When surface evolution meets Fokker-Planck equation: a novel tangential velocity model for uniform parametrization

Jiangong Pan[a], Guozhi Dong[b], Hailong Guo[c], Zuoqiang Shi[d,e,*]

[a]*Department of Mathematical Sciences, Tsinghua University, Beijing, 100084, China*
[b]*School of Mathematics and Statistics, HNP-LAMA, Central South University, Changsha, 410083, China*
[c]*School of Mathematics and Statistics, The University of Melbourne, Parkville, VIC 3010, Australia*
[d]*Yau Mathematical Sciences Center, Tsinghua University, Beijing, 100084, China*
[e]*YanqiLake Beijing Institute of Mathematical Sciences and Applications, Beijing, 101408, China*

Abstract

A common issue in simulating geometric evolution of surfaces is unexpected clustering of points that may cause numerical instability. We propose a novel artificial tangential velocity method for this matter. The artificial tangential velocity is generated from a surface density field governed by a Fokker–Planck equation to guide the point distribution. A target distribution matching algorithm is developed leveraging the surface Kullback-Leibler divergence of density functions. The numerical method is formulated within a fully meshless framework using the moving least squares approximation, thereby eliminating the need for mesh generation and allowing flexible treatment of unstructured point cloud data. Extensive numerical experiments are conducted to demonstrate the robustness, accuracy, and effectiveness of the proposed approach across a variety of surface evolution problems, including the mean curvature flow.

Keywords: evolving surface, point clouds, artificial tangential velocity, Fokker–Planck equation, target distribution algorithm, mean curvature flow
2000 MSC: 35R01, 53E10, 65M75, 35K10

1. Introduction

We investigate the evolution of a closed surface $\Gamma(t) \subset \mathbb{R}^3$ governed by a prescribed velocity field $\boldsymbol{v} \in \mathbb{R}^3 \times [0, T]$. The surface evolution is described by the following system:

$$\begin{aligned} \frac{d}{dt}\boldsymbol{X}(\boldsymbol{x}, t) &= \boldsymbol{v}(\boldsymbol{X}(\boldsymbol{x}, t), t), \quad \boldsymbol{x} \in \Gamma_0, \quad t \geq 0, \\ \boldsymbol{X}(\boldsymbol{x}, 0) &= \boldsymbol{x}, \qquad\qquad\quad \boldsymbol{x} \in \Gamma_0, \end{aligned} \tag{1}$$

*Corresponding author.
Email addresses: mathpjg@sina.com (Jiangong Pan), guozhi.dong@csu.edu.cn (Guozhi Dong), hailong.guo@unimelb.edu.au (Hailong Guo), zqshi@tsinghua.edu.cn (Zuoqiang Shi)

where Γ_0 denotes a closed surface embedded in $\mathbb{R}^3$, and $\Gamma(t)$ is the image of the flow map $\boldsymbol{X}(\cdot, t)$ at time t. One of the major challenges in geometric surface flow lies in the degradation of mesh quality over time, as node clustering and mesh distortion may occur, eventually leading to numerical breakdown. This issue affects both mesh-based and mesh-free approaches. To address this, Bänsch et al. introduced a novel mesh reconstruction technique that reinitializes the mesh when excessive node aggregation or deformation is detected [1]. In mesh-based methods, another effective strategy involves leveraging parametric geometric flows guided by harmonic maps of a reference surface with well-distributed mesh points. Related methodologies can also be found in [2, 3].

Among mesh-based approaches, a seminal contribution was made by Dziuk in 1990, who pioneered the numerical treatment of surface evolution under geometric flows [4]. He introduced the (parametric) finite element method (FEM) for approximating the mean curvature flow (MCF) of closed surfaces in three-dimensional space. Given an approximate surface $\Gamma_h(t_j) \subset \mathbb{R}^3$ represented via triangular faces of a polyhedron, the parametric FEM computes a parametrization $u_h^{j+1} : \Gamma_h(t_j) \to \mathbb{R}^3$ of the evolved surface mesh $\Gamma_h(t_{j+1}) = u_h^{j+1}(\Gamma_h(t_j))$ by solving the following weak formulation on the known surface $\Gamma_h(t_j)$: Find u_h^{j+1} in the three-dimensional vector-valued finite element space $S_h(\Gamma_h(t_j))^3$ such that

$$\int_{\Gamma_h(t_j)} \frac{u_h^{j+1} - \mathrm{id}}{\tau} \cdot \chi_h + \int_{\Gamma_h(t_j)} \nabla_{\Gamma_h(t_j)} u_h^{j+1} \cdot \nabla_{\Gamma_h(t_j)} \chi_h = 0, \quad \forall \chi_h \in S_h\left(\Gamma_h(t_j)\right)^3.$$

The discrete flow map $X_h^{j+1} : \Gamma_h(0) \to \Gamma_h(t_{j+1})$ is then updated via composition: $X_h^{j+1} = u_h^{j+1} \circ X_h^j$. Since its inception, this method has been widely adopted for simulating surface evolution under various geometric flows, such as mean curvature flow and Willmore flow [1, 5, 6]. The development of numerical methods for solving partial differential equations on evolving surfaces continues to receive significant attention [7, 8, 9, 10, 11, 12, 13, 14]

In a series work [15, 16, 17], Barrett, Garcke, and Nürnberg introduced a novel variational formulation for the normal component of the velocity equation, permitting tangential motion of the approximated surface. This approach, known as the BGN method, implicitly defines the tangential velocity by enforcing that the mapping from $\Gamma_h(t_j)$ to $\Gamma_h(t_{j+1})$ is a discrete harmonic map. This formulation significantly enhances mesh quality and numerical robustness without resorting to explicitly redistribute the mesh. The BGN scheme for MCF is stated as follows: find $u_h^{j+1} \in S_h(\Gamma_h(t_j))^3$ and $H_h^{j+1} \in S_h(\Gamma_h(t_j))$ satisfying the weak formulation:

$$\left(\frac{u_h^{j+1} - \mathrm{id}}{\tau} \cdot \hat{\boldsymbol{n}}_h^j, \xi_h\right) \Gamma_h(t_j)^h + \left(H_h^{j+1}, \xi_h\right) \Gamma_h(t_j)^h = 0, \quad \forall \xi_h \in S\left(\Gamma_h(t_j)\right),$$

$$\left(H_h^{j+1} \hat{\boldsymbol{n}}_h^j, \chi_h\right) \Gamma_h(t_j)^h - \int \Gamma_h(t_j) \nabla_{\Gamma_h(t_j)} u_h^{j+1} \cdot \nabla_{\Gamma_h(t_j)} \chi_h = 0, \quad \forall \chi_h \in S\left(\Gamma_h(t_j)\right)^3,$$

where the superscript h denotes a mass-lumped inner product on the discrete surface $\Gamma_h(t_j)$, and $\hat{\boldsymbol{n}}_h^j$ is a vertex-wise averaged unit normal vector on $\Gamma_h(t_j)$. Since then, the BGN method has been extended and adapted to various applications. For example, Bao et al. [18, 19, 20] developed parametric FEM schemes for surface diffusion incorporating artificial tangential

velocities, energy stability, and volume preservation. These methods have been employed to simulate interface dynamics in two-phase incompressible Navier-Stokes flows [21, 22, 23, 24], particularly in the contexts involving contact line migration and axisymmetric geometric evolution.

In [25], Elliott and Fritz introduced the DeTurck flow, which reparameterizes the original geometric flow by coupling it with the harmonic heat flow in a reference domain. This reparameterization introduces an implicit tangential velocity component that enhances the distribution of mesh points on the evolving surface. In [26], Hu and Li demonstrated that as the time step approaches zero, the velocity produced by the BGN method asymptotically converges to a limiting velocity field $\boldsymbol{w}$ characterized by

$$\boldsymbol{w} \cdot \boldsymbol{n} = \boldsymbol{v} \cdot \boldsymbol{n}, \quad \Delta_\Gamma \boldsymbol{w} = \kappa \boldsymbol{n}, \tag{2}$$

where $\boldsymbol{v}$ is the velocity associated with the original geometric flow and κ is an auxiliary scalar function. Their analysis reveals that the tangential component defined by (2) minimizes the instantaneous surface deformation rate, thereby offering a theoretical explanation for the consistently high mesh quality observed in practice. Building on this line of research, Duan and Li recently proposed an artificial tangential motion strategy in [27], which enforces that the flow map $\boldsymbol{X}(\cdot, t) : \Gamma(0) \to \Gamma(t)$ minimizes the deformation energy

$$E[\boldsymbol{X}(\cdot, t)] = \frac{1}{2} \int_{\Gamma(0)} \left| \nabla_{\Gamma(0)} \boldsymbol{X}(\cdot, t) \right|^2,$$

subject to the constraint $\left(\frac{\partial \boldsymbol{X}}{\partial t} \circ \boldsymbol{X}^{-1}\right) \cdot \boldsymbol{n} = \boldsymbol{v} \cdot \boldsymbol{n}$. The corresponding Euler–Lagrange system reveals that this formulation enforces $\boldsymbol{X}(\cdot, t)$ to be a harmonic map with minimal deformation, thus effectively suppressing mesh distortion during surface evolution. Li et al. also presented a series of convergence analysis results based on the grid method [28, 29, 30, 31].

In this work, we are interested in the numerical simulation of geometric flows with unstructured point cloud data. Fokker-Planck equation (FPE) has been a tool to describe the density evolution of particles [32]. Given the velocity field $\boldsymbol{v}$ of a particle system in a Euclidean space, the trajectory of its density function ρ follows the FPE:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \boldsymbol{v}) = 0.$$

With this inspiration, through a coupling of geometric flows and specific FPEs, we propose a novel artificial tangential velocity model for stable numerical simulations of surface evolutions. In the Euclidean space, giving some target density p, it is known that if the velocity is chosen to be

$$\boldsymbol{v} = \nabla \log p - \nabla \log \rho,$$

then the density $\rho(t)$ will converge to the target density p along the trajectory of the FPE. This result can be generalized to general surfaces and helps to give appropriate tangential velocity to control the density of the points. To the end, we derive a coupled system of surface evolution with the extra tangential velocity and a transform of some FPE to avoid

undesired point clustering during surface evolution. The details can be found in Section 2. Distinguished from traditional approaches such as the surface finite element method, which rely on structured meshes and incur significant mesh generation costs, the proposed numerical methods fit well to a meshless framework based on the moving least squares approximation. This avoids the effort for mesh construction and enables flexible handling of unstructured point cloud data. Extensive numerical experiments demonstrate the robustness and effectiveness of the proposed methods across various point cloud scenarios.

The remainder of this paper is structured as follows. In Section 2, we derive the novel artificial tangential velocity by constructing a surface density diffusion equation. Section 3 presents the temporal and spatial discretization of the proposed model, including a brief overview of the moving least squares method for approximating differential operators on point clouds. Finally, we demonstrate the effectiveness and convergence of our approach through a series of numerical experiments in Section 4, highlighting its capability to characterize the evolution of surfaces given by point clouds and its robust application to MCF.

2. Tangential velocity model inspired from Fokker-Planck equation

2.1. Briefs of Fokker-Planck equations in Euclidean domain

Here, we give a brief introduction to FPEs [32]. To simplify the notation, we restrict ourselves to the Euclidean domain in this subsection. For a particle system governed by the velocity field $\boldsymbol{v}$,

$$\frac{d}{dt}\boldsymbol{X}(\boldsymbol{x},t) = \boldsymbol{v}(\boldsymbol{X}(\boldsymbol{x},t),t), \quad \boldsymbol{x}\in\mathbb{R}^d, \quad t>0,$$

it is known that the associate density ρ obeys the following FPE

$$\frac{\partial \rho}{\partial t} + \nabla\cdot(\rho\boldsymbol{v}) = 0. \tag{3}$$

Based on this FPE, with a given target density $p(\boldsymbol{x})$, we can design velocity field such that the density converges to the target density. One popular choice is to let

$$\boldsymbol{v}(\boldsymbol{x},t) = \nabla\log p(\boldsymbol{x}) - \nabla\log\rho(\boldsymbol{x},t). \tag{4}$$

This velocity field is derived from the gradient flow of Kullback-Leibler (KL) divergence

$$D_{KL}(\rho(t)||p) = \int_{\mathbb{R}^d}\rho(\boldsymbol{x},t)\log\frac{\rho(\boldsymbol{x},t)}{p(\boldsymbol{x})}d\boldsymbol{x}.$$

Calculating its time derivative and taking into account the equations (3) and (4), we have

$$\begin{aligned}\frac{d}{dt}D_{KL}(\rho(t)||p) &= \int_{\mathbb{R}^d}\frac{\partial}{\partial t}\rho(\boldsymbol{x},t)\log\frac{\rho(\boldsymbol{x},t)}{p(\boldsymbol{x})} + \rho(\boldsymbol{x},t)\frac{\partial}{\partial t}\log\rho(\boldsymbol{x},t)d\boldsymbol{x}\\ &= \int_{\mathbb{R}^d} -\nabla\cdot(\rho(\boldsymbol{x},t)\boldsymbol{v}(\boldsymbol{x},t))(1+\log\rho(\boldsymbol{x},t)-\log p(\boldsymbol{x}))d\boldsymbol{x}\\ &= \int_{\mathbb{R}^d}\rho(\boldsymbol{x},t)\boldsymbol{v}(\boldsymbol{x},t)\cdot(\nabla\log\rho(\boldsymbol{x},t)-\nabla\log p(\boldsymbol{x}))d\boldsymbol{x}\end{aligned}$$

$$= -\int_{\mathbb{R}^d} \rho(\boldsymbol{x},t)|\nabla \log \rho(\boldsymbol{x},t) - \nabla \log p(\boldsymbol{x})|^2 d\boldsymbol{x}.$$

As a density function, ρ is supposed to be nonnegative. This implies that with the velocity field in (4), $D_{KL}(\rho||p)$ is monotonically decreasing along the trajectory of (3).

