathbf{y}^{n+1}, t^{n+1}) = (\mathcal{D}^{n+1}_{-,\Delta t/2})^{-1} \mathcal{D}^{n+1}_{+,\Delta t/2} \mathcal{I}\mathcal{A}_{\Delta t} f(\Sigma^n, \mathbf{y}^n, t^n) \tag{33}$$

is still first order accurate due to the simple operator splitting.

To obtain higher order accuracy for the overall scheme, we need to split the operators more accurately. One way is to use the Strang splitting which nicely cancels the first order error due to the simple splitting. There are two resulting numerical schemes which are symmetric to each other by switching the corresponding operators,

$$f(\Sigma^{n+1}, \mathbf{y}^{n+1}, t^{n+1}) = \left[\mathcal{I}\mathcal{A}_{\Delta t/2} \right] \left[(\mathcal{D}^{n+1/2}_{-,\Delta t/2})^{-1} \mathcal{D}^{n+1/2}_{+,\Delta t/2} \right] \left[\mathcal{I}\mathcal{A}_{\Delta t/2} \right] f(\Sigma^n, \mathbf{y}^n, t^n) \tag{34}$$

and

$$f(\Sigma^{n+1}, \mathbf{y}^{n+1}, t^{n+1}) = \left[(\mathcal{D}^{n+1}_{-,\Delta t/4})^{-1} \mathcal{D}^{n+1}_{+,\Delta t/4} \right] \left[\mathcal{I}\mathcal{A}_{\Delta t} \right] \left[(\mathcal{D}^{n}_{-,\Delta t/4})^{-1} \mathcal{D}^{n}_{+,\Delta t/4} \right] f(\Sigma^n, \mathbf{y}^n, t^n) . \tag{35}$$

To speed up the computations, we combine adjacent operators in the time marching methods and obtain the following schemes,

$$\begin{aligned} f(\Sigma^{n+1}, \mathbf{y}^{n+1}, t^{n+1}) \quad = \quad & \left[\mathcal{I}\mathcal{A}_{\Delta t/2} \right] \left[(\mathcal{D}^{n+1/2}_{-,\Delta t/2})^{-1} \mathcal{D}^{n+1/2}_{+,\Delta t/2} \right] \left[\mathcal{I}\mathcal{A}_{\Delta t} \right] \cdots \\ & \left[(\mathcal{D}^{3/2}_{-,\Delta t/2})^{-1} \mathcal{D}^{3/2}_{+,\Delta t/2} \right] \left[\mathcal{I}\mathcal{A}_{\Delta t} \right] \left[(\mathcal{D}^{1/2}_{-,\Delta t/2})^{-1} \mathcal{D}^{1/2}_{+,\Delta t/2} \right] \left[\mathcal{I}\mathcal{A}_{\Delta t/2} \right] f(\Sigma^0, \mathbf{y}^0, t^0) \end{aligned} \tag{36}$$

and

$$f(\Sigma^{n+1},\mathbf{y}^{n+1},t^{n+1}) = \left[(\mathcal{D}^{n+1}_{-,\Delta t/4})^{-1}\mathcal{D}^{n+1}_{+,\Delta t/4}\right][\mathcal{I}\mathcal{A}_{\Delta t}]\left[(\mathcal{D}^{n}_{-,\Delta t/2})^{-1}\mathcal{D}^{n}_{+,\Delta t/2}\right]\cdots$$
$$\left[(\mathcal{D}^{2}_{-,\Delta t/2})^{-1}\mathcal{D}^{2}_{+,\Delta t/2}\right][\mathcal{I}\mathcal{A}_{\Delta t}]\left[(\mathcal{D}^{0}_{-,\Delta t/4})^{-1}\mathcal{D}^{0}_{+,\Delta t/4}\right] f(\Sigma^0,\mathbf{y}^0,t^0)\,. \quad (37)$$

3.3.3. A fully implicit Crank-Nicolson scheme

In the previous Section, we have introduced various implicit schemes based on operator splitting, including the explicit scheme (18), the backward Euler scheme (28), the Crank-Nicolson scheme (33) and also two schemes based on Strang Splitting (36) and (37). It is also desirable to develop a Crank-Nicolson scheme with **no** operator splitting, e.g.

$$\frac{f^{n+1}-f^n}{\Delta t} = \frac{1}{2}\left[\Delta_{\Sigma(t^n)}f^n + \Delta_{\Sigma(t^{n+1})}f^{n+1}\right]. \quad (38)$$

Note that that different sets of foot-points are used to sample the two surfaces $\Sigma(t^n)$ and $\Sigma(t^{n+1})$. To solve f on the same set of foot-points, we propose the following strategy. We introduce a new variable $F=\Delta_\Sigma f$ which is advected and is interpolated at the new foot-points as described in Section 3.1. The resulting scheme can be written as

$$F(\Sigma^n,\mathbf{y}^n,t^n) = \Delta_{\Sigma(t^n)}f(\Sigma^n,\mathbf{y}^n,t^n)$$
$$f(\Sigma^{n+1},\mathbf{y}^{n+1},t^{n+1}) = (\mathcal{D}^{n+1}_{-,\Delta t/2})^{-1}\left[\mathcal{I}\mathcal{A}_{\Delta t}f(\Sigma^n,\mathbf{y}^n,t^n) + \frac{\Delta t}{2}\mathcal{I}\mathcal{A}_{\Delta t}F(\Sigma^n,\mathbf{y}^n,t^n)\right]. \quad (39)$$

3.4. Smoothing

For motions depending on high order derivatives of geometrical quantities, such as the surface Laplacian of the mean curvature, it may be necessary to apply smoothing to filter out high frequency modes in the solution during the evolution to maintain stability of the numerical scheme. One simple way is to smooth using a Gaussian-type filter (e.g., [24]). Given $f(\tilde{x}_i)$ with $\tilde{x}_i$ is the coordinate in the $(\mathbf{n}')^\perp$-direction. We compute the weighted average quantity on the point $\tilde{x}=\tilde{x}^*$ by

$$f(\tilde{x}^*) = \frac{\sum_i w(\tilde{x},\tilde{x}^*)f(\tilde{x}_i)}{\sum_i w(\tilde{x},\tilde{x}^*)}, \quad (40)$$

where

$$w(\tilde{x},\tilde{x}^*) = \exp\left(\frac{-(\tilde{x}-\tilde{x}^*)^2}{2\sigma^2}\right) \quad (41)$$

and $\sigma = O(h)$.

3.5. Surface diffusion flow and Willmore flow

In this section, we consider interface motion that depends on high order geometry quantities. We will consider surface diffusion (e.g., [27, 7, 34]) and the related Willmore flow [42]. For surface diffusion, we have a family of compact, closed hypersurfaces $\Sigma(t)$ satisfying the evolution equation

$$v_n = \Delta_{\Sigma(t)}H \quad (42)$$

with $\Sigma(0)=\Sigma_0$, the initial surface, and v_n is the normal velocity defined on the surface $\Sigma(t)$. The quantity H is the total curvature of $\Sigma(t)$. Let $A(t)$ and $V(t)$ be the surface area of the surface $\Sigma(t)$ and its enclosed volume, respectively, then

$$\frac{1}{2}\frac{dA(t)}{dt} = \int_{\Sigma(t)} v_n H ds = \int_{\Sigma(t)} H\Delta_{\Sigma(t)}H ds = -\int_{\Sigma(t)} |\nabla_{\Sigma(t)}H|^2 ds \le 0$$
$$\frac{dV(t)}{dt} = \int_{\Sigma(t)} v_n ds = 0\,, \quad (43)$$

where $\nabla_{\Sigma(t)}$ is the surface gradient. This implies that the surface area of the surface $\Sigma(t)$ is non-increasing in time while the enclosed volume remains unchanged.

