Kernel-based collocation methods for heat transport on evolving surfaces

Meng Chen[a], Leevan Ling[b]

[a]*Department of Mathematics, Nanchang University, Nanchang, China*
[b]*Department of Mathematics, Hong Kong Baptist University, Kowloon Tong, Hong Kong*

Abstract

We propose algorithms for solving convective-diffusion partial differential equations (PDEs), which model surfactant concentration and heat transport on evolving surfaces, based on extrinsic kernel-based meshless collocation methods. The algorithms can be classified into two categories: one collocates PDEs extrinsically and analytically, and the other approximates surface differential operators by meshless pseudospectral approaches. The former is specifically designed to handle PDEs on evolving surfaces defined by parametric equations, and the latter works on surface evolutions based on point clouds. After some convergence studies and comparisons, we demonstrate that the proposed method can solve challenging PDEs posed on surfaces with high curvatures with discontinuous initial conditions with correct physics.

Keywords: Kansa methods, radial basis functions, point clouds, convective-diffusion equations, mass conservation, overdetermined formulations.

1. Introduction

Heat transport on moving surfaces are modelled by convection-diffusion partial differential equations (PDEs) that frequently appear in physical and biological fields. Applications include dealloying by surface dissolution [1], pattern formation on evolving biological surfaces [2], modelling geometric

⋆This work was supported by a Hong Kong Research Grant Council GRF Grant.
Email addresses: chenmeng821@sina.com (Meng Chen), lling@hkbu.edu.hk (Leevan Ling)

biomembranes [3], and cell motility and chemotaxis [4]. The formal problem statement is as follows. Let $\mathcal{S}(t) \subset \mathbb{R}^d$ be a continuously evolving surface with codimension one; for each $t \in [0, T]$, we also assume that the surface is smooth, closed, connected, and complete. Furthermore, let $\boldsymbol{n} = \boldsymbol{n}(\cdot, t)$ and $\boldsymbol{\tau} = \boldsymbol{\tau}(\cdot, t)$ represent the unit normal and the tangent vector of $\mathcal{S}(t)$.

We consider convective-diffusion equations defined on $\mathcal{S}(t)$ in the form of

$$\partial_t u + \boldsymbol{v} \cdot \nabla u + (\nabla_{\mathcal{S}} \cdot \boldsymbol{v})u - \varepsilon \Delta_{\mathcal{S}} u = f \qquad \text{in } \mathcal{S}(t) \times [0, T], \tag{1}$$

subject to homogeneous Dirichlet initial conditions

$$u(\cdot, 0) = u_0 \quad \text{on } \mathcal{S}(0),$$

where $\varepsilon \geq 0$ is the diffusive coefficient, and $\boldsymbol{v} := \boldsymbol{v}_n + \boldsymbol{v}_\tau$ is the velocity of surface motion whose normal and tangent components are denoted by $\boldsymbol{v}_n$ and $\boldsymbol{v}_\tau$ respectively. The surface differential operators in (1) are defined as

$$\nabla_{\mathcal{S}} := (I_d - \boldsymbol{n}\boldsymbol{n}^T)\nabla$$

and

$$\Delta_{\mathcal{S}} := \nabla_{\mathcal{S}} \cdot \nabla_{\mathcal{S}},$$

where I_d is the d-dimensional identity matrix and ∇ is the standard gradient operator in Cartesian coordinates.

The problem we consider includes several physical models in the literature. In the absence of diffusive flux and source, i.e., $\varepsilon = 0$ and $f = 0$, the viscosity solution u to (1) as $\varepsilon \to 0^+$ satisfies a mass conservation law [5], namely

$$\frac{\mathrm{d}}{\mathrm{d}t} \int_{\mathcal{S}(t)} u \, \mathrm{d}\mathcal{S} = 0. \tag{2}$$

Readers can find more details in [2, 6, 7, 8].

To solve the target PDE (1) on the evolving surface, a surface finite element method (SFEM) was proposed with the use of weak and variational formulas based on triangular meshes extrinsically defined on surfaces in Cartesian coordinates [6, 9]. Embedding techniques for static problems can also be extended to deal with surface PDEs posed in higher-dimensional domains [10, 11, 12, 13, 14]. Some domain-type time-dependent FEMs have been implemented in narrow bands containing moving surfaces [7, 8, 15, 16]. Implicit time-stepping schemes have also been used for temporal discretization in the

FEM literature mentioned above, except in [8] where weak forms were utilized in both space and time. In this paper, we apply kernel-based meshless collocation techniques to solve (1). The proposed method is extrinsic, which means that no embedding domain out of the surface is required. One variant of our method naturally works with point clouds, which makes it capable to handle solution-driven surface evolutions.