Actually, given (4), the FPE (3) becomes

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \nabla \log p) = \Delta \rho.$$

In the next subsection, we will see that it is useful to introduce an auxiliary variable $s := \log \rho$, and subsequently we derive the following system of equations:

$$\begin{aligned} \frac{d}{dt}\boldsymbol{X}(\boldsymbol{x},t) =& (\nabla \log p - \nabla s)(\boldsymbol{X}(\boldsymbol{x},t),t), \\ \frac{d}{dt}s(\boldsymbol{X}(\boldsymbol{x},t),t) =& -\Delta \log p + \Delta s. \end{aligned}$$

If the target distribution is set to be uniform, i.e. $p = \text{const}$, then the above equations can be simplified to

$$\begin{aligned} \frac{d}{dt}\boldsymbol{X}(\boldsymbol{x},t) =& -\nabla s(\boldsymbol{X}(\boldsymbol{x},t),t), \\ \frac{d}{dt}s(\boldsymbol{X}(\boldsymbol{x},t),t) =& \Delta s. \end{aligned}$$

2.2. Tangential velocity model on surfaces

Now we go back to our surface evolution problems with the motivation of introducing tangential velocity field based on the surface diffusion of density function. Suppose that we add an extra tangential velocity field $\boldsymbol{v}_T$ to the surface velocity on the right-hand side of (1), then the surface evolution becomes

$$\begin{aligned} \frac{d}{dt}\boldsymbol{X}(\boldsymbol{x},t) &= \boldsymbol{v}(\boldsymbol{X}(\boldsymbol{x},t),t) + \boldsymbol{v}_T(\boldsymbol{X}(\boldsymbol{x},t),t), \quad \boldsymbol{x} \in \Gamma_0, \quad t \geq 0, \\ \boldsymbol{X}(\boldsymbol{x},0) &= \boldsymbol{x}, \qquad\qquad\qquad\qquad\qquad\quad \boldsymbol{x} \in \Gamma_0, \end{aligned} \tag{5}$$

where $\boldsymbol{v}_T(\boldsymbol{X}(\boldsymbol{x},t),t) \in \mathcal{T}_{\Gamma(t)}(\boldsymbol{X}(\boldsymbol{x},t))$, $\mathcal{T}_{\Gamma(t)}(\boldsymbol{X}(\boldsymbol{x},t))$ denotes the tangential space of $\Gamma(t)$ at $\boldsymbol{X}(\boldsymbol{x},t)$ and $\Gamma(t)$ is the surface corresponding to $\boldsymbol{X}(\cdot,t)$. Note that this extra tangential velocity will not change the shape of $\Gamma(t)$ but only the distribution of points comparing to the one associated with (1).

Let $\rho(\boldsymbol{X}(\boldsymbol{x},t),t)$ be the density function of $\Gamma(t)$, which describes the distribution of the points on the surface $\Gamma(t)$. With the given velocity field in (5), ρ is evolved following the FPE:

$$\begin{aligned} \partial_t^* \rho(\boldsymbol{X}(\boldsymbol{x},t),t) &= -\rho(\boldsymbol{X}(\boldsymbol{x},t),t)\mathrm{div}_{\Gamma(t)}(\boldsymbol{v}(\boldsymbol{X}(\boldsymbol{x},t),t) + \boldsymbol{v}_T(\boldsymbol{X}(\boldsymbol{x},t),t)), \quad \boldsymbol{x} \in \Gamma_0, \quad t \geq 0, \\ \rho(\boldsymbol{x},0) &= \rho_0(\boldsymbol{x}), \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \boldsymbol{x} \in \Gamma_0, \end{aligned}$$

where $\mathrm{div}_{\Gamma(t)}$ is the divergence operator on $\Gamma(t)$, and $\partial_t^* \rho$ is the material derivative, i.e.

$$\partial_t^* \rho(\boldsymbol{X}(\boldsymbol{x},t),t) = \frac{d}{dt}\rho(\boldsymbol{X}(\boldsymbol{x},t),t).$$

Here $\boldsymbol{X}(\cdot,t)$ is a diffeomorphism between Γ_0 and $\Gamma(t)$. Dividing by ρ on both sides, we have

$$\begin{aligned} \frac{d}{dt}\log\rho(\boldsymbol{X}(\boldsymbol{x},t),t) &= -\mathrm{div}_{\Gamma(t)}(\boldsymbol{v}(\boldsymbol{X}(\boldsymbol{x},t),t) + \boldsymbol{v}_T(\boldsymbol{X}(\boldsymbol{x},t),t)), \quad \boldsymbol{x}\in\Gamma_0, \quad t\geq 0, \\ \rho(\boldsymbol{x},0) &= \rho_0(\boldsymbol{x}), \qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \boldsymbol{x}\in\Gamma_0. \end{aligned}$$

Denote

$$s(\boldsymbol{X}(\boldsymbol{x},t),t) := \log\rho(\boldsymbol{X}(\boldsymbol{x},t),t),$$

then s satisfies

$$\begin{aligned} \frac{d}{dt}s(\boldsymbol{X}(\boldsymbol{x},t),t) &= -\mathrm{div}_{\Gamma(t)}(\boldsymbol{v}(\boldsymbol{X}(\boldsymbol{x},t),t) + \boldsymbol{v}_T(\boldsymbol{X}(\boldsymbol{x},t),t)), \quad \boldsymbol{x}\in\Gamma_0, \quad t\geq 0, \\ s(\boldsymbol{x},0) &= \log\rho_0(\boldsymbol{x}), \qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \boldsymbol{x}\in\Gamma_0. \end{aligned} \tag{6}$$

The key idea is to choose the extra tangential velocity field being

$$\boldsymbol{v}_T(\boldsymbol{X}(\boldsymbol{x},t),t) = -\eta\nabla_{\Gamma(t)}s(\boldsymbol{X}(\boldsymbol{x},t),t), \quad \eta>0,$$

where $\nabla_{\Gamma(t)}$ is the gradient operator on $\Gamma(t)$ and $\eta>0$ is a parameter. With the above tangential velocity field, the equation (6) becomes

$$\begin{aligned} \frac{d}{dt}s(\boldsymbol{X}(\boldsymbol{x},t),t) &= -\mathrm{div}_{\Gamma(t)}(\boldsymbol{v}(\boldsymbol{X}(\boldsymbol{x},t),t)) + \eta\Delta_{\Gamma(t)}s(\boldsymbol{X}(\boldsymbol{x},t),t), \quad \boldsymbol{x}\in\Gamma_0, \quad t\geq 0, \\ s(\boldsymbol{x},0) &= \log\rho_0(\boldsymbol{x}), \qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \boldsymbol{x}\in\Gamma_0, \end{aligned} \tag{7}$$

where $\Delta_{\Gamma(t)}$ is the Laplace-Beltrami operator on $\Gamma(t)$. Equation (7) is a diffusion equation with a source term. When the velocity $\boldsymbol{v}=0$ (no source anymore) and the surface $\Gamma(t)$ is fixed, its solution tends to a constant on a closed smooth surface. This inspires us to couple the evolutionary equation (7) with the geometric evolution for more uniform distribution, as well as the redistribution algorithm in the next subsection.

In summary, we propose the following coupled system of equations of which the numerical discretizations will lead to more stable numerical simulation of (1):

$$\begin{cases} \dfrac{d}{dt}\boldsymbol{X}(\boldsymbol{x},t) = \boldsymbol{v}(\boldsymbol{X}(\boldsymbol{x},t),t) - \eta\nabla_{\Gamma(t)}s(\boldsymbol{X}(\boldsymbol{x},t),t), & \boldsymbol{x}\in\Gamma_0, \quad t\geq 0, \\ \dfrac{d}{dt}s(\boldsymbol{X}(\boldsymbol{x},t),t) = -\mathrm{div}_{\Gamma(t)}\boldsymbol{v}(\boldsymbol{X}(\boldsymbol{x},t),t) + \eta\Delta_{\Gamma(t)}s(\boldsymbol{X}(\boldsymbol{x},t),t), & \boldsymbol{x}\in\Gamma_0, \quad t\geq 0, \\ \boldsymbol{X}(\boldsymbol{x},0) = \boldsymbol{x}, \quad s(\boldsymbol{x},0) = \log\rho_0(\boldsymbol{x}), & \boldsymbol{x}\in\Gamma_0. \end{cases} \tag{8}$$

2.3. Points redistribution

In the case that the distribution of the points becomes non-uniform even with the extra tangential velocity, we propose to decouple the surface evolution and the FPE, and add a point redistribution step by fixing the surface. More precisely, we set $\boldsymbol{v}$ to be zero in (8) and the surface is fixed in this case. Then we solve the FPE on this fixed surface and make the points uniformly distributed over the surface. After this redistribution step, we go back to (8) and resume the evolution of the surface with the velocity at the last stopping time.

Moreover, in the redistribution step, we can choose other target distribution beside the uniform distribution. In some applications, it is better to have the distribution dependent on the geometrical structure of the surface, e.g., curvature. Our model is capable of providing this flexibility. Denote the target distribution to be p, then the redistribution step is to solve the following equation:

$$\begin{aligned} \frac{d}{dt}\boldsymbol{X}(\boldsymbol{x},t) &= \eta\nabla_{\Gamma(t)}\log p(\boldsymbol{X}(\boldsymbol{x},t)) - \eta\nabla_{\Gamma(t)}s(\boldsymbol{X}(\boldsymbol{x},t),t), \quad \boldsymbol{x}\in\Gamma_t, \quad t\geq 0, \\ \frac{d}{dt}s(\boldsymbol{X}(\boldsymbol{x},t),t) &= -\eta\Delta_{\Gamma(t)}\log p(\boldsymbol{X}(\boldsymbol{x},t)) + \eta\Delta_{\Gamma(t)}s(\boldsymbol{X}(\boldsymbol{x},t),t), \quad \boldsymbol{x}\in\Gamma_t, \quad t\geq 0. \end{aligned} \tag{9}$$

Remark 1. *With fixed surface Γ and target distribution p, (9) is known as the gradient flow associate with the KL divergence*

$$D_{KL}(q||p) = \int_\Gamma q(z)\log\frac{q(z)}{p(z)}dz,$$

for $q = \exp s$. See the analogous discussion in Euclidean domain from Section 2.1.

3. Discretization over point cloud

Since the surfaces are presented by point clouds, we employ the moving least square method to approximate the differential operator on point clouds and use backward differentiation formula (BDF) schemes in temporal direction.