A related geometric flow is the Willmore flow [42], which involves a fourth order nonlinear evolution equation. The Willmore energy for a compact closed hypersurface is

$$E(\Sigma) = \frac{1}{2}\int_{\Sigma} H^2 ds\,. \tag{44}$$

In addition to its intrinsic mathematical interest, this model has been used to characterize the energy of vesicles (e.g., [16, 33]). Any critical point of this functional is called a Willmore surface, which satisfies the equation

$$\Delta_{\Sigma} H + 2H(H^2 - K) = 0\,, \tag{45}$$

where K is the Gaussian curvature. To determine such a surface, one may evolve a compact surface to the Willmore surface via Willmore flow

$$\Sigma(t)' = -\nabla E(\Sigma) \tag{46}$$

which implies the motion by the following normal velocity

$$v_n = \Delta_{\Sigma} H + 2H(H^2 - K)\,. \tag{47}$$

Both of these flows depend on the surface Laplacian of the mean curvature. To numerically approximate this quantity at each foot-point, we first interpret the mean curvature as a function defined on the interface. The corresponding numerical quantity at each foot-point can be approximated from the local reconstruction. For instance, in two dimensions for example, the mean curvature at the foot-point can be approximated by the local quadratic reconstruction $f(s)$ using

$$\kappa(s^*) = \left.\frac{f''(s)}{\{1 + [f'[s]]^2\}^{3/2}}\right|_{s=s^*}\,. \tag{48}$$

For more details on computing this quantity at all foot-points, we refer interested readers to [20]. With the mean curvature defined at all foot-points, we can then compute the surface Laplacian as in (9) by taking $f = H$.

Since the underlying PDE is fourth order, solving the resulting evolution equation using an explicit scheme leads to a CFL time step restriction of $\Delta t = O(h^4)$. Here we describe a semi-implicit method to solve the surface diffusion motion and the Willmore flow.

We consider the following relations between the normal $\mathbf{n}$ and the tangent vector $\mathbf{t}$ on the interface

$$\begin{aligned} \mathbf{t}_s &= -\kappa \mathbf{n} \\ \mathbf{n}_s &= \kappa \mathbf{t}\,. \end{aligned} \tag{49}$$

The fourth derivative of the interface is therefore given by

$$\begin{aligned} \mathbf{y}_{ssss} &= \mathbf{t}_{sss} \\ &= (-\kappa \mathbf{n})_{ss} \\ &= (-\kappa_s \mathbf{n} - \kappa^2 \mathbf{t})_s \\ &= (-\kappa_{ss} + \kappa^3)\mathbf{n} - 3\kappa\kappa_s \mathbf{t} \\ &= (-\kappa_{ss} - \kappa^2 \mathbf{y}_{ss}\cdot\mathbf{n})\mathbf{n} - 3\kappa\kappa_s \mathbf{t}\,. \end{aligned} \tag{50}$$

This implies that the velocity induced by surface diffusion can be written as

$$\begin{aligned} \mathbf{v} &= -\kappa_{ss}\mathbf{n} \\ &= \mathbf{y}_{ssss} - \kappa^3 \mathbf{n} + 3\kappa\kappa_s \mathbf{t} \\ &= \mathbf{y}_{ssss} + \kappa^2 \mathbf{y}_{ss} + \mathbf{v}\cdot\mathbf{t}\,. \end{aligned} \tag{51}$$

Since the tangential velocity on the interface will not change the corresponding shape, we simply advect the foot-points numerically using the velocity

$$\mathbf{v} = \mathbf{y}_{ssss} + \kappa^2 \mathbf{y}_{ss} \,. \tag{52}$$

In particular, we compute κ^2 explicitly using the curvature on the interface Σ^n at $t = t^n$, i.e. $(\kappa^n)^2$, and treat $\mathbf{y}_{ssss}$ and $\mathbf{y}_{ss}$ implicitly as explained in Section 3.3.2, where s is the old metric defined on the interface at $t = t^n$. This gives

$$\begin{aligned} \mathbf{y}^{n+1} - \mathbf{y}^n &= \Delta t \left[\mathbf{y}^{n+1}_{ssss} + (\kappa^n)^2 \mathbf{y}^{n+1}_{ss} \right] \\ \mathbf{y}^{n+1} &= \left[I - \Delta t \frac{\partial^4}{\partial s^4} - \Delta t (\kappa^n)^2 \frac{\partial^2}{\partial s^2} \right]^{-1} \mathbf{y}^n \,. \end{aligned} \tag{53}$$

4. Application to flows with local inextensibility

In this section, we will consider modeling only a two dimensional flow with a local inextensibility constraint given by [18, 30]

$$(v_t)_s - \kappa v_n = 0 \,, \tag{54}$$

where s is the arclength parametrization, κ is the signed curvature and v_n and v_t are the normal and the tangential velocities, respectively. It is possible to generalize this approach to three dimensions. For instance, preserving the surface area, we have

$$\mathrm{div}_s \mathbf{u} = 0 \,, \tag{55}$$

where $\mathrm{div}_s = \mathrm{tr}\nabla_s$ is the surface divergence. This generalizes the local inextensibility condition from two dimensions (54) to

$$\mathrm{div}_s \mathbf{v}_t - H v_n = 0 \,, \tag{56}$$

where $\mathbf{v}_t$ is the tangential velocity and H is the total curvature [29].

To impose this constraint in the flow induced by the original energy $E^o(\Sigma)$, we introduce a locally defined Lagrange multiplier Λ so that the new energy $E^n(\Sigma)$ we are minimizing is given by (e.g., [35])

$$E^n(\Sigma) = E^o(\Sigma) + \int_\Sigma \Lambda \cdot (s_\Sigma - s_0) d\Sigma \,, \tag{57}$$

where s_0 and s_Σ are the arclength parametrization of the initial interface Σ_0 and the current interface Σ, respectively. Let v_n^o and v_t^o be the normal and the tangential velocities which decrease the original energy $E^o(\Sigma)$ using gradient descent. The velocity that decreases $E^n(\Sigma)$ is therefore given by

$$\mathbf{u} = (v_n^o + \Lambda\kappa)\mathbf{n} + (v_t^o + \Lambda_s)\mathbf{t} \,. \tag{58}$$

Thus Λ plays the role of an interfacial tension. The function Λ defined on the interface Σ is now determined by requiring that $\mathbf{u}$ satisfies the local inextensibility condition

$$(\mathbf{u} \cdot \mathbf{t})_s - \kappa(\mathbf{u} \cdot \mathbf{n}) = 0 \,. \tag{59}$$

This yields the following Helmholtz equation for Λ

$$-\Lambda_{ss} + \kappa^2 \Lambda = (v_t^o)_s - \kappa v_n^o \,. \tag{60}$$

After we solve this equation, we update the interface according to equation (58).

It can be shown that this modified flow (58) still minimizes the original energy $E^o(\Sigma)$. For instance, we have

$$\begin{aligned} \frac{dE^o}{dt} &= \int_\Sigma -(v_n^o \mathbf{n} + v_t^o \mathbf{t}) \cdot \mathbf{u} d\Sigma \\ &= -\int_\Sigma \left[(v_n^o)^2 + (v_t^o)^2 \right] d\Sigma - \int_\Sigma \left(v_n^o \Lambda\kappa + v_t^o \Lambda_s \right) d\Sigma \\ &= -\int_\Sigma \left[(v_n^o)^2 + (v_t^o)^2 \right] d\Sigma + \int_\Sigma \left(\Lambda^2 \kappa^2 + \Lambda_s^2 \right) d\Sigma \,. \end{aligned} \tag{61}$$

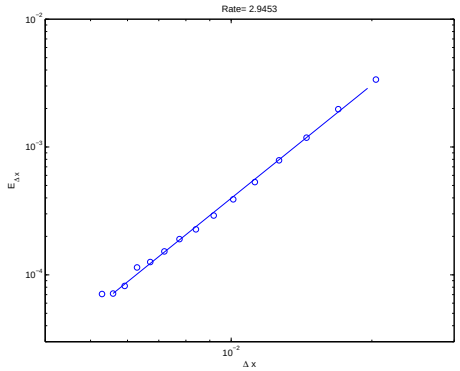

Figure 4: Convergence of solving advection equation on a rotating circle.