The rest of this paper is organized as follows. Firstly, we deal with the case where the evolutions of surfaces are prescribed by some parameterizations that are known a prior. Section 2 contains a novel algorithm for this case. Next, we focus on cases where the surfaces are defined by some point clouds. In Section 3, we propose a second algorithm that does not require analytic information about normals and velocities of surface motions. In Section 4, we first test the accuracy and convergence of our first algorithm. Next, we test the proposed algorithms against mass balance in the case of (2). Lastly, as a pilot study, we examine the robustness of our method in dealing with merging surfaces, which involves surfaces with high curvatures. We conclude the results and observations in Section 5.

2. A collocation method for PDEs posed on parameterized surfaces with prescribed evolution

To begin, we consider the case that the surface $\mathcal{S}(t)$ and its motion are both given by some time-dependent parameterization. Assume the evolving surface is defined as

$$\mathcal{S}(t) := \Big\{ \boldsymbol{x}(\boldsymbol{\varphi}, t) \in \mathbb{R}^d, \text{ for } t \in [0, T], \ \boldsymbol{\varphi} \in \mathcal{B} \subset \mathbb{R}^{d-1} \Big\}, \tag{3}$$

where $\boldsymbol{\varphi} \in \mathcal{B}$ are parameters in certain parameter space that define surface points $\boldsymbol{x}(\boldsymbol{\varphi}, t) = \big[x_i(\boldsymbol{\varphi}, t)\big]_{i=1}^{d}$ for any time $t \in [0, T]$. The following example aims to clarify our notations.

Example:

Consider an evolving circle with changing radius. Using polar coordinates, we can pick $\varphi = \boldsymbol{\varphi} \in \mathcal{B} = [0, 2\pi] \subset \mathbb{R}$ and for some function $r(t) > 0$ to define surface points on $\mathcal{S}(t)$ at any time $t \geq 0$ by

$$\boldsymbol{x}(\varphi, t) = [x_1(\varphi, t), x_2(\varphi, t)] = [r(t)\cos\varphi,\ r(t)\sin\varphi] \in \mathcal{S}(t) \subset \mathbb{R}^2.$$

Obviously, the velocity for this surface motion is given by the partial derivative of $\boldsymbol{x}$ with respect to t. $\square$

In our notations, we have velocity vector in (1) given by

$$\boldsymbol{v}(\boldsymbol{\varphi}, t) = \frac{\partial}{\partial t}\boldsymbol{x}(\boldsymbol{\varphi}, t) = \left[\partial_t x_i(\boldsymbol{\varphi}, t)\right]_{i=1}^{d}. \tag{4}$$

With all required analytic information about the evolving surface ready, we can start discretizing (1) first in time and then in space.

Firstly, we semi-discretize (1) based on some partition $\{t_m\}_{m=0}^{N}$ of the interval $[0, T]$. For simplicity, we assume the partition is equispaced with time stepping size $\triangle t$ and we will employ the standard θ-scheme. Note that the problem (1) in hand is given in the form of $\partial_t u = F(u)$, which is *not* yet ready for applying the θ-scheme. Instead, we implicitly discretize the material derivatives [7, 15] of the solution to (1) as follows.

We abbreviate $\mathcal{S}(t^m) =: \mathcal{S}^m$ the snapshot of our evolving surfaces at time t_m for $m = 1, \ldots, T/\triangle t$, on which we have surface points

$$\boldsymbol{x}^m = \boldsymbol{x}(\boldsymbol{\varphi}, t^m) \subset \mathcal{S}^m, \quad \boldsymbol{\varphi} \in \mathcal{B}.$$

Let us also denote the solution u and right-hand function f at t_m as

$$u^m := u(\boldsymbol{x}^m, t^m) = u\Big(\boldsymbol{x}(\boldsymbol{\varphi}, t^m), t^m\Big) \quad \text{and} \quad f^m := f(t_m).$$

We approximate the material derivative of u and obtain

$$\begin{aligned} \frac{\partial}{\partial t}u + \boldsymbol{v} \cdot \nabla u &= \frac{\mathrm{d}}{\mathrm{d}t}u\Big(\boldsymbol{x}(\boldsymbol{\varphi}, t), t\Big) \\ &\approx \frac{u(\boldsymbol{x}^m, t^m) - u(\boldsymbol{x}^{m-1}, t^{m-1})}{\triangle t} = \frac{u^m - u^{m-1}}{\triangle t}. \end{aligned}$$