3.1. The moving least square method

We first discuss derivatives of functions defined on hypersurfaces. Let $\Gamma\subset\mathbb{R}^3$ be a hypersurface and suppose that it is locally parameterized by $(\alpha,\beta)\in\mathbb{R}^2$. For a given patch of Γ, it can be written as $\Gamma(\alpha,\beta) = (x(\alpha,\beta), y(\alpha,\beta), z(\alpha,\beta))$. The metric tensor $G = [g_{ij}]$ is represented by $g_{ij} =< \Gamma_\alpha, \Gamma_\beta >$, where $\Gamma_\alpha = (x_\alpha, y_\alpha, z_\alpha)$ and $\Gamma_\beta = (x_\beta, y_\beta, z_\beta)$ are two tangent vectors at $\boldsymbol{x}\in\Gamma$. The tangent space $T_{\boldsymbol{x}}\Gamma$ at $\boldsymbol{x}\in\Gamma$ is then spanned by $\Gamma_\alpha(\boldsymbol{x})$ and $\Gamma_\beta(\boldsymbol{x})$. Let $f:\Gamma\to\mathbb{R}$ and $f\in C^2(\Gamma)$. Under this parameterization, one can calculate the gradient of f using the formula below [33]

$$\nabla_\Gamma f = [\Gamma_\alpha,\Gamma_\beta]\,G^{-1}\nabla f = \left(g^{11}\frac{\partial f}{\partial\alpha} + g^{12}\frac{\partial f}{\partial\beta}\right)\Gamma_\alpha + \left(g^{21}\frac{\partial f}{\partial\alpha} + g^{22}\frac{\partial f}{\partial\beta}\right)\Gamma_\beta,$$

where g^{ij} are the components of G^{-1}, the inverse of the metric tensor G. Denote $g = \det(G)$, the Laplace-Beltrami operation to f is

$$\Delta_\Gamma f = \frac{1}{\sqrt{g}} \left(\frac{\partial}{\partial \alpha} \left(\sqrt{g} g^{11} \frac{\partial f}{\partial \alpha} \right) + \frac{\partial}{\partial \alpha} \left(\sqrt{g} g^{12} \frac{\partial f}{\partial \beta} \right) + \frac{\partial}{\partial \beta} \left(\sqrt{g} g^{21} \frac{\partial f}{\partial \alpha} \right) + \frac{\partial}{\partial \beta} \left(\sqrt{g} g^{22} \frac{\partial f}{\partial \beta} \right) \right). \tag{10}$$

In this article, only point clouds sampled from hypersurfaces are available, but not the surfaces themselves. Thus, in the rest of this section, we briefly review the moving least squares method (MLS) [34] to approximate the surfaces in a local coordinate system and then compute the metric tensors as well as the derivatives of functions at each point. First of all, we use principal component analysis to construct the local coordinate system. Given a point cloud $\boldsymbol{P} = \{\boldsymbol{p}_i | i = 1, 2, \ldots, N_{\boldsymbol{x}}\}$ with $N_{\boldsymbol{x}}$ points sampled from a two-dimensional manifold in $\mathbb{R}^3$. Define $\Lambda(\boldsymbol{p}_i)$ to be the set of adjacent points to $\boldsymbol{x}_i$ obtained by the K-nearest-neighbor (KNN) method. Define the covariance matrix P_i at $\boldsymbol{x}_i$, $P_i = \sum_{k \in \Lambda(\boldsymbol{p}_i)} (\boldsymbol{p}_k - \boldsymbol{c}_i)^T (\boldsymbol{p}_k - \boldsymbol{c}_i)$, where $\boldsymbol{c}_i$ is the local barycenter $\boldsymbol{c}_i = \frac{1}{K} \sum_{k \in \Lambda(\boldsymbol{p}_i)} \boldsymbol{p}_k$. Then through P_i, we can obtain the sorted eigenvalues $\lambda_{i,1} > \lambda_{i,2} > \lambda_{i,3}$ and the corresponding eigenvectors $(\boldsymbol{e}_{i,1}, \boldsymbol{e}_{i,2}, \boldsymbol{e}_{i,3})$.

Since the surface we considering is two-dimensional, in the local coordinate system $\{\boldsymbol{x}_i; \boldsymbol{e}_{i,1}, \boldsymbol{e}_{i,2}, \boldsymbol{e}_{i,3}\}$ of the point $\boldsymbol{x}_i$ and local coordinates $(\alpha_i, \beta_i, \gamma_i)$, the local second-order binary polynomial $\gamma_i(\alpha, \beta)$ is obtained by minimizing the following weighted sum:

$$\sum_{k \in \Lambda(\boldsymbol{x}_i)} \omega(\|\boldsymbol{x}_k - \boldsymbol{x}_i\|)(\gamma_i(\alpha_{i,k}, \beta_{i,k}) - \gamma_{i,k})^2,$$

where $\Lambda(\boldsymbol{x}_i)$ is the set of adjacent points to $\boldsymbol{x}_i$, and $\omega(d_k)$ is the weight coefficient with $d_k = \|\boldsymbol{x}_k - \boldsymbol{x}\|$ being the Euclidean distances between $\boldsymbol{x}$ and the position of data point $\boldsymbol{x}_k$. Thus, $\Gamma_i = (\alpha, \beta, \gamma_i(\alpha, \beta))$ is a smooth approximation of some underlying surface near the point $\boldsymbol{x}_i$ under the local coordinate system $\{\boldsymbol{x}_i; \boldsymbol{e}_{i,1}, \boldsymbol{e}_{i,2}, \boldsymbol{e}_{i,3}\}$.

Remark 2. *Although the weight function in MLS does not affect the calculation accuracy, the stability will be challenged. There are many alternative functions in [33]. We use the most popular Wendland function*

$$\omega(d) = \left(1 - \frac{d}{D}\right)^4 \left(\frac{4d}{D} + 1\right),$$

which is defined on the interval $d \in [0, D]$ *and* $\omega(0) = 1$, $\omega(D) = 0$, $\omega'(0) = 0$ *and* $\omega''(0) = 0$.

Particularly, we use the local binary second-order polynomials $\gamma_i(\alpha, \beta) = c_0 + c_1\alpha + c_2\beta + c_3\alpha^2 + c_4\alpha\beta + c_5\beta^2$. The local basis consists of two tangent vectors given by $\Gamma_\alpha(\boldsymbol{x}_i) = (1, 0, \frac{\partial \gamma_i}{\partial \alpha}) = (1, 0, c_1)$ and $\Gamma_\beta(\boldsymbol{x}_i) = (0, 1, \frac{\partial \gamma_i}{\partial \beta}) = (0, 1, c_2)$. Based on this local polynomial surface, we compute the metric tensor $G(\boldsymbol{x})$ and its inverse $G^{-1}(\boldsymbol{x})$ which are functions

dependent on γ_i. Then, the gradient in local coordinate system can be computed as follows

$$\begin{aligned}\nabla_\Gamma f(\boldsymbol{x}_i) = \left(g^{11}(\boldsymbol{x}_i)\frac{\partial f}{\partial \alpha}(\boldsymbol{x}_i) + g^{12}(\boldsymbol{x}_i)\frac{\partial f}{\partial \beta}(\boldsymbol{x}_i)\right)\Gamma_\alpha(\boldsymbol{x}_i) \\ + \left(g^{21}(\boldsymbol{x}_i)\frac{\partial f}{\partial \alpha}(\boldsymbol{x}_i) + g^{22}(\boldsymbol{x}_i)\frac{\partial f}{\partial \beta}(\boldsymbol{x}_i)\right)\Gamma_\beta(\boldsymbol{x}_i),\end{aligned} \tag{11}$$

and Laplace-Beltrami operation is realized via

$$\begin{aligned}\Delta_\Gamma f(\boldsymbol{x}_i) = A_0(\boldsymbol{x}_i)\frac{\partial f}{\partial \alpha}(\boldsymbol{x}_i) + A_1(\boldsymbol{x}_i)\frac{\partial f}{\partial \beta}(\boldsymbol{x}_i) + A_2(\boldsymbol{x}_i)\frac{\partial^2 f}{\partial \alpha^2}(\boldsymbol{x}_i) \\ + A_3(\boldsymbol{x}_i)\frac{\partial^2 f}{\partial \alpha \partial \beta}(\boldsymbol{x}_i) + A_4(\boldsymbol{x}_i)\frac{\partial^2 f}{\partial \beta^2}(\boldsymbol{x}_i),\end{aligned} \tag{12}$$

where

$$\begin{aligned}
&g^{11}(\boldsymbol{x}_i) = \frac{1+c_2^2}{1+c_1^2+c_2^2}, \quad g^{12}(\boldsymbol{x}_i) = \frac{-c_1c_2}{1+c_1^2+c_2^2}, \\
&g^{21}(\boldsymbol{x}_i) = \frac{-c_1c_2}{1+c_1^2+c_2^2}, \quad g^{22}(\boldsymbol{x}_i) = \frac{1+c_1^2}{1+c_1^2+c_2^2}. \\
&A_0(\boldsymbol{x}_i) = \frac{1}{\sqrt{g(\boldsymbol{x}_i)}}\left((\sqrt{g})_\alpha(\boldsymbol{x}_i)g^{11}(\boldsymbol{x}_i) + (\sqrt{g})_\beta(\boldsymbol{x}_i)g^{21}(\boldsymbol{x}_i)\right) + g^{11}_\alpha(\boldsymbol{x}_i) + g^{21}_\beta(\boldsymbol{x}_i), \\
&A_1(\boldsymbol{x}_i) = \frac{1}{\sqrt{g(\boldsymbol{x}_i)}}\left((\sqrt{g})_\alpha(\boldsymbol{x}_i)g^{12}(\boldsymbol{x}_i) + (\sqrt{g})_\beta(\boldsymbol{x}_i)g^{22}(\boldsymbol{x}_i)\right) + g^{12}_\alpha(\boldsymbol{x}_i) + g^{22}_\beta(\boldsymbol{x}_i), \\
&A_2(\boldsymbol{x}_i) = g^{11}(\boldsymbol{x}_i), \quad A_3(\boldsymbol{x}_i) = \left(g^{12}(\boldsymbol{x}_i) + g^{21}(\boldsymbol{x}_i)\right), \quad A_4(\boldsymbol{x}_i) = g^{22}(\boldsymbol{x}_i).
\end{aligned} \tag{13}$$

The formulas in (12) are derived from Equation (10). Note that these approximated quantities depend only on local polynomial coefficients $c_1, c_2, ..., c_5$.

We are now ready to reconstruct the function and its derivatives under these local coordinate systems with the quantities estimated in (13). Similarly, we reconstruct the function f_s locally from a set of function values $\{f_{i,k}\}$ using second-order binary polynomial $f_i(\alpha, \beta)$ in the local coordinate system $\{\boldsymbol{x}_i; \boldsymbol{e}_{i,1}, \boldsymbol{e}_{i,2}, \boldsymbol{e}_{i,3}\}$ via minimizing the following weighted sum

$$\sum_{k\in\Lambda(\boldsymbol{x}_i)} \omega(\|\boldsymbol{p}_k - \boldsymbol{x}_i\|)(f_i(\alpha_{i,k}, \beta_{i,k}) - f_{i,k})^2.$$

Then suppose $f_i(\alpha, \beta) = \bar{c}_0 + \bar{c}_1\alpha + \bar{c}_2\beta + \bar{c}_3\alpha^2 + \bar{c}_4\alpha\beta + \bar{c}_5\beta^2$. We update the formulas in (11) and (12) taking into account that the derivatives at $\boldsymbol{x}_i$ is at the origin in the local coordinate system, i.e., $\alpha = 0$ and $\beta = 0$. Then we end up with

$$\begin{aligned}
&\nabla_\Gamma f(\boldsymbol{x}_i) = \left(g^{11}(\boldsymbol{x}_i)\bar{c}_1 + g^{12}(\boldsymbol{x}_i)\bar{c}_2\right)(\boldsymbol{e}_{i,1} + c_1\boldsymbol{e}_{i,3}) + \left(g^{21}(\boldsymbol{x}_i)\bar{c}_1 + g^{22}(\boldsymbol{x}_i)\bar{c}_2\right)(\boldsymbol{e}_{i,2} + c_2\boldsymbol{e}_{i,3}), \\
&\Delta_\Gamma f(\boldsymbol{x}_i) = A_0(\boldsymbol{x}_i)\bar{c}_1 + A_1(\boldsymbol{x}_i)\bar{c}_2 + 2A_2(\boldsymbol{x}_i)\bar{c}_3 + A_3(\boldsymbol{x}_i)\bar{c}_4 + 2A_4(\boldsymbol{x}_i)\bar{c}_5,
\end{aligned} \tag{14}$$

where g^{ij}, A_i are quantities provided in (13).