Now, since

$$\begin{aligned}\int_\Sigma \left(\Lambda^2\kappa^2 + \Lambda_s^2\right) d\Sigma &= -\int_\Sigma \left(v_n^o \Lambda\kappa + v_t^o \Lambda_s\right) d\Sigma \\ &\le \left[\int_\Sigma \left[(v_n^o)^2 + (v_t^o)^2\right] d\Sigma\right]^{1/2} \left[\int_\Sigma \left(\Lambda^2\kappa^2 + \Lambda_s^2\right) d\Sigma\right]^{1/2}, \end{aligned} \tag{62}$$

we obtain

$$\int_\Sigma \left(\Lambda^2\kappa^2 + \Lambda_s^2\right) d\Sigma \le \int_\Sigma \left[(v_n^o)^2 + (v_t^o)^2\right] d\Sigma\,. \tag{63}$$

And this implies that

$$\frac{dE^o}{dt} \le 0\,, \tag{64}$$

and the equality holds if $v_n^o = -\Lambda\kappa$ and $v_t^o = -\Lambda_s$.

In the case when the gradient descent of the original energy $E^o(\Sigma)$ satisfies the local inextensibility condition, we have

$$(v_t^o)_s - \kappa v_n^o = 0\,, \tag{65}$$

and so

$$-\Lambda_{ss} + \kappa^2\Lambda = 0\,. \tag{66}$$

This equation has a trivial solution $\Lambda = 0$. In particular, if the steady state solution is a circle, this trivial solution to Λ is a solution to (60). Indeed, the solution is not unique since one can have the circle rotating in a constant but arbitrary speed such that $(v_t^0)_s = 0$. In this paper however, we will not study the existence and uniqueness to the equation (60) for general interfaces.

Unlike those equations we obtained in developing implicit schemes for the advection-diffusion equation, this Helmholtz equation is inhomogeneous and, even more challenging, the coefficient in the Helmholtz operator, κ^2, has a very large range for general shapes of the interface. The simple Gauss-Seidel type iteration has difficulty in converging to a steady state solution. In the current implementation, we solve equation (60) using the SVD.

5. Example

5.1. Advection equation on a circle

The first example is a two-dimensional case where a circle centered at $(0.5, 0.75)$ with radius 0.15 is rotated by the velocity

$$\begin{aligned} u &= 2(0.5 - y) \\ v &= 2(x - 0.5)\,. \end{aligned} \tag{67}$$

The circle rotates about $(0.5, 0.5)$ with period $T = \pi$. On the circle, we solve the advection equation

$$\frac{Df}{Dt} = 0\,, \tag{68}$$

where

$$\frac{D}{Dt} = \frac{\partial}{\partial t} + u\frac{\partial}{\partial x} + v\frac{\partial}{\partial y} \tag{69}$$

is the material derivative along the trajectory of a particle on the interface. The initial condition defined on the circle is given by

$$f(x, y, 0) = x\,. \tag{70}$$

The exact solution to this interface problem is given by

$$f(x(T), y(T), T) = x_0\,, \tag{71}$$

where $x(T) = x(0) = x_0$ and $y(T) = y(0)$. Figure 4 shows the rate of convergence of our method applied to solve this simple advection equation on an evolving surface, which is approximately third order (2.9453).

5.2. Surface diffusion on a sphere

We next simulate the diffusion of a material along a moving surface. We first repeat a similar two-dimensional example as in the previous test where we again consider the rotation of a circle as defined above. However, to match the scaling of the diffusion time, we modify the rigid body rotation by

$$\begin{aligned} u &= 200(0.5 - y) \\ v &= 200(x - 0.5)\,. \end{aligned} \tag{72}$$

The circle now rotates about $(0.5, 0.5)$ with period $T = \pi/100$. On the moving circle, we solve the following advection-diffusion equation

$$\frac{Df}{Dt} = \Delta_S f\,, \tag{73}$$

where D/Dt is the material derivative along the trajectory of a particle on the interface. The initial condition defined on the circle is given by

$$f(\theta, 0) = \sin(n\theta)\,, \tag{74}$$

where $n = 2$, $\theta = \tan^{-1}[(y - 0.75)/(x - 0.5)]$. The exact solution to this problem is given by

$$f(X(t), Y(t), t) = \exp\left(-\frac{n^2 t}{r^2}\right) f(X(0), Y(0), 0)\,, \tag{75}$$

where r is the radius of the circle, and $(X(t), Y(t))$ is the trajectory satisfying

$$\begin{aligned} \frac{dX(t)}{dt} &= u \\ \frac{dY(t)}{dt} &= v\,. \end{aligned} \tag{76}$$

The maximum value of $f(\theta, t)$ on the interface is then given by

$$\max_{\theta} f(\theta, t) = \exp\left(-\frac{n^2 t}{r^2}\right)\,. \tag{77}$$

In figure 5, we plot the computed maximum value of $f(\theta, t)$ on the circle at various times using the explicit scheme with an underlining grid of resolution 257×257 (a), and also their corresponding errors (b). We

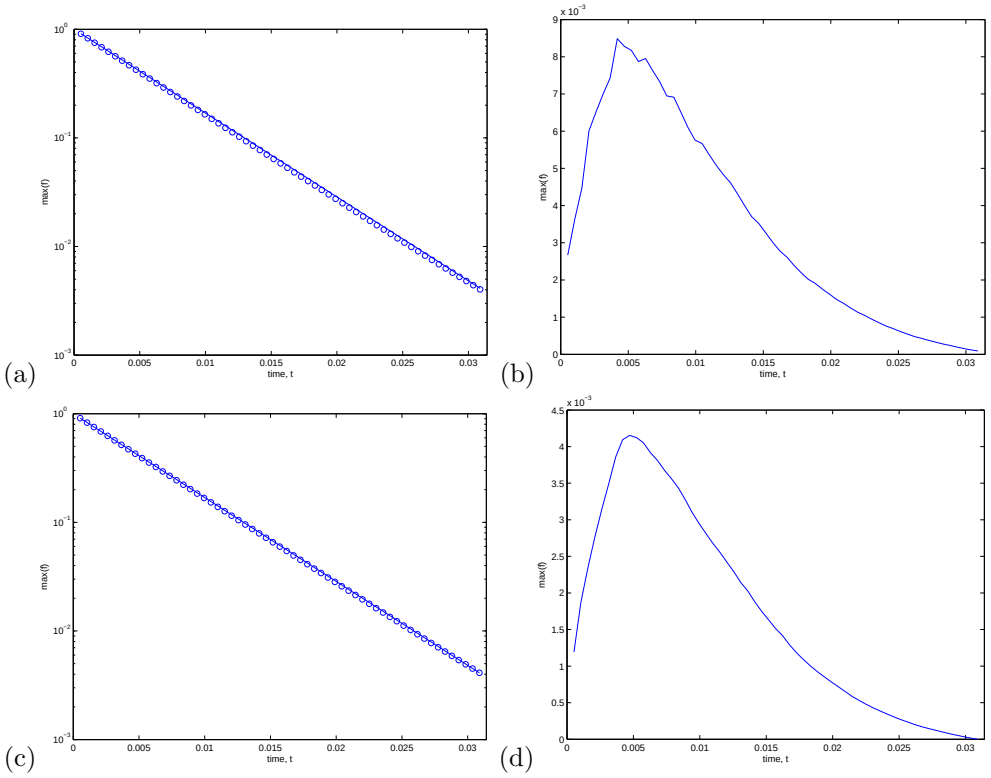

Figure 5: Maximum value of $f(\theta, t)$ defined on the circle under the rigid body rotation and its error using the explicit scheme with the underlying uniform mesh of resolution 257^2 (first row) and 513^2 (second row). The circles on the left subfields are the computed solution. The solid line is the exact solution.