Thus, the time discretization of (1) by θ-method is given as

$$\frac{u^m - u^{m-1}}{\triangle t} + (1-\theta)\mathcal{A}^m u^m + \theta \mathcal{A}^{m-1} u^{m-1} = (1-\theta) f^m + \theta f^{m-1}, \tag{5}$$

where the surface elliptic operator $\mathcal{A}^m$ is defined as

$$\mathcal{A}^m := (\nabla_{\mathcal{S}^m} \cdot \boldsymbol{v}) - \varepsilon \Delta_{\mathcal{S}^m} \tag{6}$$

on the surface $\mathcal{S}^m$ and the velocity $\boldsymbol{v} = \boldsymbol{v}(\,\cdot\,, t^m)$ is to be evaluated exactly. We can further simplify (5) to isolate the unknown solution u^m and get a time-independent second-order surface PDE:

$$\mathcal{L}^m u^m := \Big(1 + (1-\theta)\triangle t \mathcal{A}^m\Big) u^m = g^m, \tag{7}$$

where the right-hand function g^m, which depends on the known functions u^{m-1} and f, is known and given by

$$g^m := \Big(1 - \theta\triangle t \mathcal{A}^{m-1}\Big) u^{m-1} + \triangle t\Big((1-\theta) f^m + \theta f^{m-1}\Big). \tag{8}$$

Aiming towards fully discretized problems, we employ our previously proposed overdetermined Kansa method [17] to solve surface PDEs. We set up an overdetermined kernel-based collocation method [18] to solve surface PDE $\mathcal{L}^m u^m = g^m$ on surfaces $\mathcal{S}^m$ and $\mathcal{S}^{m-1}$ (due to $\mathcal{L}^m$ and g^m) for solution u^m.

Based on the theories in [19], we can take some commonly used symmetric positive definite (SPD) kernel and restrict it to any one of the surfaces $\mathcal{S}^m$ to obtain an SPD surface kernel

$$\Psi(\,\cdot\,,\,\cdot\,) : \mathcal{S}^m \times \mathcal{S}^m \to \mathbb{R}$$

with a certain smoothness order $\mu > (d-1)/2$. Let $\varphi_Z \subset \mathcal{B}$ be a set of n_Z points in the parameter space; similarly, let $\varphi_X \subset \mathcal{B}$ be another set of $n_X > n_Z$ points. For each m, let $Z^m = \boldsymbol{x}(\varphi_Z, t^m)$ and $X^m = \boldsymbol{x}(\varphi_X, t^m)$ be the set of trial centers and collocation points. We shall use denser collocation points because of available convergence theories for collocation methods in domains [18] and on surfaces [12]. In both works, they require X to be sufficiently denser with respect to Z resulting in *overdetermined formulas*, in order to establish stability estimates[1].

Trial functions at time t^m are expressed by a linear combination

$$u^m = \Psi(\,\cdot\,, Z^m)\boldsymbol{\lambda}_Z^m := \sum_{z_i \in Z^m} \lambda_i^m \Psi(\,\cdot\,, z_i), \tag{9}$$

where $\boldsymbol{\lambda}_Z^m := \{\lambda_i^m\}_{i=1}^{n_Z}$ is the unknown coefficient vector to be determined. As in the original Kansa method, we collocate (7) and (8) at set X^m to yield an $n_X \times n_Z$ matrix system

$$[\mathcal{L}^m \Psi(X^m, Z^m)]\boldsymbol{\lambda}_Z^m = g^m_{|X}. \tag{10}$$

[1]In this context, stability means that errors can be bounded in continuous norms with discrete residuals.

In the Cartesian counterpart, overdetermined formulations guarantee stability as mentioned before, which ultimately leads to error estimates [12, 18].

Given the parametric equation (3), the surface differential operator in (6) can be rewritten without any implicit dependency on the surface. Thus, $\mathcal{L}^m\Psi : \mathcal{S}^m \times \mathcal{S}^m \to \mathbb{R}$ can also be analytically evaluated for each $m = 1, \ldots, T/\triangle t$. The right hand vector $g^m_{|X} = g^m(X^m)$ is the nodal value of (8) evaluated at X^m. The overdetermined system (10) should be solved in the least-squares sense. We summarize with Algorithm 1.