3.2. Time discretization

In this part, we consider time semi-discretization of the system of equations (8). Let time steps $t_i = (i-1)\Delta t$, $i = 1, 2, \cdots, N_t$, $\tau = \frac{T}{N_t - 1}$. The time derivative $\frac{d\boldsymbol{X}(\cdot,t)}{dt}$ is approximated using BDF:

$$\frac{d}{dt}\boldsymbol{X}(\cdot,t_k) \sim \begin{cases} \frac{\boldsymbol{X}(\cdot,t_k)-\boldsymbol{X}(\cdot,t_{k-1})}{\tau} + \mathcal{O}(\tau^1), \\ \frac{\frac{3}{2}\boldsymbol{X}(\cdot,t_k)-2\boldsymbol{X}(\cdot,t_{k-1})+\frac{1}{2}\boldsymbol{X}(\cdot,t_{k-2})}{\tau} + \mathcal{O}(\tau^2), \\ \frac{\frac{11}{6}\boldsymbol{X}(\cdot,t_k)-3\boldsymbol{X}(\cdot,t_{k-1})+\frac{3}{2}\boldsymbol{X}(\cdot,t_{k-2})-\frac{1}{3}\boldsymbol{X}(\cdot,t_{k-3})}{\tau} + \mathcal{O}(\tau^3), \end{cases} \tag{15}$$

and the material derivative of s is approximately computed as

$$\frac{d}{dt}s(\boldsymbol{X}_k,t_k) \sim \begin{cases} \frac{s(\boldsymbol{X}_k,t_k)-s(\boldsymbol{X}_{k-1},t_{k-1})}{\tau} + \mathcal{O}(\tau^1), \\ \frac{\frac{3}{2}s(\boldsymbol{X}_k,t_k)-2s(\boldsymbol{X}_{k-1},t_{k-1})+\frac{1}{2}s(\boldsymbol{X}_{k-2},t_{k-2})}{\tau} + \mathcal{O}(\tau^2), \\ \frac{\frac{11}{6}s(\boldsymbol{X}_k,t_k)-3s(\boldsymbol{X}_{k-1},t_{k-1})+\frac{3}{2}s(\boldsymbol{X}_{k-2},t_{k-2})-\frac{1}{3}s(\boldsymbol{X}_{k-3},t_{k-3})}{\tau} + \mathcal{O}(\tau^3). \end{cases} \tag{16}$$

To simplify the notation, we use $\boldsymbol{X}_k = \boldsymbol{X}(\cdot,t_k)$ and $s_k = s(\boldsymbol{X}_k,t_k)$.

Combining (16) and (15), we get the time semi-discrete format of (8):

$$\begin{aligned} \frac{a\boldsymbol{X}_k - \hat{\boldsymbol{X}}_{k-1}}{\tau} &= \boldsymbol{v}(\bar{\boldsymbol{X}}_k,t_k) - \eta\nabla_{\bar{\Gamma}_k}s_k, \\ \frac{as_k - \hat{s}_{k-1}}{\tau} &= -\mathrm{div}_{\bar{\Gamma}_k}\boldsymbol{v}(\bar{\boldsymbol{X}}_k,t_k) + \eta\Delta_{\bar{\Gamma}_k}s_k, \\ \boldsymbol{X}(\boldsymbol{x},0) &= \boldsymbol{x}, \quad s(\boldsymbol{x},0) = \log\rho_0(\boldsymbol{x}), \end{aligned} \tag{17}$$

where a, $\hat{\boldsymbol{X}}_{k-1}$ and $\hat{s}_{k-1}$ are defined as

$$\begin{aligned} &\text{BDF1: } a = 1, && \hat{\boldsymbol{X}}_{k-1} = \boldsymbol{X}_{k-1}, && \hat{s}_{k-1} = s_{k-1}, \\ &\text{BDF2: } a = \frac{3}{2}, && \hat{\boldsymbol{X}}_{k-1} = 2\boldsymbol{X}_{k-1} - \frac{1}{2}\boldsymbol{X}_{k-2}, && \hat{s}_{k-1} = 2s_{k-1} - \frac{1}{2}s_{k-2}, \\ &\text{BDF3: } a = \frac{11}{6}, && \hat{\boldsymbol{X}}_{k-1} = 3\boldsymbol{X}_{k-1} - \frac{3}{2}\boldsymbol{X}_{k-2} + \frac{1}{3}\boldsymbol{X}_{k-3}, && \hat{s}_{k-1} = 3s_{k-1} - \frac{3}{2}s_{k-2} + \frac{1}{3}s_{k-3}. \end{aligned} \tag{18}$$

$\bar{\boldsymbol{X}}_k$ in (17) is the BDF-k extrapolation, e.g.,

$$\begin{aligned} &\text{BDF1:} && \bar{\boldsymbol{X}}_k = \boldsymbol{X}_{k-1}, \\ &\text{BDF2:} && \bar{\boldsymbol{X}}_k = 2\boldsymbol{X}_{k-1} - \boldsymbol{X}_{k-2}, \\ &\text{BDF3:} && \bar{\boldsymbol{X}}_k = 3\boldsymbol{X}_{k-1} - 3\boldsymbol{X}_{k-2} + \boldsymbol{X}_{k-3}. \end{aligned} \tag{19}$$

Moreover, when applying higher-order BDF methods (e.g., BDF2 and BDF3), suitable initial values should be provided to reach the desired temporal accuracy. To mitigate the impact of low-accuracy initial approximations, we adopt the fourth-order Runge–Kutta (RK4) method [35]. The implementation details with respect to BDF2 and BDF3 schemes are provided in Algorithm 1 and Algorithm 2, respectively. The BDF1 scheme, which is rather straightforward, is omitted for brevity.

Algorithm 1 BDF2 Algorithm

Input: Sample points $\boldsymbol{X}_0 = \boldsymbol{x}$ on initial surface $\Gamma(0)$, Initial value s_0, Time step τ, End time T;

Output: Discrete solution $\boldsymbol{X}_{T/\tau}$, $s_{T/\tau}$ at end time;

1: Taking $\boldsymbol{X}_0$, s_0, τ and $\bar{\Gamma}_1 = \{\boldsymbol{X}_0^i | i = 1, \ldots, N_{\boldsymbol{x}}\}$ as inputs, calculate $\boldsymbol{X}_1$, s_1 using the BDF1 format of (17);
2: Let $k = 1$;
3: **while** $k < T/\tau$ **do**
4: Taking $\boldsymbol{X}_{k-1}$ and $\boldsymbol{X}_{k-2}$ as inputs, calculate $\bar{\boldsymbol{X}}_k$ using the BDF2 format of (19);
5: Taking $\boldsymbol{X}_{k-1}$, $\boldsymbol{X}_k$, s_{k-1}, s_k, τ and $\bar{\Gamma}_{k+1} = \{\boldsymbol{X}_k^i | i = 1, \ldots, N_{\boldsymbol{x}}\}$ as inputs, calculate $\boldsymbol{X}_{k+1}$, s_{k+1} using the BDF2 format of (17);
6: Let $k = k + 1$;
7: **end while**
8: **return** $\boldsymbol{X}_{T/\tau}, s_{T/\tau}$.

Algorithm 2 BDF3 Algorithm

Input: Sample points $\boldsymbol{X}_0 = \boldsymbol{x}$ on initial surface $\Gamma(0)$, Initial value s_0, Time step τ, End time T;

Output: Discrete solution $\boldsymbol{X}_{T/\tau}$, $s_{T/\tau}$ at end time;

1: Taking $\boldsymbol{X}_0$, s_0, τ and $\bar{\Gamma}_1 = \{\boldsymbol{X}_0^i | i = 1, \ldots, N_{\boldsymbol{x}}\}$ as inputs, calculate $\boldsymbol{X}_1$, s_1 using the RK4 method;
2: Taking $\boldsymbol{X}_1$, s_1, τ and $\bar{\Gamma}_2 = \{\boldsymbol{X}_1^i | i = 1, \ldots, N_{\boldsymbol{x}}\}$ as inputs, calculate $\boldsymbol{X}_2$, s_2 using the BDF2 format of (17);
3: Let $k = 2$;
4: **while** $k < T/\tau$ **do**
5: Taking $\boldsymbol{X}_{k-1}$, $\boldsymbol{X}_{k-2}$ and $\boldsymbol{X}_{k-3}$ as inputs, calculate $\bar{\boldsymbol{X}}_k$ using the BDF3 format of (19);
6: Taking $\boldsymbol{X}_{k-2}$, $\boldsymbol{X}_{k-1}$, $\boldsymbol{X}_k$, s_{k-2}, s_{k-1}, s_k, τ and $\bar{\Gamma}_{k+1} = \{\boldsymbol{X}_k^i | i = 1, \ldots, N_{\boldsymbol{x}}\}$ as inputs, calculate $\boldsymbol{X}_{k+1}$, s_{k+1} using the BDF3 format of (17);
7: Let $k = k + 1$;
8: **end while**
9: **return** $\boldsymbol{X}_{T/\tau}, s_{T/\tau}$.

3.3. Fully discrete schemes on point clouds

For the spatial discretization, we consider a point cloud $\boldsymbol{P} = \{\boldsymbol{x}_i | i = 1, 2, \ldots, N_{\boldsymbol{x}}\}$ with $N_{\boldsymbol{x}}$ points sampled from a two-dimensional manifold in $\mathbb{R}^3$. The scalar function $f_s : \Gamma \to \mathbb{R}$ is approximated by $F = [f_1, f_2, \ldots, f_{N_{\boldsymbol{x}}}]^T$ in the sense of $f_i \sim f_s(\boldsymbol{x}_i)$ and (14) becomes

$$\nabla_\Gamma f_s \sim [M_{G1}\ M_{G2}\ M_{G3}]F = M_G F, \quad \Delta_\Gamma f_s \sim M_L F. \tag{20}$$

In (20), M_{G1}, M_{G2}, M_{G3} and M_L are $N_{\boldsymbol{x}} \times N_{\boldsymbol{x}}$ matrices which are typically sparse. Then, we use (20) to obtain the full-discrete form of (8):

$$\begin{aligned} \frac{a}{\tau}\boldsymbol{X}_k + \eta M_G^T(\bar{\boldsymbol{X}}_k)s_k &= \frac{1}{\tau}\hat{\boldsymbol{X}}_{k-1} + \boldsymbol{v}(\bar{\boldsymbol{X}}_k, t_k), \\ \frac{a}{\tau}s_k - \eta M_L(\bar{\boldsymbol{X}}_k)s_k &= \frac{1}{\tau}\hat{s}_{k-1} - M_G(\bar{\boldsymbol{X}}_k)\boldsymbol{v}(\bar{\boldsymbol{X}}_k, t_k), \\ \boldsymbol{X}(\boldsymbol{x}, 0) = \boldsymbol{x}, \quad s(\boldsymbol{x}, 0) &= \log \rho_0(\boldsymbol{x}), \end{aligned} \tag{21}$$

where a, $\hat{\boldsymbol{X}}_{k-1}$, $\hat{S}_{k-1}$ and $\bar{\boldsymbol{X}}_k$ are defined in (18). In the case of point clouds, in order to estimate the density $\hat{\rho}_0(\boldsymbol{x}_i)$ of the underlying surfaces, we rely on some reconstructed triangle template $\Lambda_\triangle(\boldsymbol{x}_i)$ out of the point clouds. The construction procedure for these triangles is detailed in [36], which involves forming Delaunay triangles from points $\Lambda(\boldsymbol{x}_i)$ in the vicinity of $\boldsymbol{x}_i$ and subsequently excluding any triangles that do not include $\boldsymbol{x}_i$ as a vertex. Then, the initial density $\rho_0(\boldsymbol{x})$ is estimated via the following formula:

$$\rho_0(\boldsymbol{x}_i) \propto \hat{\rho}_0(\boldsymbol{x}_i) = \frac{1}{S(\boldsymbol{x}_i)} = \frac{3}{\sum\limits_{K_j \in \Lambda_\triangle(\boldsymbol{x}_i)} (Area(K_j))}. \tag{22}$$

Since the velocity field $\boldsymbol{v}(\boldsymbol{X}(\boldsymbol{x}, t), t)$ is given arbitrarily, and we use a constant η in our model for simplicity, the tangential velocity field $\boldsymbol{v}_T(\boldsymbol{X}(\boldsymbol{x}, t), t)$ might still be insufficient to counteract the component of the given velocity $\boldsymbol{v}(\boldsymbol{X}(\boldsymbol{x}, t), t)$ in the tangential direction. The aggregation effect of points may still exist in such cases. This phenomenon has been observed in some of our subsequent experiments, cf. Figure 4.3.5 for details. In this case, we can add point redistribution step as mentioned in Section 2.3. The core idea is to halt the normal motion by setting $\boldsymbol{v} = 0$ in (21), and solve the following equation:

$$\begin{aligned} \frac{a}{\tau}\boldsymbol{X}_k + \eta M_G^T(\bar{\boldsymbol{X}}_k)s_k &= \frac{1}{\tau}\hat{\boldsymbol{X}}_{k-1}, \\ \frac{a}{\tau}s_k - \eta M_L(\bar{\boldsymbol{X}}_k)s_k &= \frac{1}{\tau}\hat{s}_{k-1}, \\ \boldsymbol{X}(\boldsymbol{x}, T) = \boldsymbol{X}_T, \quad s(\boldsymbol{x}, T) &= \log \rho_T(\boldsymbol{x}). \end{aligned} \tag{23}$$

ρ_T obtained via (22) at time T. The computational procedure is outlined in Algorithm 3. Naturally, the redistribution can be invoked once the variation in s exceeds a predefined threshold, thereby maintaining accuracy throughout the surface evolution process.