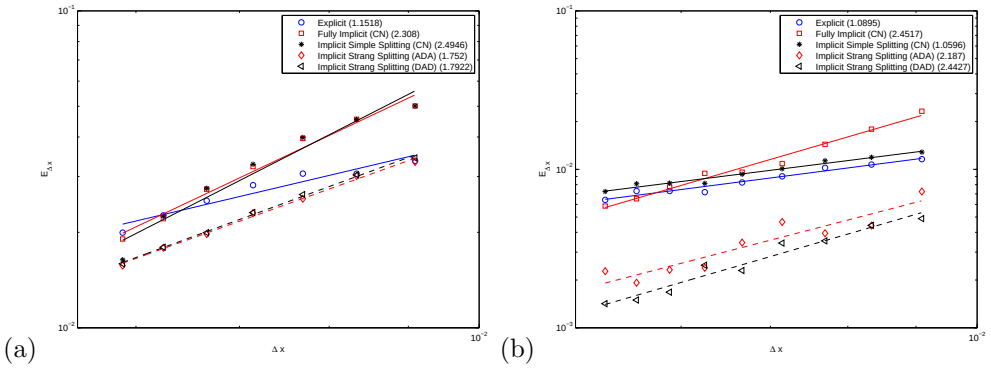

Figure 6: Convergence of various numerical schemes for solving the heat equation on an evolving surface with motion governed by (a) the simple rigid body rotation and (b) the outward normal expansion.

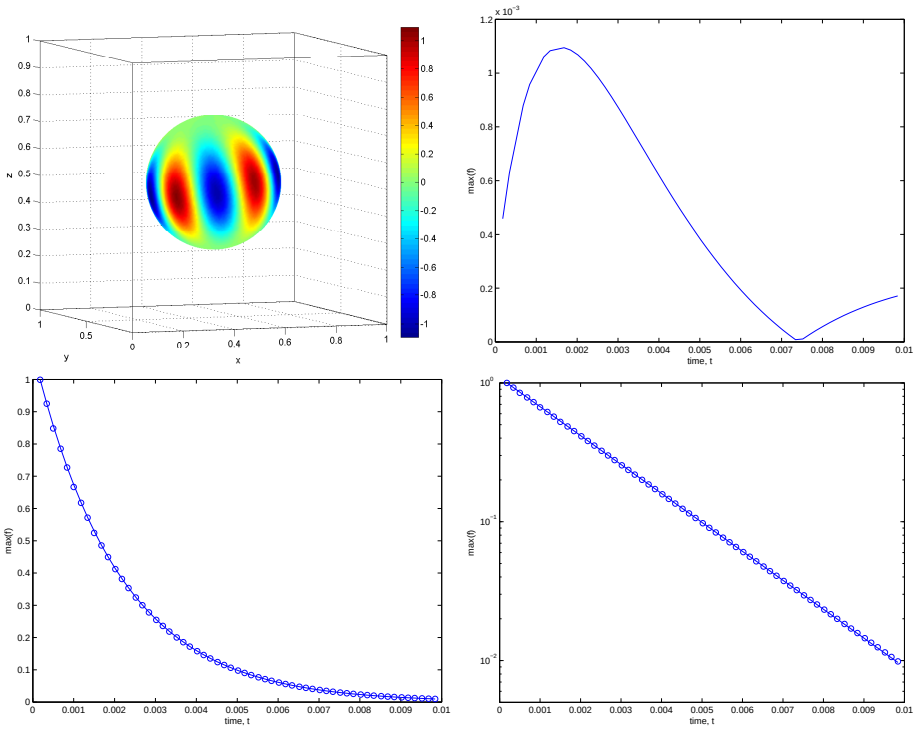

Figure 7: Maximum value of $f(\theta, \phi, t)$ defined on the sphere and its error using the underlying uniform mesh of resolution 101^3. The circles on the second row are the computed solution. The solid line is the exact solution.

also plot the corresponding quantities using an underlining grid of resolution 513×513 in figure 5 on the second row.

In figure 6, we demonstrate the convergence of various numerical schemes for solving the heat equation on an evolving surface. In figure 6 (a), we show the convergence results of the rigid body rotation demonstrated above. In figure 6 (b), we solve the surface heat equation on an initial circle of radius $r_0 = 0.1$ centered at $(0.5, 0.5)$ expanding in the normal direction with speed 200, i.e. $v_n = 200$, for time $t_f = \pi/4000$. The initial condition of a function f defined on the interface is given by $f(\theta; t = 0) = \sin(n\theta)$ with $n = 2$. The exact solution on the circle at the final time can be found explicitly,

$$f(\theta; t = t_f) = \exp\left[-\frac{n^2 t_f}{r_0(r_0 + v_n t_f)}\right] \sin(n\theta)\,. \tag{78}$$

As expected, the explicit scheme gives approximately first order convergence while the fully implicit CN scheme (39), and the Strang splitting schemes (36) and (37) all give approximately second order convergence. An interesting observation is that the CN scheme based on simple splitting exhibits a different convergence behavior for these two types of motion. For the rigid rotation, the underlying metric remains unchanged, i.e. the surface Laplacian operator $\Delta_{\Sigma(t^n)} = \Delta_{\Sigma(t^{n+1})}$. Thus the simple splitting in fact coincides with that from the fully implicit CN scheme (39), which is why the methods (33) and (39) give similar results in figure 6 (a).

We repeat a similar test in three-dimensions but without an applied velocity. We consider a sphere of radius 0.25 centered at $(0.5, 0.5, 0.5)$ with a function defined on the surface initially given by a linear combination of two spherical harmonic functions

$$f(\theta, \phi, t = 0) = \sin^5\theta \cos(5\phi) + \sin^4\theta \cos\theta \cos(4\phi)\,. \tag{79}$$

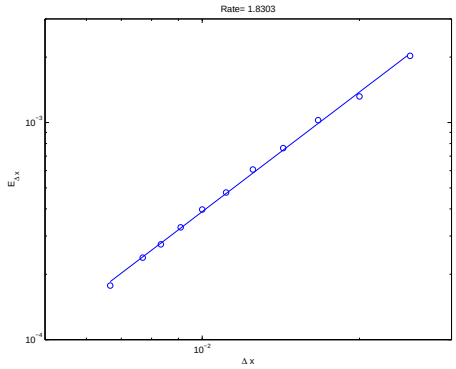

Figure 8: Rate of convergence for solving the diffusion equation on a sphere.

The exact solution to the heat equation on the surface

$$\frac{\partial f}{\partial t} = \Delta_s f \tag{80}$$

is given by

$$f(\theta, \phi, t) = \exp\left(-\frac{30t}{r_0^2}\right) f(\theta, \phi, 0)\,. \tag{81}$$

We show the maximum value of $f(\theta, \phi, t)$ on the surface at different times and error in figure 7 using a mesh of $151 \times 151 \times 151$. In figure 8, we have plotted the error defined as

$$E_{\Delta x} = \exp\left(-\frac{0.3}{r_0^2}\right) \max_{\Sigma} f(\theta, \phi, 0) - \max_{\Sigma} f(\theta, \phi, 0.01)\,, \tag{82}$$

with $\max_\Sigma f(\theta, \phi, 0) = (5 + \sqrt{41})^4/[100\sqrt{10}(7 + \sqrt{41})^{3/2}]$. The figure shows that the rate of convergence is approximately 2 (1.8303).