Algorithm 1 : For solving PDE (1) posed on a parametrized surface (3)

Initialization :
  set $t = t^0$, *select* $0 \le \theta \le 1$;
  define $\varphi_Z, \varphi_X \subset \mathcal{B}$, *compute* $Z^0, X^0 \subset \mathcal{S}^0$;
  interpolate u_0 *at* Z^0 *to obtain* $\boldsymbol{\lambda}^0_Z$;
  symbolically derive analytic formulas of $\mathcal{A}^m$ *in* (6);
  $m = 1$;
While $(t \le T)$
  compute $\{X^m, Z^m\} \subset \mathcal{S}^m$;
  assemble and solve (10) *in the least-squares sense*;
  $m = m + 1$;
  $t = t^0 + (m + 1)\triangle t$;
End

3. Another collocation method for point clouds

In the case of surfaces defined by point clouds, some approximated method is required to discretize space without analytic formulas for the normal vectors. In this section, we assume that the initial surface $\mathcal{S}^0$ is given by a set of points in $\mathbb{R}^d$. We also assume the surface evolution velocity $\boldsymbol{v} = \boldsymbol{v}(\boldsymbol{x}, t)$ is independent to the PDE solution. As surface points $\boldsymbol{x}(\boldsymbol{\varphi}, t)$ evolving from $\mathcal{S}^m$ to $\mathcal{S}^{m+1}$ $(m \ge 0)$ with velocity $\boldsymbol{v}$, we can approximate new points $\boldsymbol{x}^{m+1}$ on $\mathcal{S}^{m+1}$ by

$$\boldsymbol{x}^{m+1} \approx \boldsymbol{x}^m + \boldsymbol{v}\triangle t, \tag{11}$$

as shown in Figure 1. This process allows us to track all surfaces and place data points X and Z via local interpolation on them as in Algorithm 1; see Figure 2 for a schematic demonstration. The problem here is that $\mathcal{A}^m$ in (6) can no longer be obtained analytically. Our solution is to take the RBF pseudospectral approach [20] to approximate the surface Laplacian. The

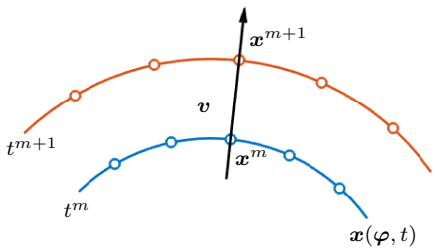
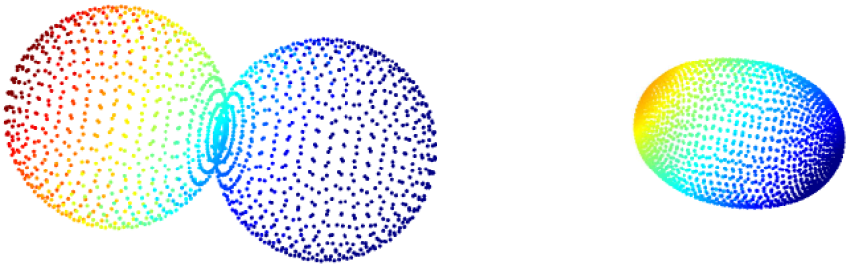


Figure 1: The point-to-point motion of a point $\boldsymbol{x}(\boldsymbol{\varphi}, t)$ on a surface from t^m to t^{m+1} with the velocity $\boldsymbol{v}$.



(a) $t = 0$ (b) $t = 1.5$

Figure 2: The snapshots of an evolving surface based on point clouds by mean-curvature motion.

ultimate goal is to obtain an approximated version of overdetermined linear system as in (10).

We will consider implementations of the following surface derivatives in extrinsic coordinates. Recall that $\nabla_{\mathcal{S}} := (I_d - \boldsymbol{n}\boldsymbol{n}^T)\nabla$. We can approximate $\boldsymbol{n} = \boldsymbol{n}(\boldsymbol{x})$ for $\boldsymbol{x} \in \mathcal{S}^m$ from the point cloud $\{\boldsymbol{x}_k^m\}_{k=1}^K$ specifying $\mathcal{S}^m$ by

$$\boldsymbol{n} = \frac{\nabla\Psi(\,\cdot\,, Z)\Psi(Z, Z)^{-1}\mathbf{1}_{n_Z}}{\|\nabla\Psi(\,\cdot\,, Z)\Psi(Z, Z)^{-1}\mathbf{1}_{n_Z}\|} : \mathcal{S}^m \to \mathbb{R}^d, \quad Z \subset \mathcal{S}^m \tag{12}$$

where $\mathbf{1}_{n_Z}$ is the all-ones vector of size n_Z, see [21, 22]. When implementing (12), we use subsets of Z surrounding the points of interest for the sake of efficiency. Once we have all normal vectors at X approximated, the computation of entries in the approximated gradient matrix $\widetilde{\nabla}_{\mathcal{S}}(X, Z)$ is straightforward.