Once a target distribution $p(\boldsymbol{x})$ is desired to be matched, the equation (9) is discretized as follow:

$$\begin{aligned} \frac{a}{\tau}\boldsymbol{X}_k + \eta M_G^T(\bar{\boldsymbol{X}}_k)s_k &= \frac{1}{\tau}\hat{\boldsymbol{X}}_{k-1} + \eta M_G^T(\bar{\boldsymbol{X}}_k)\log p(\bar{\boldsymbol{X}}_k), \\ \frac{a}{\tau}s_k - \eta M_L(\bar{\boldsymbol{X}}_k)s_k &= \frac{1}{\tau}\hat{s}_{k-1} - \eta M_L(\bar{\boldsymbol{X}}_k)\log p(\bar{\boldsymbol{X}}_k), \\ \boldsymbol{X}(\boldsymbol{x}, T) = \boldsymbol{X}_T, \quad s(\boldsymbol{x}, T) &= \log \rho_T(\boldsymbol{x}). \end{aligned} \tag{24}$$

The computational procedure is outlined in Algorithm 4.

Algorithm 3 Re-distribution Algorithm

Input: Evolution points $\boldsymbol{X}_T$, Initial value s_T, Estimated value $\hat{\rho}_T$, Time step τ, Small time step $\tilde{\tau} \sim \tau^2$, Threshold $\varepsilon_0 = 5e-5$;

Output: Discrete solution $\boldsymbol{X}_c$, s_c at end time;

Correction $\hat{s}_T = \log \hat{\rho}_T$;
Taking $\boldsymbol{X}_T$, $\hat{s}_T$, τ and $\bar{\Gamma}_{T+1} = \{\boldsymbol{X}_T^i | i = 1, \ldots, N_{\boldsymbol{x}}\}$ as inputs, calculate $\boldsymbol{X}_{T+1}$, s_{T+1} using the BDF1 format of (23);
Let $k = 1$ and $\varepsilon_s = 2$;
while $\varepsilon_s > \varepsilon_0$ **do**
    Taking $\boldsymbol{X}_{T+k-1}$ and $\boldsymbol{X}_{T+k-2}$ as inputs, calculate $\bar{\boldsymbol{X}}_{T+k}$ using the BDF2 format of (19);
    Taking $\boldsymbol{X}_{T+k-1}$, $\boldsymbol{X}_{T+k}$, s_{T+k-1}, s_{T+k}, τ and $\bar{\Gamma}_{T+k+1} = \{\boldsymbol{X}_{T+k}^i | i = 1, \ldots, N_{\boldsymbol{x}}\}$ as inputs, calculate $\boldsymbol{X}_{T+k+1}$, s_{T+k+1} using the BDF2 format of (23);
    Calculate $\varepsilon_s = |\max(s_{T+k+1}) - \min(s_{T+k+1})|$;
    Let $k = k + 1$;
end while
return $\boldsymbol{X}_c, s_c$.

Algorithm 4 Target distribution matching Algorithm

Input: Evolution points $\boldsymbol{X}_T$, Initial value s_T, Target distribution p, Estimated value $\hat{\rho}_T$, Time step τ, Small time step $\tilde{\tau} \sim \tau^2$, Threshold $\varepsilon_0 = 5e-5$;

Output: Discrete solution $\boldsymbol{X}_c$, s_c at end time;

Correction $\hat{s}_T = \log \hat{\rho}_T$;
Taking $\boldsymbol{X}_T$, $\hat{s}_T$, τ and $\bar{\Gamma}_{T+1} = \{\boldsymbol{X}_T^i | i = 1, \ldots, N_{\boldsymbol{x}}\}$ as inputs, calculate $\boldsymbol{X}_{T+1}$, s_{T+1} using the BDF1 format of (24);
Let $k = 1$ and $\varepsilon_s = 2$;
while $\varepsilon_s > \varepsilon_0$ **do**
    Taking $\boldsymbol{X}_{T+k-1}$ and $\boldsymbol{X}_{T+k-2}$ as inputs, calculate $\bar{\boldsymbol{X}}_{T+k}$ using the BDF2 format of (19);
    Taking $\boldsymbol{X}_{T+k-1}$, $\boldsymbol{X}_{T+k}$, s_{T+k-1}, s_{T+k}, τ and $\bar{\Gamma}_{T+k+1} = \{\boldsymbol{X}_{T+k}^i | i = 1, \ldots, N_{\boldsymbol{x}}\}$ as inputs, calculate $\boldsymbol{X}_{T+k+1}$, s_{T+k+1} using the BDF2 format of (24);
    Calculate $\varepsilon_s = |\max(s_{T+k+1}) - \min(s_{T+k+1})|$;
    Let $k = k + 1$;
end while
return $\boldsymbol{X}_c, s_c$.

4. Numerical Experiments

In this section, we present results of numerical experiments to show the convergence, feasibility and efficiency of the proposed method, with particular focus on the improvement to the distribution of points in the evolution. Moreover, we apply the proposed method to simulating the MCF motion by point clouds. All source code and datasets used in the experiments are available at https://github.com/Poker-Pan/EMSL-SE. We emphasis that

in all the experiments, only point clouds are used and the triangles are merely for the display of our results.

In addition, in the subsequent experiments, to demonstrate the performance of our method, we use the area element of the point, which is defined as follows

$$S_p(\boldsymbol{x}_i) = \frac{1}{3} \sum_{K_j \in \Lambda_\triangle(\boldsymbol{x}_i)} (Area(K_j)), \tag{25}$$

where $\Lambda_\triangle(\boldsymbol{x}_i)$ is a set composed of triangles. It is a Delaunay triangle formed by the points in $\Lambda(\boldsymbol{x}_i)$, and it excludes triangles that do not contain fixed points in $\boldsymbol{x}_i$.

4.1. Validation of temporal convergence

We test the convergence rate of the proposed method using a sphere $\Gamma(0) = \{\boldsymbol{x} \in \mathbb{R}^3, |\boldsymbol{x}| = \frac{1}{2}\}$ as the initial surface where the point cloud is sampled. The velocity field $\boldsymbol{v}(\boldsymbol{X}(\boldsymbol{x}, t), t)$ is given as follows:

$$\boldsymbol{v}(\boldsymbol{X}(\boldsymbol{x}, t), t) = \boldsymbol{X}(\boldsymbol{x}, t)(1 - |\boldsymbol{X}(\boldsymbol{x}, t)|).$$

The error of the numerical solution at time $t = T$, as provided by the proposed method with step size τ, number of vertices $N_{\boldsymbol{x}}$, is measured by

$$\varepsilon^t_{\boldsymbol{x}}(N_{\boldsymbol{x}}, \tau) = \max_{j=1,\ldots,N_{\boldsymbol{x}}} |\boldsymbol{X}_{N_{\boldsymbol{x}},\tau}(\boldsymbol{x}_j, T) - \boldsymbol{X}_{N_{\boldsymbol{x}},2\tau}(\boldsymbol{x}_j, T)|,$$
$$\varepsilon^t_{s}(N_{\boldsymbol{x}}, \tau) = \max_{j=1,\ldots,N_{\boldsymbol{x}}} |S_{N_{\boldsymbol{x}},\tau}(\boldsymbol{x}_j, T) - S_{N_{\boldsymbol{x}},2\tau}(\boldsymbol{x}_j, T)|.$$

To evaluate the temporal convergence, we use a point cloud sufficiently sampled on the initial surface with a total number of points $N_{\boldsymbol{x}} = 30054$. Additionally, the initial condition is set to $s(\boldsymbol{x}, 0) = 0$. The experimental results are presented in Table 4.1.1. The temporal convergence rate of our method is consistent with the theoretical results corresponding to the BDF scheme.

Table 4.1.1: The time convergence test of the spatial point $\boldsymbol{x}$ and s with $T = 1.0$, $N_{\boldsymbol{x}} = 30054$ and $\eta = 100$.

	BDF1		BDF2		BDF3	
$1/\tau$	$\varepsilon^t_{\boldsymbol{x}}$	ord	$\varepsilon^t_{\boldsymbol{x}}$	ord	$\varepsilon^t_{\boldsymbol{x}}$	ord
160	1.47×10^{-4}	-	4.59×10^{-6}	-	6.25×10^{-8}	-
320	7.37×10^{-5}	1.00	1.16×10^{-6}	1.98	7.76×10^{-9}	3.01
640	3.68×10^{-5}	1.00	2.92×10^{-7}	1.99	9.70×10^{-10}	3.00
1280	1.84×10^{-5}	1.00	7.34×10^{-8}	2.00	1.20×10^{-10}	3.01
$1/\tau$	ε^t_{s}	ord	ε^t_{s}	ord	ε^t_{s}	ord
160	1.33×10^{-3}	-	4.03×10^{-5}	-	4.67×10^{-7}	-
320	6.68×10^{-4}	1.00	1.00×10^{-5}	2.00	5.86×10^{-8}	3.00
640	3.33×10^{-4}	1.00	2.52×10^{-6}	2.00	7.34×10^{-9}	3.00
1280	1.66×10^{-4}	1.00	6.31×10^{-7}	2.00	9.09×10^{-10}	3.01

Under this velocity, the surface $\Gamma(t)$ is a sphere centered at the origin with radius r. Considering only the radial component and neglecting tangential motion of points on the spherical surface, the radius $r(t)$ satisfies the following equation:

$$\frac{dr}{dt} = r(1-r).$$

Then the solution satisfying the initial value condition $r(0) = \frac{1}{2}$ is $r(t) = \frac{1}{1+e^{-t}}$. The error of radius at time $t = T$, as provided by the proposed method with step size τ and number of vertices $N_{\boldsymbol{x}}$, is measured by

$$\varepsilon_r^t(N_{\boldsymbol{x}}, \tau) = \max_{j=1,\ldots,N_{\boldsymbol{x}}} \left| \|\boldsymbol{X}_{N_{\boldsymbol{x}},\tau}(\boldsymbol{x}_j, T)\|_2 - r(T) \right|.$$

The experimental results are presented in Table 4.1.2. The temporal convergence of the radius is also consistent with the theoretical analysis of the BDF scheme.

Table 4.1.2: The time convergence test of radius r with $T = 1.0$, $N_{\boldsymbol{x}} = 30054$ and $\eta = 100$.

	BDF1		BDF2		BDF3	
$1/\tau$	ε_r^t	ord	ε_r^t	ord	ε_r^t	ord
80	294×10^{-4}	-	614×10^{-6}	-	7.13×10^{-8}	-
160	147×10^{-4}	1.00	155×10^{-6}	1.98	8.84×10^{-9}	3.01
320	737×10^{-5}	1.00	390×10^{-7}	1.99	1.08×10^{-9}	3.03
640	368×10^{-5}	1.00	979×10^{-8}	2.00	1.35×10^{-10}	2.99
1280	184×10^{-5}	1.00	245×10^{-8}	2.00	1.68×10^{-11}	3.01

4.2. Validation of spatial convergence

We evaluate the spatial error and convergence rate of the proposed method using a torus as the initial surface. The velocity field $\boldsymbol{v}(\boldsymbol{X}(\boldsymbol{x}, t), t)$ is defined as follows:

$$\boldsymbol{v}(\boldsymbol{X}(\boldsymbol{x}, t), t) = 500 \begin{pmatrix} \cos(\pi x)\sin(\pi y)\sin(\pi z) \\ \sin(\pi x)\cos(\pi y)\sin(\pi z) \\ \sin(\pi x)\sin(\pi y)\cos(\pi z) \end{pmatrix},$$

initial density value $\rho_0(\boldsymbol{x}) = 1$.

The error of the numerical solution at time $t = T$, as provided by the proposed method with step size τ, number of vertices $N_{\boldsymbol{x}}$, is measured by

$$\varepsilon_{\boldsymbol{x}}^s(N_{\boldsymbol{x}}, \tau) = \max_{j=1,\ldots,N_{\boldsymbol{x}}} |\boldsymbol{X}_{N_{\boldsymbol{x}},\tau}(\boldsymbol{x}_j, T) - \boldsymbol{X}_{4N_{\boldsymbol{x}},\tau}(\boldsymbol{x}_j, T)|,$$
$$\varepsilon_s^s(N_{\boldsymbol{x}}, \tau) = \max_{j=1,\ldots,N_{\boldsymbol{x}}} |S_{N_{\boldsymbol{x}},\tau}(\boldsymbol{x}_j, T) - S_{4N_{\boldsymbol{x}},\tau}(\boldsymbol{x}_j, T)|.$$

To evaluate spatial convergence, we employ a sufficiently small time step size $\tau = 1 \times 10^{-4}$ to ensure that the temporal discretization error is negligible.