5.3. Motion by surface diffusion and Willmore flow

In this section, we simulate high order geometric motion due to surface diffusion and Willmore flow. Figures 9 - 11 show a simple case where an initial seven-fold symmetric star shaped interface evolves by surface diffusion. In figures 9 and 10 we plot the solutions at various times during the evolution computed by the simple explicit scheme using an underlying mesh with $\Delta x = 1/128$ and $\Delta x = 1/256$, respectively. High order geometric information, i.e. the surface Laplacian of the curvature, is computed by first treating the curvature as a function defined at the foot-points, then computing the corresponding derivatives of the local least squares reconstruction of this resulting function. Gaussian smoothing of the curvature as described in 3.4 is applied before interpolation and differentiation. For a relatively large $\Delta t = 32\Delta x^4$, the solutions show instability in the evolution. The $\Delta t = O(\Delta x^4)$ constraint makes the computation inefficient for small Δx. Figure 11 shows the solutions at the final time t_f computed using the semi-implicit scheme from Eq. (53) described in Section 3.5 using various Δt's. In the upper row, figures 11 (a-b), we use an underlying mesh with $\Delta x = 1/128$. The timestep Δt is chosen to be $128\Delta x^4$ and $512\Delta x^4$, respectively. These timesteps are significantly larger than the timestep restriction in the explicit scheme. For the case where $\Delta t \simeq 512\Delta x^4$, the solution is in fact obtained by doing only four steps of time marching. This explains why the solution in figure 11 (b) is not accurate at all. Similarly in figures 11 (c-d), we have shown the solution at the time t_f using various Δt's with a finer mesh. All evolutions are stable but the accuracy in the solution decreases when we increase the time step.

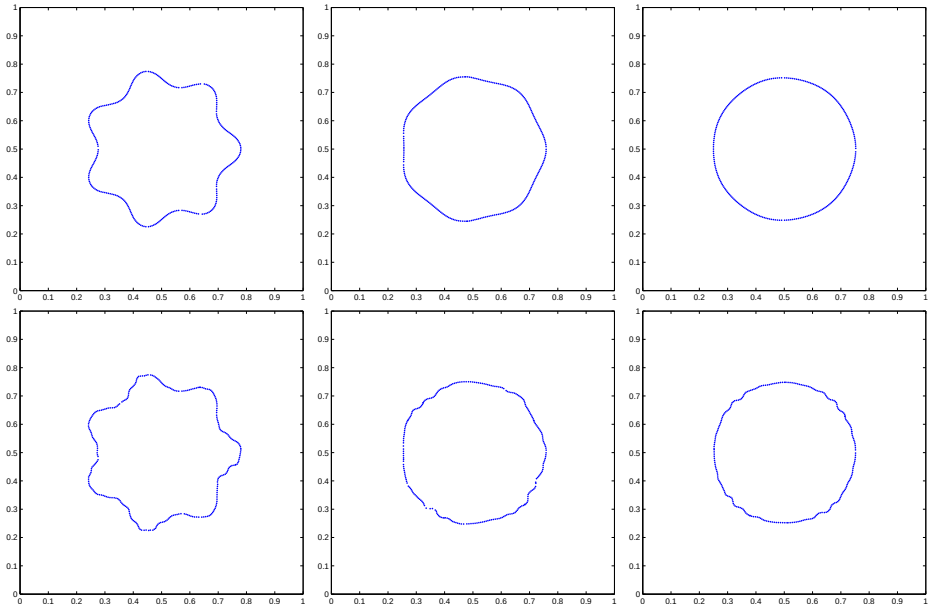

Figure 9: Evolution of an initial 7-fold symmetric star shaped interface by surface diffusion using the explicit method with $\Delta x = 1/128$ and (upper row) $\Delta t = 16\Delta x^4$ and (bottom row) $\Delta t = 32\Delta x^4$.

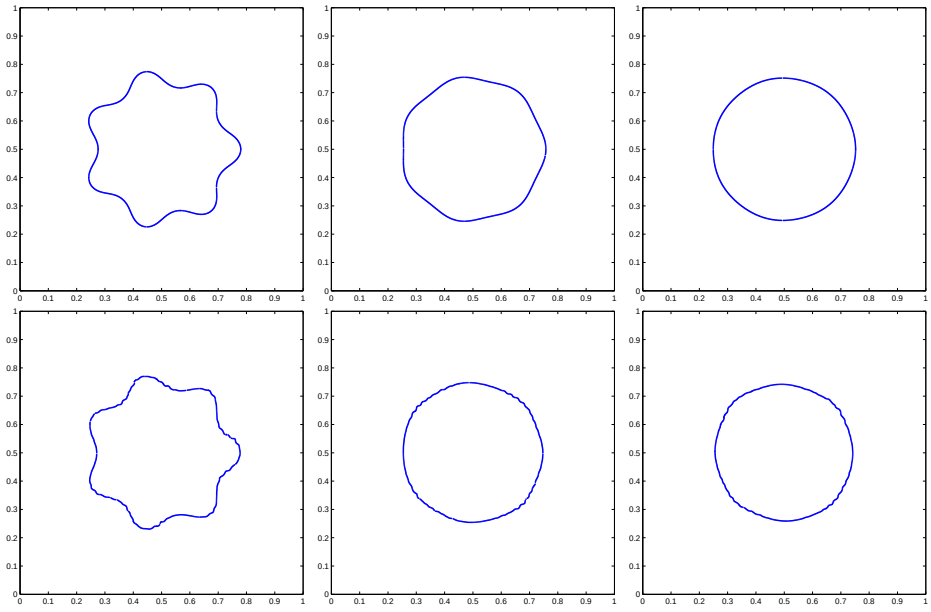

Figure 10: Evolution of an initial 7-fold symmetric star shaped interface by surface diffusion using the explicit method with $\Delta x = 1/256$ and (upper row) $\Delta t = 16\Delta x^4$ and (bottom row) $\Delta t = 32\Delta x^4$.

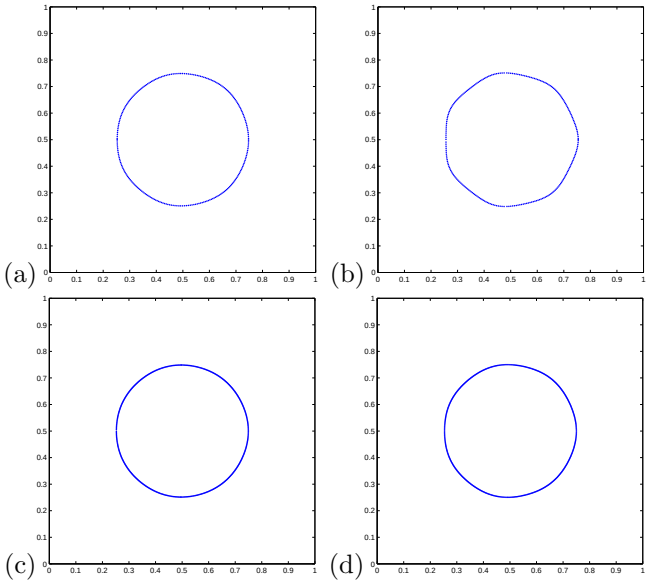

Figure 11: Solution of an initial 7-fold symmetric star shaped interface by surface diffusion at $t_f = 7.5 \times 10^{-6}$ using the semi-implicit method with (a) $\Delta x = 1/128$ and $\Delta t = 128\Delta x^4 = \Delta x^2/128 \simeq t_f/16$, (b) $\Delta x = 1/128$ and $\Delta t = 512\Delta x^4 = \Delta x^2/32 \simeq t_f/4$, (c) $\Delta x = 1/256$ and $\Delta t = 512\Delta x^4 = \Delta x^2/128 \simeq t_f/64$, and (d) $\Delta x = 1/256$ and $\Delta t = 2048\Delta x^4 = \Delta x^2/32 \simeq t_f/16$.