The approximation error here is due to error in $\boldsymbol{n}$ and the exactness of data point locations on $\mathcal{S}^m$.

The same technique is undesirable for the approximation of the Laplace-Beltrami operator $\Delta_{\mathcal{S}} := \nabla_{\mathcal{S}} \cdot \nabla_{\mathcal{S}}$, as the surface divergence operator will act upon $\boldsymbol{n}$ and we simply want to avoid approximating second derivatives from point clouds. In the RBF pseudospectral approach, we rewrite (9) in terms of the unknown nodal data $u^m_{|Z} := u^m(Z)$ of u^m at trial centers $Z \subset \mathcal{S}^m$. This yields

$$u^m = \Psi(\,\cdot\,, Z)\Psi(Z, Z)^{-1} u^m_{|Z}. \tag{13}$$

We can apply the approximation and evaluate the surface gradient at Z while keeping $u^m_{|Z}$ unknown to yield the following $n_Z \times d$ matrix

$$(\widetilde{\nabla}_{\mathcal{S}} u^m)_{|Z} = \widetilde{\nabla}_{\mathcal{S}} \Psi(Z, Z)\Psi(Z, Z)^{-1} u^m_{|Z}.$$

At this point, we have approximated nodal values of each component of $\nabla_{\mathcal{S}}$ at Z, which still depends on the unknown $u^m_{|Z}$. Without explicitly writing out the index of these components for simplicity, we can express them similar to (13) by its interpolant

$$\widetilde{\nabla}_{\mathcal{S}} u^m = \Psi(\,\cdot\,, Z)\Psi(Z, Z)^{-1} (\widetilde{\nabla}_{\mathcal{S}} u^m)_{|Z},$$

to which we can apply an approximate surface divergence at $X \subset \mathcal{S}^m$ to yield

$$\widetilde{\Delta}_{\mathcal{S}} u^m = \big(\widetilde{\nabla}_{\mathcal{S}} \,\cdot\, \Psi\big)(X, Z)\Psi(Z, Z)^{-1} (\widetilde{\nabla}_{\mathcal{S}} u^m)_{|Z}.$$

This apparently complicated procedure did come with some neat simplifications [23]. Let $\boldsymbol{x} \in \mathcal{S}^m$ be a surface point on which the (approximated) normal vector is denoted by $\boldsymbol{n}(\boldsymbol{x})$. Also let

$$P(\boldsymbol{x}) = I_d - \boldsymbol{n}(\boldsymbol{x})\boldsymbol{n}(\boldsymbol{x})^T =: [p_1(\boldsymbol{x}), \ldots, p_d(\boldsymbol{x})] \in \mathbb{R}^{d \times d}$$

be the (approximated) projection operator to $\boldsymbol{x} \in \mathcal{S}^m$ with columns p_k, $k = 1, \ldots, d$. For each k, we define row-vector functions

$$G_k(\boldsymbol{x}, Z) := p_k(\boldsymbol{x})^T [\nabla \Psi(\boldsymbol{x}, Z)][\Psi(Z, Z)]^{-1}.$$

Then, we have

$$[\widetilde{\nabla}_{\mathcal{S}} u^m]_k = G_k(\,\cdot\,, Z) u^m_{|Z}, \quad k = 1, \ldots, d, \tag{14}$$

$$\widetilde{\Delta}_{\mathcal{S}} u^m = \sum_{k=1}^{d} G_k(\,\cdot\,, Z) G_k(Z, Z) u^m_{|Z}, \tag{15}$$

or, alternatively, we can use $(\widetilde{\nabla}_{\mathcal{S}} u^m)_{|X}$ as data to approximate surface Laplacian by

$$\widetilde{\Delta}_{\mathcal{S}} u^m \;=\; \sum_{k=1}^{d} G_k(\,\cdot\,, X) G_k(X, Z) u^m_{|Z}. \tag{16}$$

These equations, along with (13), are sufficient for obtaining an approximated version of the overdetermined linear system in (10) for solving (7) in terms of unknown $u_{|Z}$. Using the right most inverse of the interpolation matrix of Ψ at Z and the relationship $\boldsymbol{\lambda}^m_Z = [\Psi(Z,Z)]^{-1} u^m_{|Z}$, we can recast the system with unknown $\boldsymbol{\lambda}^m_Z$. Below is a pseudocode that summarizes the algorithm above: Algorithm 2, specifically denoted as Algorithm 2a by (15) and Algorithm 2b by (16).