Table 4.2.1: The spatial convergence test of the spatial point $\boldsymbol{x}$ and s with $T = 1.0$, $\tau = 1 \times 10^{-4}$ by Algorithm 1.

η		1			10			100		
$N_{\boldsymbol{x}}$	h	$\varepsilon^s_{\boldsymbol{x}}$	$ord_{N_{\boldsymbol{x}}}$	ord_h	$\varepsilon^s_{\boldsymbol{x}}$	$ord_{N_{\boldsymbol{x}}}$	ord_h	$\varepsilon^s_{\boldsymbol{x}}$	$ord_{N_{\boldsymbol{x}}}$	ord_h
256	0.450	7.77×10^{-5}	-	-	7.21×10^{-4}	-	-	4.27×10^{-3}	-	-
1024	0.237	5.06×10^{-5}	0.31	0.67	5.00×10^{-4}	0.26	0.57	2.98×10^{-3}	0.26	0.56
4096	0.120	1.62×10^{-5}	0.82	1.68	1.30×10^{-4}	0.97	1.98	5.47×10^{-4}	1.22	2.50
16384	0.060	5.14×10^{-6}	0.83	1.66	4.17×10^{-5}	0.82	1.65	1.56×10^{-4}	0.90	1.81
$N_{\boldsymbol{x}}$	h	ε^s_s	$ord_{N_{\boldsymbol{x}}}$	ord_h	ε^s_s	$ord_{N_{\boldsymbol{x}}}$	ord_h	ε^s_s	$ord_{N_{\boldsymbol{x}}}$	ord_h
256	0.450	6.45×10^{-2}	-	-	6.53×10^{-2}	-	-	6.70×10^{-2}	-	-
1024	0.237	1.97×10^{-2}	0.85	1.85	2.02×10^{-2}	0.85	1.83	1.81×10^{-2}	0.94	2.04
4096	0.120	4.37×10^{-3}	1.09	2.22	3.24×10^{-3}	1.32	2.70	2.37×10^{-3}	1.47	3.00
16384	0.060	1.11×10^{-3}	0.99	1.99	7.40×10^{-4}	1.06	2.14	6.61×10^{-4}	0.92	1.85

Table 4.2.1 presents the errors and convergence rate for the spatial variables $\boldsymbol{x}$ and s under various choices of η. As the number of spatial points increases, the errors consistently decrease. The estimated convergence rate with respect to the step size h is approximately twice that of the spatial rate in $\boldsymbol{x}$, aligning well with theoretical expectations. However, it is observed that the convergence rate with respect to h fluctuates around 2, which is reasonable taking into account that the unstructured points allocation may cause perturbation of errors in computations.

4.3. Validation of point distribution improvement

Spherical surface with simple velocity. We consider the same velocity field in the surface evolution as the example in Section 4.1, that is:

$$\boldsymbol{v}(\boldsymbol{X}(\boldsymbol{x},t),t) = \boldsymbol{X}(\boldsymbol{x},t)(1 - |\boldsymbol{X}(\boldsymbol{x},t)|).$$

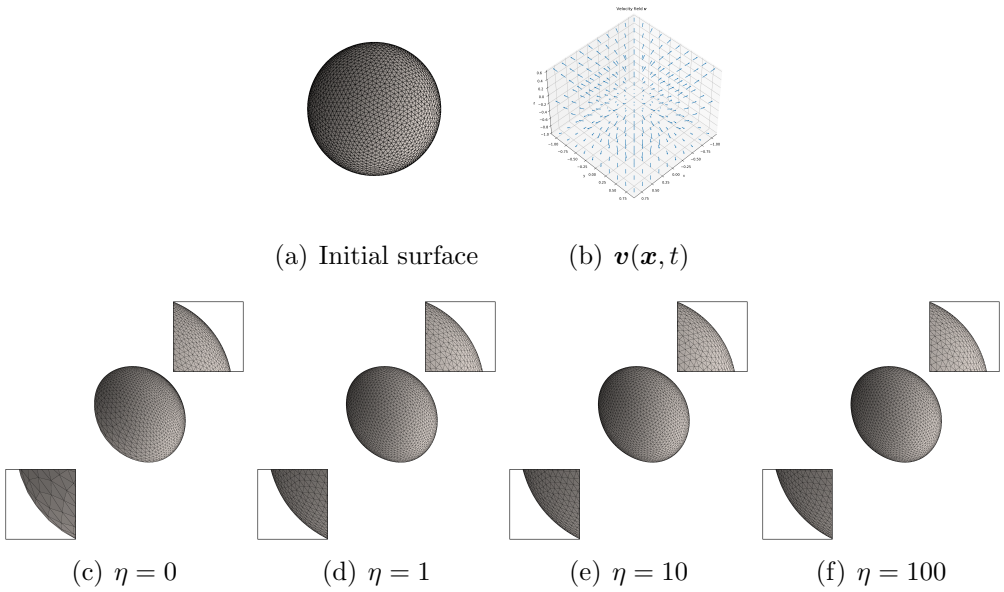
Visually it is displayed in Figure 4.3.1(b).

However, the initial surface is different to the one in Section 4.1. Here $\Gamma(t) = \{\boldsymbol{x} = (x,y,z) \in \mathbb{R}^3 : \varphi(\boldsymbol{x},t) = \frac{1}{2}\}$ is described by a level set function

$$\varphi(\boldsymbol{x},t) = (x - \frac{1}{4})^2 + (y - \frac{1}{4})^2 + (z - \frac{1}{4})^2.$$

It is a sphere with a radius of $r = 0.5$ and the center is shifted from the origin to $(0.25, 0.25, 0.25)$. Under such a configuration, due to the different magnitudes and directions of the velocities on both sides of the ball, there will be the phenomenon of point aggregation and sparsity without interaction to the surface evolution. We use the BDF2 scheme with $N_{\boldsymbol{x}} = 2904$ and $\tau = 1 \times 10^{-4}$ in our numerical tests.

Figure 4.3.1 illustrates the distribution of spatial points $\boldsymbol{x}$ under various values of $\eta = 0, 1, 10, 100$. In the cases of $\eta = 1, 10, 100$, the points distribute more evenly on the sphere, and the point clouds show better shapes of the sphere comparing to the case of $\eta = 0$, that is without the artificial tangential velocity introduced to the evolution. Furthermore, we



(a) Initial surface (b) $\boldsymbol{v}(\boldsymbol{x}, t)$

(c) $\eta = 0$ (d) $\eta = 1$ (e) $\eta = 10$ (f) $\eta = 100$

Figure 4.3.1: Surfaces at $T = 2$ computed by BDF2 method (Algorithm 1) with different η.

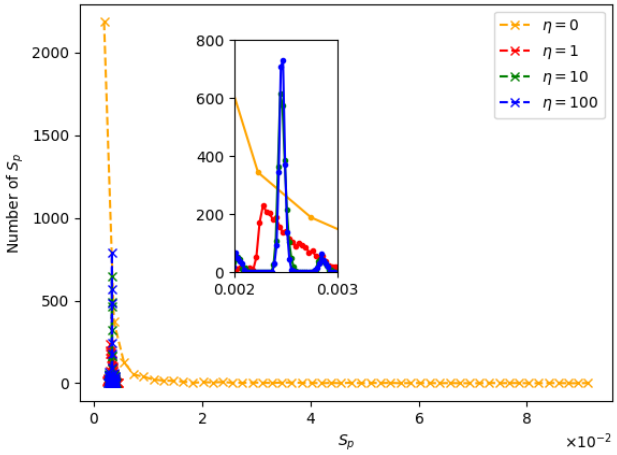


Figure 4.3.2: Distribution of area elements of each point, (25), for Algorithm 1 with different η. The surface is evolved up to time $T = 2$.

present the histogram of S_p in Figure 4.3.2. It shows that adding the tangential velocity is able to have more desirable distribution of S_p, and therefore a more uniform density of the point cloud.

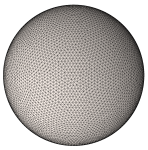
Spherical surface with complex velocity. We consider a scenario involving a complex velocity field coupled with a relatively simple surface evolution. The prescribed velocity field is defined as follows:

$$\boldsymbol{v}(\boldsymbol{X}(\boldsymbol{x},t),t) = -\frac{\xi_t \nabla \xi}{|\nabla \xi|^2}.$$

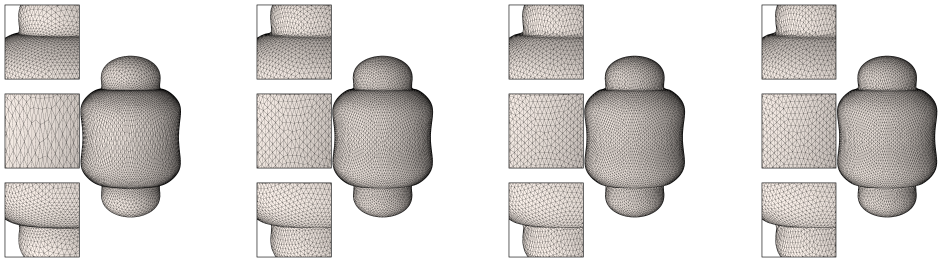
where $\xi(\boldsymbol{x},t) = \frac{x^2}{a^2(t)} + \frac{y^2}{a^2(t)} + G(\frac{z^2}{b^2(t)})$, $G(s) = 200s(s - \frac{199}{200})$, $a(t) = 0.1 + 0.05\sin 2\pi t$ and $b(t) = 1 + 0.2\sin 4\pi t$. Evolving surface $\Gamma(t) = \{\boldsymbol{x} = (x,y,z) \in \mathbb{R}^3 : \varphi(\boldsymbol{x},t) = 1\}$ is described by a level set function

$$\varphi(\boldsymbol{x},t) = x^2 + y^2 + z^2.$$

It is a unit sphere centered at the origin. If the tangential velocity component is omitted, spatial points exhibit significant clustering near the top and bottom regions by time $T = 0.6$, while points around the middle of the surface become sparsely distributed. These artifacts are clearly illustrated in Figure 4.3.3(b). We use again BDF2 scheme with $N_{\boldsymbol{x}} = 7446$ and $\tau = 1 \times 10^{-4}$ for surface evolution.



(a) Initial surface



(b) $\eta = 0$ (c) $\eta = 1$ (d) $\eta = 10$ (e) $\eta = 100$

Figure 4.3.3: Surfaces at $T = 0.6$ computed by BDF2 method (Algorithm 1) with different η.

In Figure 4.3.3, we present the results computed using Algorithm 1. The results indicate that Algorithm 1 effectively evolves the surface geometry while maintaining reasonable point distribution. Furthermore, we present the histogram of S_p in Figure 4.3.4, where similar performance to the previous experiment is reported.

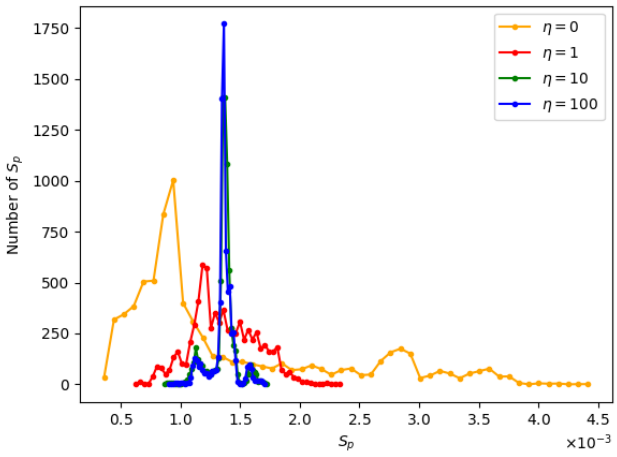


Figure 4.3.4: Distribution of area element of each point, (25), for Algorithm 1 with different η. Surface is evolved up to time $T = 0.6$.