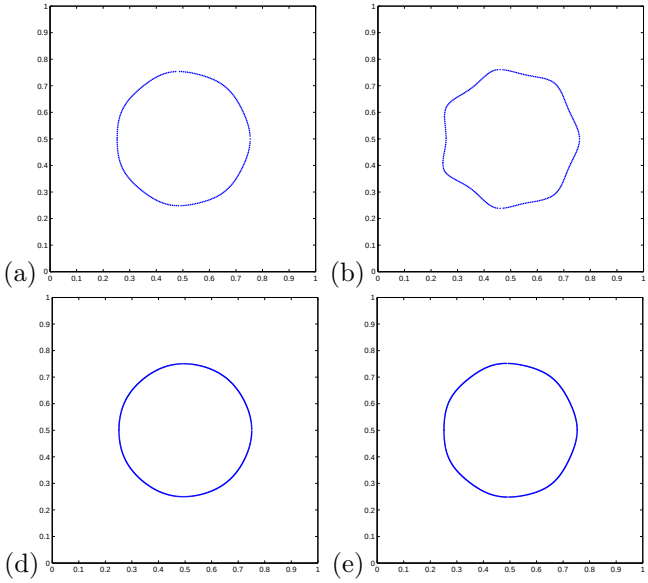

Figure 12: Solution of an initial 7-fold symmetric star shaped interface under Willmore flow at $t_f = 7.5 \times 10^{-6}$ using the semi-implicit method with (a) $\Delta x = 1/128$ and $\Delta t = 128\Delta x^4 = \Delta x^2/128 \simeq t_f/16$, (b) $\Delta x = 1/128$ and $\Delta t = 512\Delta x^4 = \Delta x^2/32 \simeq t_f/4$, (c) $\Delta x = 1/256$ and $\Delta t = 512\Delta x^4 = \Delta x^2/128 \simeq t_f/64$, and (d) $\Delta x = 1/256$ and $\Delta t = 2048\Delta x^4 = \Delta x^2/32 \simeq t_f/16$.

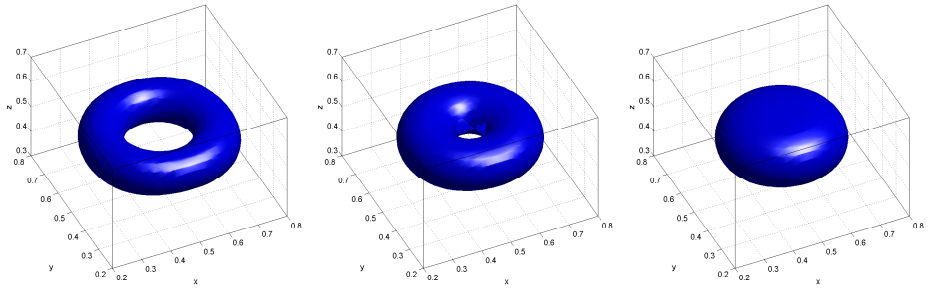

Figure 13: Motion of a torus by surface diffusion. The radius at the initial time is 0.075. The topological change is nicely captured by the GBPM.

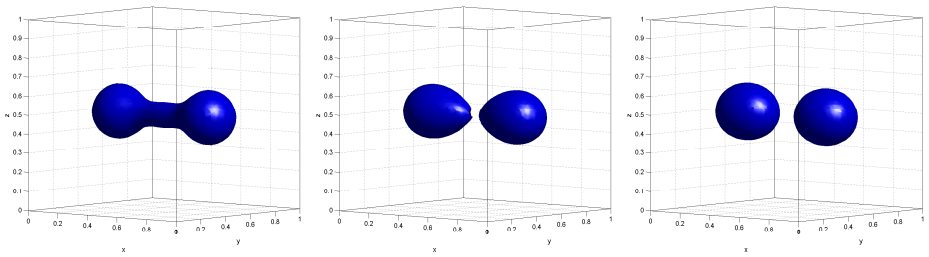

Figure 14: Motion of a dumbbell shape by surface diffusion. The radius of the middle cylindrical part at the initial time is 0.075. The topological change is nicely captured by the GBPM.

We have also shown the solution to the Willmore flow with the same initial condition in figure 12. As in the surface diffusion case, the semi-implicit scheme effectively removes the restrictive time step constraint $\Delta t = O(\Delta x^4)$.

The GBPM can be easily extended to three dimensions as described in section 3.2. To demonstrate this, in figures 13-15, the evolution of a torus and a dumbbell shaped interface under surface diffusion are shown. The topological changes in figures 13 and 14 are captured nicely using the GBPM. The idea is to deactivate any grid point and its associated foot-point whose Lagrangian information at the corresponding foot-point conflicts with any of its neighbor. Removal and addition of meshless sampling points is simple for the GBPM.

We have also performed analogous simulations for motion by the Willmore flow; the results are shown in figures 16 - 18. As observed previously (e.g., [26]) the dumbbell does not pinch off under Willmore flow

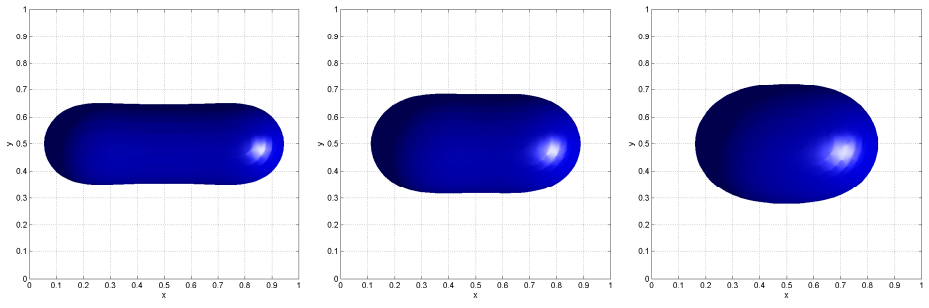

Figure 15: Motion of a dumbbell shape by surface diffusion up to $t = 10^{-4}$. The radius of the middle cylindrical part at the initial time is 0.150. The topological change is nicely captured by the GBPM.

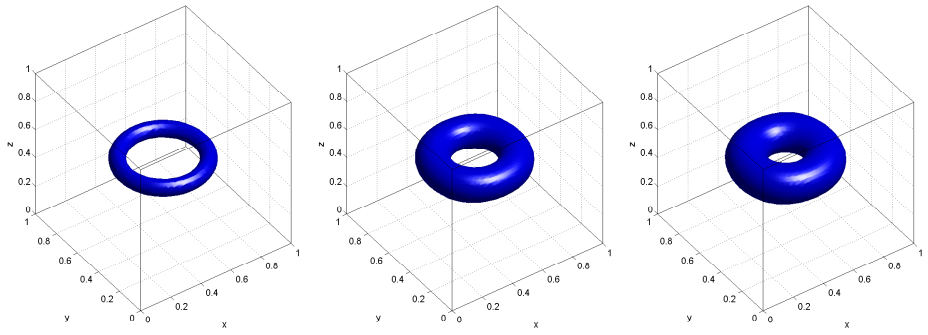

Figure 16: Motion of a torus by Willmore flow up to $t = 3 \times 10^{-4}$. The radius at the initial time is 0.05.

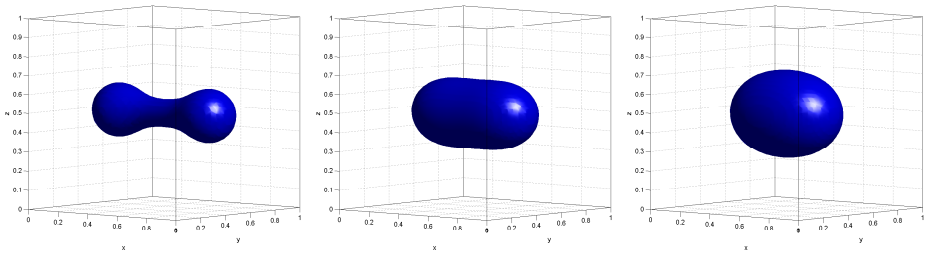

Figure 17: Motion of a dumbbell shape by Willmore flow up to $t = 3 \times 10^{-4}$. The radius of the middle cylindrical part at the initial time is 0.075.

but rather evolves to a sphere.

5.4. Local inextensible flows

Consider first a case in which a circle of radius r_0 is evolved by a flow in the normal direction with $v_n^0 = 1$ and $v_t^0 = 0$. This velocity field does not satisfy the inextensibility constraint. In particular, the arclength grows linearly with time given by $2\pi(r_0 + t)$. Imposing the local inextensible constraint given by equation (60), we have

$$-\Lambda_{\theta\theta} + \Lambda = -r\,. \tag{83}$$

Assuming the solution Λ is independent of θ, we have $\Lambda = -r$ and therefore equation (58) gives $\mathbf{u} = \mathbf{0}$ which means that the initial circle should stay circle with the same radius. In figure 19 (a) we have shown the change in the total arclength of a circle with an initial radius $r_0 = 0.1$. The blue circles are the computed

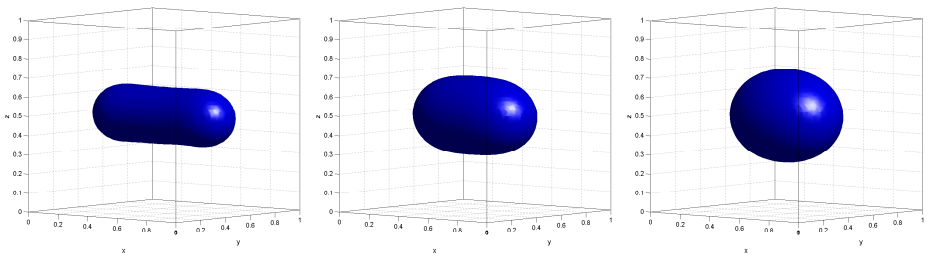

Figure 18: Motion of a dumbbell shape by Willmore flow up to $t = 3 \times 10^{-4}$. The radius of the middle cylindrical part at the initial time is 0.150.

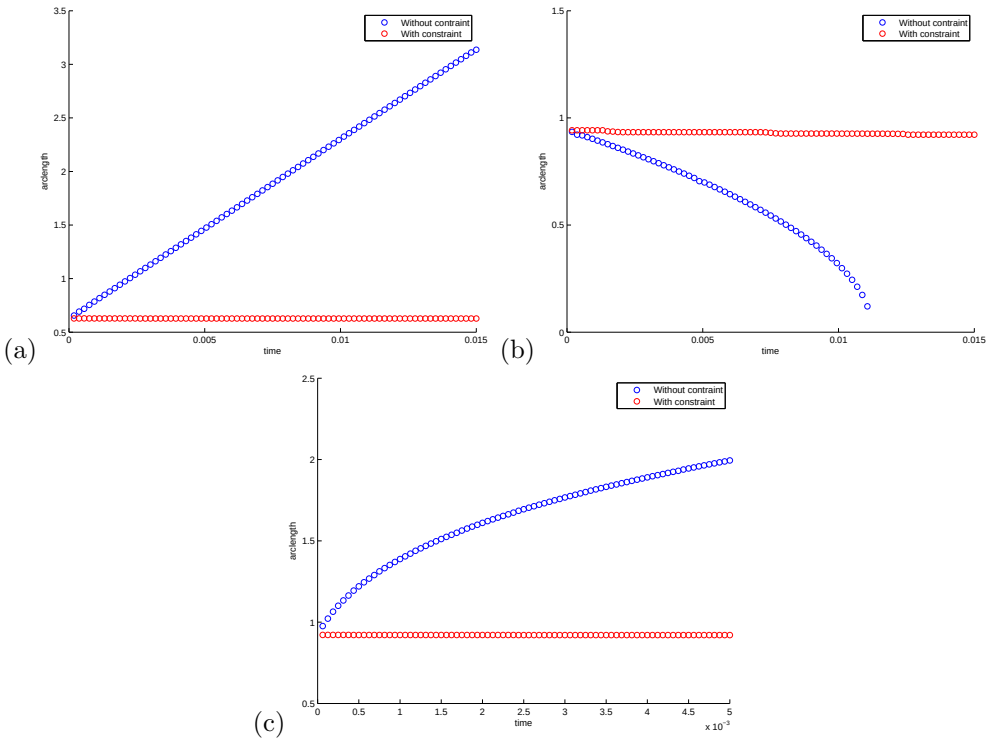

Figure 19: Evolution of arclength for several geometric flows and their inextensible analogue. (a) Motion in the outward normal direction, (b) motion by mean curvature, and (c) the Willmore flow of an initial circle of radius $r_0 = 0.15$ with/without the local inextensibility constraint.

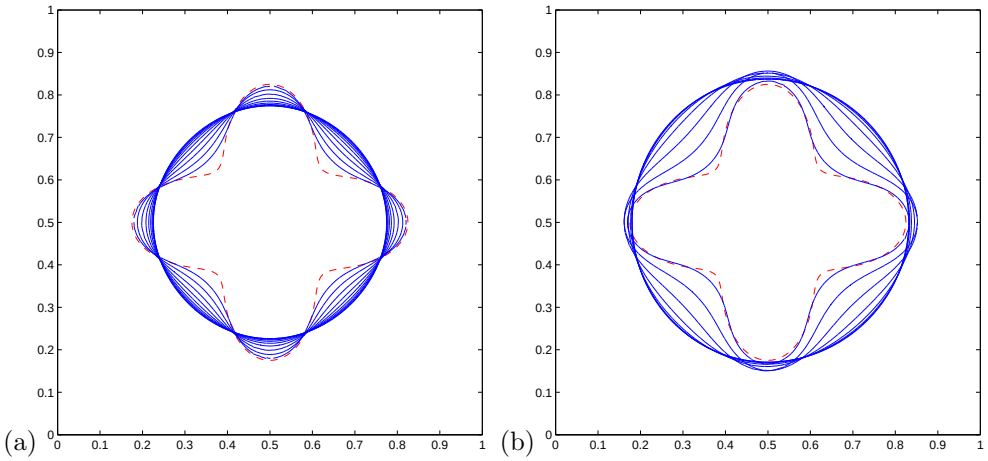

Figure 20: Evolution of an initial 4-fold symmetric interface with $r_0 = 0.25$ and $\epsilon = 0.30$ under the Willmore flow (a) without and (b) with the local inextensible constraint. The red dashed line is the initial condition. The interface at various times are plotted in blue solid lines.

total arclength under the flow $v_n^0 = 1$ without the local inextensibility constraint. The computed total length of the interface when the local inextensible constraint is applied is shown in red.

A slightly more complicated case is the motion by mean curvature. We start with an initial circle of radius $r_0 = 0.15$, i.e. the curvature $\kappa = 1/r_0$. The total arclength of the circle is given by $r(t) = \sqrt{0.15^2 - 2t}$ for $0 \le t \le 0.15^2/2$. In figure 19 (b) we show the computed arclength of the evolution of the circle with and without the local inextensible constraint. Similar to the previous case, the circle stays unchanged with inextensibility.

The next example is the Willmore flow (46). In two dimensions, the corresponding normal velocity reduces to

$$v_n = \Delta_\Sigma \kappa - \frac{1}{2}\kappa^3 . \tag{84}$$

And, therefore, an initially circular interface grows in the outward direction with radius given by $r(t) = (2t + r_0^4)^{1/4}$. In figure 19 (c), we show the computed arclength of an initial circle of radius $r_0 = 0.15$ under Willmore flow with and without the local inextensible constraint. The results are similar to the previous cases.

We next consider a more extreme case for Willmore flow with an initial 4-fold symmetric interface profile

$$r = r_0 \cdot [1 + \epsilon \cos(4\theta)] \tag{85}$$

where $r_0 = 0.25$ and $\epsilon = 0.30$. In figure 20, we show the evolution of the initial interface under Willmore flow with (a) and without (b) the local inextensible constraint. In both of these figures, we plot the initial condition as the red dashed line. The interface at different times is plotted using blue solid lines. Even though both flows (the constrained and the unconstrained flows) tend to a circle, the radii of the circles are different since the constrained flow preserves the interface length.