Dumbbell-shaped surface with complex velocity. We consider an evolving surface $\Gamma(t) = \{\boldsymbol{x} = (x, y, z) \in \mathbb{R}^3 : \varphi(\boldsymbol{x}, t) = 1\}$ described by a level set function

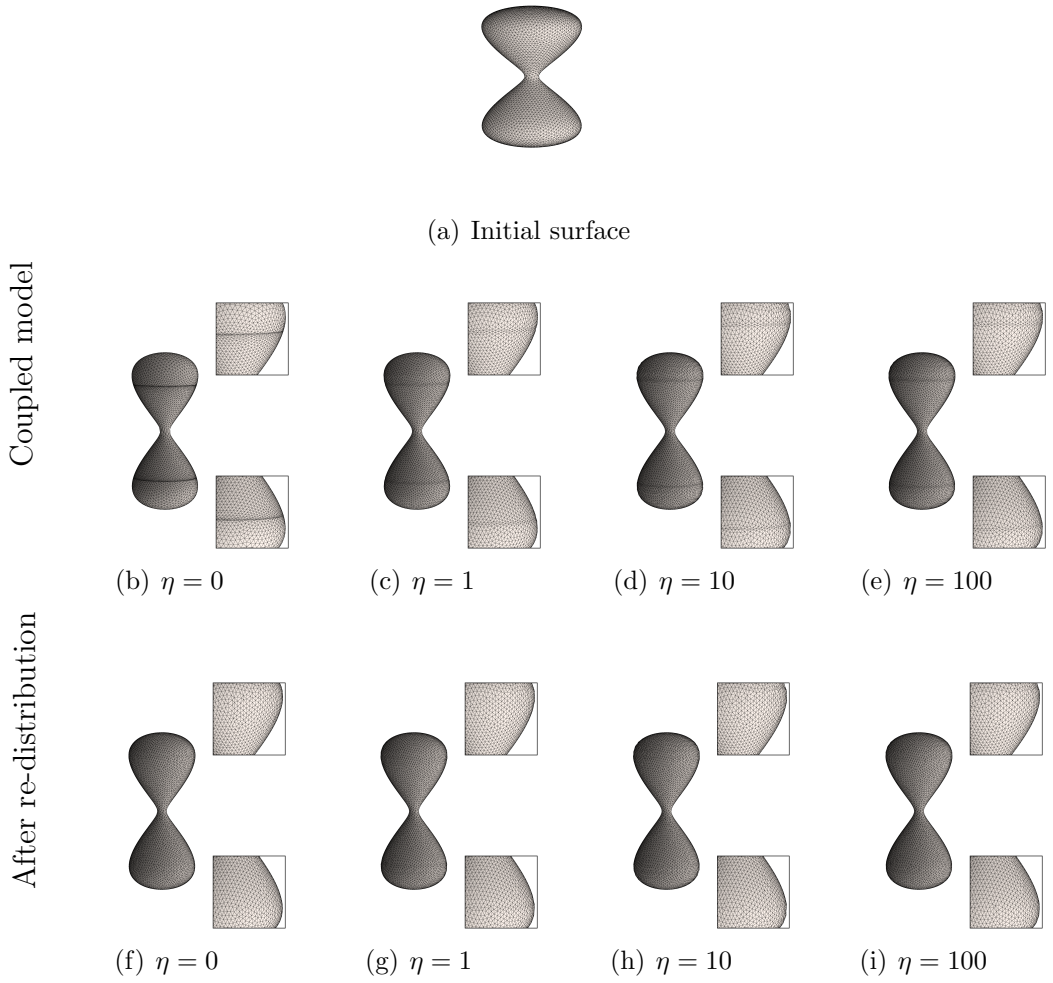
$$\varphi(\boldsymbol{x}, t) = \frac{x^2}{a^2(t)} + \frac{y^2}{a^2(t)} + G(\frac{z^2}{b^2(t)}),$$

where $G(s) = 200s(s - \frac{199}{200})$, $a(t) = 0.1 + 0.05 \sin 2\pi t$ and $b(t) = 1 + 0.2 \sin 4\pi t$. The surface evolves under the following velocity field:

$$\boldsymbol{v}(\boldsymbol{X}(\boldsymbol{x}, t), t) = -\frac{\varphi_t \nabla \varphi}{|\nabla \varphi|^2}.$$

This example had been considered in [37] for surface finite element methods, where tangential velocity method was used to improve mesh quality. When the evolution time $T \geq 0.6$, the vertices are clustering and aligned along narrow bands, leading to anisotropic point distribution. Such degeneration is a bottleneck for long-term simulations and compromises numerical stability. This phenomenon has been observed in our point cloud algorithm (e.g. Algorithm 1) as well, as clearly illustrated in Figure 4.3.5, where the point accumulation is visually evident. We use BDF2 scheme with $N_{\boldsymbol{x}} = 5606$ and $\tau = 1 \times 10^{-4}$ for surface evolution.

Similarly, in the first line of Figure 4.3.5, we present the results obtained using Algorithm 1, while the second line shows the improved results after applying Algorithm 3. In this example, it is evident that due to the complexity of the prescribed velocity field, by adjusting the scalar parameter η alone appears insufficient to fully prevent point aggregation. Though, it does play an important role in mitigating the clustering effect. As expected, the application of Algorithm 3 successfully achieves the desired spatial redistribution. In addition, similar results are also presented in Figure 4.3.6, where we see that Algorithm 3 makes the distribution of points more uniform.

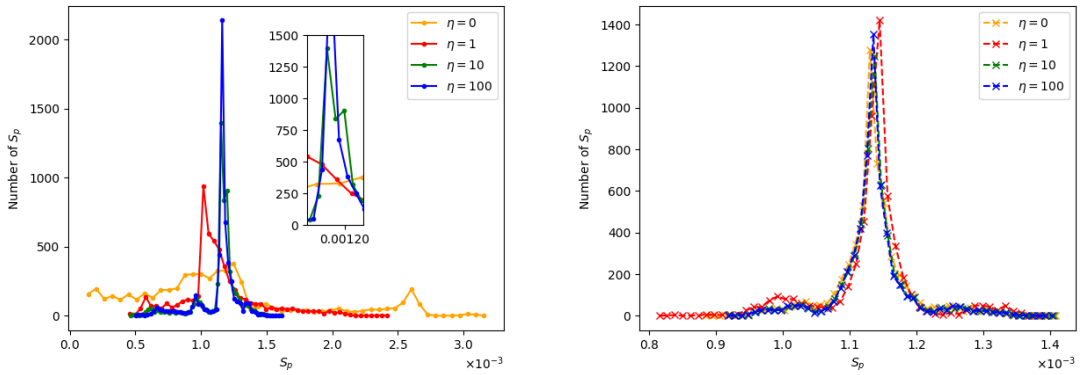


(a) Initial surface

(b) $\eta = 0$ (c) $\eta = 1$ (d) $\eta = 10$ (e) $\eta = 100$

(f) $\eta = 0$ (g) $\eta = 1$ (h) $\eta = 10$ (i) $\eta = 100$

Figure 4.3.5: Surfaces at $T = 0.6$ computed by coupled model (Algorithm 1) and re-distribution method (Algorithm 3) with different η.



(a) Distribution of area elements with different η (b) Distribution of area elements after point redistribution

Figure 4.3.6: Distribution of area element of each point, (25), for Algorithm 1 and Algorithm 3, with different η. Surface is evolved up to time $T = 0.6$.

Nonuniform distribution on surfaces. In practical applications, there might be a need for a non-uniform distribution of points, with larger density in regions of particular interest. For instance, the areas exhibit high curvature on the surface. To accommodate such requirements, we use Algorithm 4 to have targeted point redistribution. Specifically, we introduce a prescribed target distribution $p(\boldsymbol{x})$ to guide the evolution of the density function. For simplicity, we define the target distributions $p(\boldsymbol{x})$ for both the ellipsoid surface and the dumbbell-shaped surface as follows:

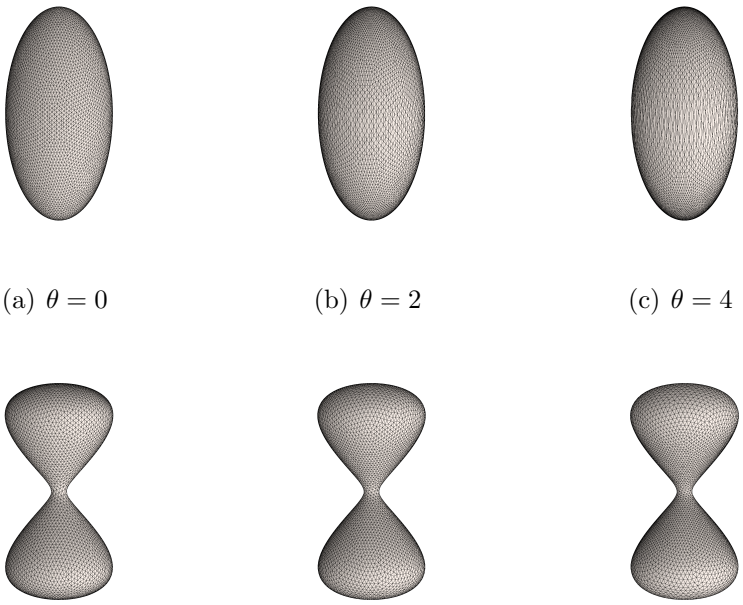
$$p_e(\boldsymbol{x}) \propto \exp\left(\theta(\sin z + 1)\right),$$
$$p_d(\boldsymbol{x}) \propto \exp\left(\theta(\cos z + 1)\right).$$

From Figure 4.3.7, it is evident that Algorithm 4 effectively redistributes points in accordance with the prescribed target distributions. On the ellipsoidal surface, the points are concentrated toward both polar points, aligning with regions of higher target density. In contrast, for the dumbbell-shaped surface, the points accumulate predominantly in the central region. Both examples demonstrate the algorithm's capability of adaptively concentrating points as it is designed for. These results confirm that Algorithm 4 can flexibly accommodate spatially varying distribution requirements in complex geometries.

4.4. Mean curvature flow

In this section, we apply our method to the renowned mean curvature flow (MCF) [38]. The MCF describes the geometric motion along the normal field scaled with the scaler mean curvature at each point of a surface as it evolves over time. This means the velocity satisfies the following equation:

$$\boldsymbol{v}(\boldsymbol{X}(\boldsymbol{x}, t), t) = -H\boldsymbol{n},$$



(a) $\theta = 0$ (b) $\theta = 2$ (c) $\theta = 4$

(d) $\theta = 0$ (e) $\theta = 2$ (f) $\theta = 4$

Figure 4.3.7: Results of point redistribution with assigned target distribution $p(\boldsymbol{x})$ (Algorithm 4).

where H represents the scalar mean curvature ($H\boldsymbol{n} = -\Delta_\Gamma \boldsymbol{X}$) and $\boldsymbol{n}$ is the outward unit normal vector field. Then the problem (8) can be changed to

$$\begin{aligned}
\frac{d}{dt}\boldsymbol{X}(\boldsymbol{x},t) &= \boldsymbol{v}(\boldsymbol{X}(\boldsymbol{x},t),t) - \eta\nabla_{\Gamma(t)}s(\boldsymbol{X}(\boldsymbol{x},t),t), && \boldsymbol{x}\in\Gamma_0, \quad t\geq 0,\\
\boldsymbol{v}(\boldsymbol{X}(\boldsymbol{x},t),t) &= \Delta_{\Gamma(t)}\boldsymbol{X}(\boldsymbol{x},t), && \boldsymbol{x}\in\Gamma_0, \quad t\geq 0,\\
\frac{d}{dt}s(\boldsymbol{X}(\boldsymbol{x},t),t) &= \eta\Delta_{\Gamma(t)}s(\boldsymbol{X}(\boldsymbol{x},t),t) - \mathrm{div}_{\Gamma(t)}\boldsymbol{v}(\boldsymbol{X}(\boldsymbol{x},t),t), && \boldsymbol{x}\in\Gamma_0, \quad t\geq 0,\\
\boldsymbol{X}(\boldsymbol{x},0) &= \boldsymbol{x}, \quad s(\boldsymbol{x},0) = \log\rho_0(\boldsymbol{x}), && \boldsymbol{x}\in\Gamma_0.
\end{aligned}$$

Using the discrete and notation in Section 3, we obtain the following fully discrete scheme

$$\begin{aligned}
\frac{a}{\tau}\boldsymbol{X}_k - \boldsymbol{v}_k + \eta M_G^T(\bar{\boldsymbol{X}}_k)s_k &= \frac{1}{\tau}\hat{\boldsymbol{X}}_{k-1},\\
M_L(\bar{\boldsymbol{X}}_k)\boldsymbol{X}_k - \boldsymbol{v}_k &= 0,\\
M_G(\bar{\boldsymbol{X}}_k)\boldsymbol{v}_k + \frac{a}{\tau}s_k - \eta M_L(\bar{\boldsymbol{X}}_k)s_k &= \frac{1}{\tau}\hat{s}_{k-1},\\
\boldsymbol{X}(\boldsymbol{x},0) = \boldsymbol{x}, \quad s(\boldsymbol{x},0) &= \log\rho_0(\boldsymbol{x}).
\end{aligned} \tag{26}$$

Then, by solving (26), we obtain the discrete trajectory for the MCF.

We begin with a specific example. The parameterization of the two-dimensional surface under consideration is as follows:

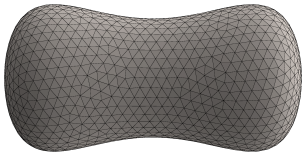
$$\boldsymbol{x}(\theta,\phi) = \begin{pmatrix} \cos\phi \\ (\frac{3}{5}\cos^2\phi + \frac{2}{5})\cos\theta\sin\phi \\ (\frac{3}{5}\cos^2\phi + \frac{2}{5})\sin\theta\sin\phi \end{pmatrix}, \quad \theta\in[0,2\pi), \quad \phi\in[0,\pi], \tag{27}$$

which is a benchmark example proposed in [25]. It is well known that without incorporating tangential velocity, the approximation of this problem typically faces mesh distortion and node clustering, leading to numerical breakdown before the surface evolving into a sphere. This typically necessitates mesh redistribution techniques, as discussed in [39]. Therefore, the using of extra tangential velocity is a cure for numerical treatment. Several mesh-based approaches have been proposed to enhance the mesh quality by introducing artificial tangential motion. Notably, the BGN algorithm in [15, 16, 16] and DeTurck's reparameterization technique in [25] are those representatives. These methods rely on an appropriate initial parameterization, such as that in (27), and are capable of evolving the surface smoothly into a sphere.

In the following, we denote by E_h the surface area of the numerically computed surface. We use the rescaling trick to maintain the area of $\boldsymbol{X}(t)$ in a reasonable range during the computation.

$$\boldsymbol{X}^* = \lambda\boldsymbol{X}, \quad s^* = s, \quad t^* = \lambda^2 t,$$

where λ is a scaling parameter, which is adjusted adaptively in the computation. We use the BDF2 scheme, $N_{\boldsymbol{x}} = 5606$ and $\tau = 1\times 10^{-3}$ as initial value for the surface evolution. Moreover, during the simulation, point redistribution (Algorithm 3) is invoked every 100



(a) Initial surface

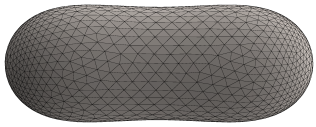
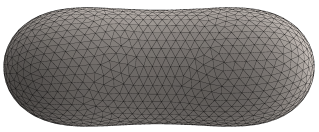
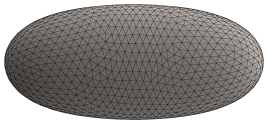
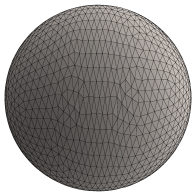
	$t = 0.05$	$t = 0.087625$	$t = 0.0896913948$
$\eta = 0$			
	(b) $E_h = 3.91E-4$	(c) $E_h = 3.67E-10$	(d) none
$\eta = 100$			
	(e) $E_h = 8.98E-4$	(f) $E_h = 2.94E-5$	(g) $E_h = 8.18E-39$

Figure 4.4.1: Algorithm 3 evolves the MCF problem (images are rescaled). (d):Without adding the artificial tangential velocity, the numerical method becomes unfeasible gradually.

time steps. This periodic correction ensures that the point cloud remains well-distributed throughout the evolution process, effectively preventing excessive concentration in localized regions.

In the first line of Figure 4.4.1, we present the results of surface evolution without applying tangential velocity correction. It is evident that at time $t = 0.087625$, the point cloud has accumulated significantly at both ends of the ellipsoid. At this stage, further evolution becomes numerically infeasible due to the severe point clustering. In contrast, the second line shows the results computed using Algorithm 3. At the same time step (Figure 4.4.1(f)), the point cloud remains uniformly distributed, allowing the surface to continue evolving. As time progresses, the surface clearly converges toward a spherical shape—visually enhanced by scaling the radius. At $t = 0.0896913948$, $E_h = 8.18 \times 10^{-39}$ indicate that the surface has nearly collapsed to a single point, as one expects for mean curvature flow (Figure 4.4.1(g)). These results clearly demonstrate the effectiveness of our algorithm while accurately capturing MCF dynamics on point clouds.

5. Conclusion

In this paper, we proposed a novel artificial tangential velocity method for the evolution of surfaces represented by point clouds. To address the issue of unexpected point clustering during surface evolution, we introduced a surface density field and derived a Fokker–Planck equation to govern its evolution. Then we formulated an evolutionary flow with an artificial tangential velocity coupled with the Fokker–Planck equation to dynamically guide the distribution of points. Redistribution and target distribution have been introduced in the case when extra configurations are needed in the algorithms. Extensive numerical experiments demonstrated the robustness and effectiveness of our method across a variety of scenarios. Overall, the proposed method and the idea provides foundational approach for accurate and stable simulation of surface evolution on point clouds. Hopefully, this approach is also possible to tackle more complex surface evolution problems.

It is worth noting that the trigger of the redistribution algorithm is pulled empirically in the current work. A more intelligent switch would be desirable, as well there are many theoretical questions open for future investigations.

Acknowledgement

The work of JP and ZS was supported by the National Natural Science Foundation of China (NSFC) No. 92370125. The work of GD was supported by NSFC No. 12471402. The work of HG was partially supported by the Andrew Sisson Fund and the Faculty Science Researcher Development Grant of The University of Melbourne.

References

[1] E. Bänsch, P. Morin, R. H. Nochetto, A finite element method for surface diffusion: the parametric case, Journal of Computational Physics 203 (1) (2005) 321–343.

[2] E. Marchandise, C. C. de Wiart, W. Vos, C. Geuzaine, J.-F. Remacle, High-quality surface remeshing using harmonic maps—Part II: Surfaces with high genus and of large aspect ratio, International Journal for Numerical Methods in Engineering 86 (11) (2011) 1303–1321.

[3] J.-F. Remacle, C. Geuzaine, G. Compere, E. Marchandise, High-quality surface remeshing using harmonic maps, International Journal for Numerical Methods in Engineering 83 (4) (2010) 403–425.

[4] G. Dziuk, An algorithm for evolutionary surfaces, Numerische Mathematik 58 (1) (1990) 603–611.

[5] A. Bonito, R. H. Nochetto, M. S. Pauletti, Parametric FEM for geometric biomembranes, Journal of Computational Physics 229 (9) (2010) 3171–3188.

[6] G. Dziuk, Computational parametric Willmore flow, Numerische Mathematik 111 (2008) 55–80.

[7] K. C. Cheung, L. Ling, S. J. Ruuth, A localized meshless method for diffusion on folded surfaces, Journal of Computational Physics 297 (2015) 194–206.

[8] K. Deckelnick, V. Styles, Stability and error analysis for a diffuse interface approach to an advection-diffusion equation on a moving surface, Numerische mathematik 139 (2018) 709–741.

[9] G. Dziuk, C. M. Elliott, Finite elements on evolving surfaces, IMA Journal of numerical analysis 27 (2) (2007) 262–292.

[10] G. Dziuk, C. M. Elliott, A fully discrete evolving surface finite element method, SIAM Journal on Numerical Analysis 50 (5) (2012) 2677–2694.

[11] C. Lehrenfeld, M. A. Olshanskii, X. Xu, A stabilized trace finite element method for partial differential equations on evolving surfaces, SIAM Journal on Numerical Analysis 56 (3) (2018) 1643–1672.

[12] Y. Li, X. Qi, J. Kim, Direct discretization method for the Cahn–Hilliard equation on an evolving surface, Journal of Scientific Computing 77 (2018) 1147–1163.

[13] A. Petras, L. Ling, C. Piret, S. J. Ruuth, A least-squares implicit RBF-FD closest point method and applications to PDEs on moving surfaces, Journal of Computational Physics 381 (2019) 146–161.

[14] J. W. Barrett, H. Garcke, R. Nürnberg, Parametric finite element approximations of curvature-driven interface evolutions, in: Handbook of numerical analysis, Vol. 21, Elsevier, 2020, pp. 275–423.

[15] J. W. Barrett, H. Garcke, R. Nürnberg, A parametric finite element method for fourth order geometric evolution equations, Journal of Computational Physics 222 (1) (2007) 441–467.

[16] J. W. Barrett, H. Garcke, R. Nürnberg, On the parametric finite element approximation of evolving hypersurfaces in $\mathbb{R}^3$, Journal of Computational Physics 227 (9) (2008) 4281–4307.

[17] J. W. Barrett, H. Garcke, R. Nürnberg, Parametric approximation of Willmore flow and related geometric evolution equations, SIAM Journal on Scientific Computing 31 (1) (2008) 225–253.

[18] W. Bao, H. Garcke, R. Nürnberg, Q. Zhao, Volume-preserving parametric finite element methods for axisymmetric geometric evolution equations, Journal of Computational Physics 460 (2022) 111180.

[19] W. Bao, Q. Zhao, A structure-preserving parametric finite element method for surface diffusion, SIAM Journal on Numerical Analysis 59 (5) (2021) 2775–2799.

[20] Q. Zhao, W. Jiang, W. Bao, An energy-stable parametric finite element method for simulating solid-state dewetting, IMA Journal of Numerical Analysis 41 (3) (2021) 2026–2055.

[21] J. W. Barrett, H. Garcke, R. Nürnberg, Eliminating spurious velocities with a stable approximation of viscous incompressible two-phase Stokes flow, Computer Methods in Applied Mechanics and Engineering 267 (2013) 511–530.

[22] J. W. Barrett, H. Garcke, R. Nürnberg, A stable parametric finite element discretization of two-phase Navier-Stokes flow, Journal of Scientific Computing 63 (2015) 78–117.

[23] G. Fu, Arbitrary Lagrangian-Eulerian hybridizable discontinuous galerkin methods for incompressible flow with moving boundaries and interfaces, Computer Methods in Applied Mechanics and Engineering 367 (2020) 113158.

[24] S. Ganesan, A. Hahn, K. Simon, L. Tobiska, ALE-FEM for two-phase and free surface flows with surfactants, Transport Processes at Fluidic Interfaces (2017) 5–31.

[25] C. M. Elliott, H. Fritz, On approximations of the curve shortening flow and of the mean curvature flow based on the deturck trick, IMA Journal of Numerical Analysis 37 (2) (2017) 543–603.

[26] J. Hu, B. Li, Evolving finite element methods with an artificial tangential velocity for mean curvature flow and Willmore flow, Numerische Mathematik 152 (1) (2022) 127–181.

[27] B. Duan, B. Li, New artificial tangential motions for parametric finite element approximation of surface evolution, SIAM Journal on Scientific Computing 46 (1) (2024) A587–A608.

[28] B. Li, Convergence of Dziuk's semidiscrete finite element method for mean curvature flow of closed surfaces with high-order finite elements, SIAM Journal on Numerical Analysis 59 (3) (2021) 1592–1617.

[29] G. Bai, J. Hu, B. Li, A convergent evolving finite element method with artificial tangential motion for surface evolution under a prescribed velocity field, SIAM Journal on Numerical Analysis 62 (5) (2024) 2172–2195.

[30] G. Bai, B. Li, A new approach to the analysis of parametric finite element approximations to mean curvature flow, Foundations of Computational Mathematics 24 (5) (2024) 1673–1737.

[31] G. Bai, B. Li, Convergence of a stabilized parametric finite element method of the Barrett-Garcke-Nürnberg type for curve shortening flow, Mathematics of Computation (2024).

[32] H. Risken, T. Caugheyz, The Fokker-Planck equation: Methods of solution and application, Journal of Applied Mechanics 58 (3) (1991) 860.

[33] M. P. do Carmo, Differential geometry of curves and surfaces, Englewood Cliffs, New Jersey (1976).

[34] J. Liang, H. Zhao, Solving partial differential equations on point clouds, SIAM Journal on Scientific Computing 35 (3) (2013) A1461–A1486.

[35] J. B. Rosser, A Runge-Kutta for all seasons, SIAM Review 9 (3) (1967) 417–452.

[36] R. Lai, J. Liang, H. Zhao, A local mesh method for solving PDEs on point clouds., Inverse Problems & Imaging 7 (3) (2013).

[37] C. M. Elliott, V. Styles, An ALE ESFEM for solving PDEs on evolving surfaces, Milan Journal of Mathematics 80 (2) (2012) 469–501.

[38] C. Mantegazza, Lecture Notes on Mean Curvature Flow, Vol. 290, Springer Science & Business Media, 2011.

[39] B. Duan, B. Li, Z. Zhang, High-order fully discrete energy diminishing evolving surface finite element methods for a class of geometric curvature flows, Annals of Applied Mathematics 37 (4) (2021) 405–436.