6. Conclusion and future work

In this work, we have developed numerical algorithms for solving advection-diffusion PDEs on moving interfaces using the grid based particle method (GBPM). We also demonstrated the ability of the GBPM to simulate high order geometric flows including changes in the interface topology. To solve the equations efficiently, we developed semi-implicit methods to overcome the severe time step constraints required for the stability of explicit methods.

In the future, we plan to extend the algorithms to incorporate global constraints such as the conservation of enclosed volume. Because of the local representation of the interface in the GBPM this is a challenging task. In addition, we plan to extend the GBPM to solve problems in which there is coupling between bulk and surface processes.

Acknowledgements

SL acknowledges the hospitality of the University of California, Irvine where preliminary work was performed. SL gratefully acknowledges partial support from the Hong Kong RGC under Grant GRF602210. JL acknowledges partial support from the National Science Foundation (NSF), Division of Mathematical Sciences (DMS). HZ was partially supported by NSF grant DMS0811254.

References

[1] D. Adalsteinsson and J.A. Sethian. Transport and diffusion of materials quantities on propagating interfaces via level set methods. *J. Comput. Phys.*, 185:271–288, 2003.

[2] E. Bansch, P. Morin, and R.H. Nochetto. A finite element method for surface diffusion: The parametric case. *J. Comput. Phys.*, 203:321–343, 2005.

[3] J.W. Barrett, H. Garcke, and R. Nurnberg. A parametric finite element method for fourth order geometric evolution equations. *J. Comput. Phys.*, 222:441–467, 2003.

[4] M. Bertalmio, L.-T. Cheng, S. Osher, and G. Sapiro. Variational problems and partial differential equations on implicit surfaces. *J. Comput. Phys.*, 174:759–780, 2001.

[5] M. Burger. Numerical simulation of anistropic surface diffusion with curvature-dependent energy. *J. Comput. Phys.*, 203:602–625, 2005.

[6] G. Chirikjian and J. Burdick. A model approach to hyper-redundant manipulator kinematics. *IEEE Trans. Robot. Autom.*, 10:343–354, 1994.

[7] F. Davi and M.E. Gurtin. On the motion of a phase interface by surface diffusion. *Journal of Applied Mathematics and Physics*, 41:782–811, 1990.

[8] Q. Du, C. Liu, and X. Wang. A phase field approach inthe numerical study of elastic bending energy for vesicle membranes. *J. Comput. Phys.*, 198:450–468, 2004.

[9] G. Dziuk and C.M. Elliott. Finite elements on evolving surfaces. *IMAJ Num. Analy.*, 27:262–292, 2007.

[10] G. Dziuk and C.M. Elliott. Surface finite elements for parabolic equations. *J. Computational Mathematics*, 25:385–407, 2007.

[11] G. Dziuk and C.M. Elliott. An Eulerian approach to transport and diffusion on evolving implicit surfacs. *Computing and Visualization in Science*, 13:17–28, 2008.

[12] C.M. Elliott and B. Stinner. Analysis of a diffusion interface approach to an advection diffusion equation on a moving surface. *Math. Mod. Meth. Appli. Sci.*, 19:787–802, 2009.

[13] J. Escher, U.F. Mayer, and G. Simonett. The surface diffusion flow for immersed hypersurfaces. *SIAM J. Math. Anal.*, 29:1419–1433, 1998.

[14] R. Freund and N. Nachitigal. QMR: A quasi-minimal residual method for non-hermitian linear systems. *Numer. Math.*, 60:315–339, 1991.

[15] J. Greer, A.L. Bertozzi, and G. Sapiro. Fourth order partial differential equations on general geometries. *J. Comput. Phys.*, 216:216–246, 2006.

[16] W. Helfrich. Elastic properties of lipid bilayers-theory and possible experiments. *Zeitschrift für Naturforschung C*, 28:693–703, 1973.

[17] A.J. James and J. Lowengrub. A surfactant-conserving volume-of-fluid method for interfacial flows with insoluble surfactant. 201:685–722, 2004.

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

[19] 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.

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

[21] 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.

[22] B. Li, J. Lowengrub, A. Ratz, and A. Voigt. Geometric evolution laws for thin crystalline films: Modeling and numerics. *J. Comm. Comp. Phys.*, 6:433–482, 2009.

[23] C.B. Macdonald and S.J. Ruuth. The implicit closest point method for the numerical solution of partial differential equations on surfaces. *SIAM J. Sci. Comput.*, 31:4330–4350, 2009.

[24] J. Macklin, P.and Lowengrub. Evolving interfaces via gradients of geometry-dependent interior poisson problems: Application to tumor growth. *J. Comput. Phys.*, 203:191–220, 2005.

[25] U.F. Mayer. Numerical solutions for the surface diffusion flow in three space dimensions. *Computational and Applied Mathematics*, 20:361–379, 2001.
[26] U.F. Mayer and G. Simonett. A numerical scheme for axisymmetric solutions of curvature driven free boundary problems, with applications to the willmore flow. *Interfaces and Free Boundaries*, 4:89–109, 2002.
[27] W.W. Mullins. Theory of thermal grooving. *Journal of Applied Physics*, 28:333–339, 1957.
[28] A. Ratz and A. Voigt. Pdes on surfaces- a diffuse interface approach. *Commun. Math. Sci.*, 4:575–590, 2006.
[29] R. Rosso. Curvature effects in vesicle-particle interactions. *Proc. R. Soc. Lond. A*, 459:829–852, 2003.
[30] R. Rosso, A.M. Sonnet, and E.G. Virga. Evolution of vesicles subject to adhesion. *Proc. R. Soc. Lond. A*, 456:1523–1545, 2000.
[31] S.J. Ruuth and B. Merriman. A simple embedding method for solving partial differential equations on surfaces. *J. Comput. Phys.*, 227:1943–1961, 2008.
[32] I.F. Sbalzarini, A. Hayer, A. Helenius, and P. Koumoutsakos. Simulations of (an)isotropic diffusion on curved biological surfaces. *Biophys. J.*, 90:878–885, 2006.
[33] U. Siefert. Configurations of fluid membranes and vesicles. *Adv. Phys.*, 46:13–137, 1997.
[34] G. Simonett. The willmore flow near spheres. *Differential and Integral Equations*, 14:1005–1014, 2001.
[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] P. Song, D. Hu, and P. Zhang. Numerical simulation of fluid membranes in two-dimensional space. *Commun. Comput. Phys.*, 3:794–821, 2008.
[37] K.E. Teigen, X. Li, J. Lowengrub, F. Wang, and A. Voigt. A diffuse-interface approach for modeling transport, diffusion and adsorption/desorption of material quantities on a deformable interface. *Commun. Math. Sci.*, 7:1009–1037, 2009.
[38] A.-K. Tornberg and M.J. Shelley. Simulating the dynamics and interactions of flexible fibers in stokes flows. *J. Comput. Phys.*, 196:8–40, 2004.
[39] S.K. Veerapaneni, D. Gueyffier, G. Biros, and D. Zorin. A numerical method for simulating the dynamics of 3d axisymmetric vesicles suspended in viscous flows. *J. Comput. Phys.*, 228:7233–7249, 2009.
[40] 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.
[41] K. Wang and H. Begleiter. Local polynomial estimate of surface laplacian. *Brain Topography*, 12:19–29, 1999.
[42] T.J. Willmore. *Riemannian Geometry*. New York: Oxford Science Publications, The Clarendon Press, Oxford University Press, 1993.
[43] J. Xu, Z. Li, J. Lowengrub, and H. Zhao. A level set method for intefacial flows with surfactant. 212:590–616, 2006.
[44] J. Xu and H.K. Zhao. An eulerian formulation for solving partial differential equations along a moving interface. *J. Sci. Comput.*, 19:573–594, 2003.