Algorithm 2 a/b: For solving PDE (1) posed on point cloud

Initialization :
  set $t = t^0$, *select* $0 \le \theta \le 1$;
  define $Z^0, X^0 \subset \mathcal{S}^0$ *based on some point cloud*
  compute $\boldsymbol{n}(X^0)$ *using* (12);
  interpolate u_0 *at* Z^0 *to obtain* $\boldsymbol{\lambda}^0_Z$;
  $m = 1$;
While $(t \le T)$
  compute $\boldsymbol{v}$ *and update* $\{X^m, Z^m\} \subset \mathcal{S}^m$;
  approximate entries in (10) *using* (13) − (14), *and* (15)/(16);
  solve (10) *in the least-squares sense*;
  $m = m + 1$;
  $t = t^0 + (m+1)\triangle t$;
End

4. Numerical examples

We provide five examples to numerically study the proposed algorithms for solving convective-diffusion equations (1) on evolving surfaces. In all examples, we use the restricted Whittle-Matérn-Sobolev kernels and $\theta = 0.5$ in (7) for the Crank-Nicolson method (CN).

To quantify the accuracy and convergence of our numerical approximations, we employ two error measures as follows. Let u^* denote the exact

solution. The $L^\infty\big([0,T];L^2(\mathcal{S}(t))\big)$ error is defined by

$$\sup_{[0,T]} ||u^* - u||_{L^2(\mathcal{S}(t))}.$$

Similarly, the $L^\infty\big([0,T];\mathcal{H}^1(\mathcal{S}(t))\big)$ semi-norm error is given by

$$||\nabla_{\mathcal{S}} u^* - \nabla_{\mathcal{S}} u||_{L^\infty\left([0,T];L^2(\mathcal{S}(t))\right)},$$

and lastly, the $L^\infty\big([0,T];\mathcal{H}^2(\mathcal{S}(t))\big)$ semi-norm error is defined by

$$||\Delta_{\mathcal{S}} u^* - \Delta_{\mathcal{S}} u||_{L^\infty\left([0,T];L^2(\mathcal{S}(t))\right)}.$$

At any given time $t^m \in [0,T]$, $m \geq 1$, by using one of the above errors e^m and e^{m-1} at two consecutive time steps corresponding to fill distances $h_{X^{m-1}}$ and h_{X^m} for collocation point sets $X^{m-1} \subset \mathcal{S}^{m-1}$ and $X^m \subset \mathcal{S}^m$ respectively, we estimate the order of convergence by

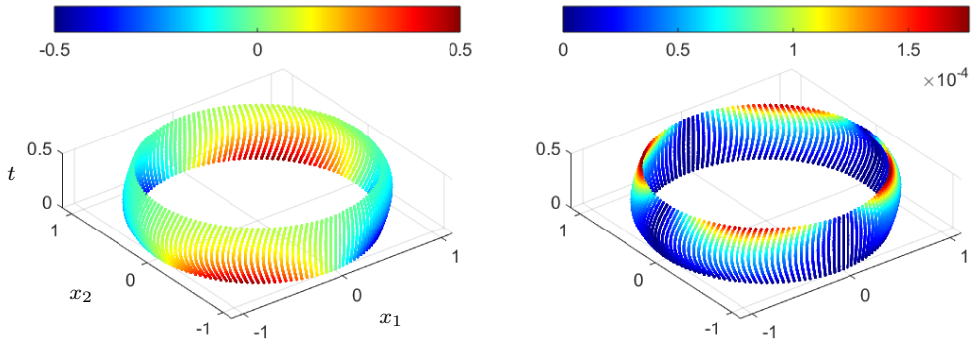
$$\text{eoc}^m := \frac{\log(e^m/e^{m-1})}{\log(h_{X^m}/h_{X^{m-1}})}. \tag{17}$$

In cases of smooth initial conditions (ICs) on surfaces without (extremely) high curvatures, we can alternatively employ overdetermined expressions with $n_X > n_Z$ or exactly determined formulas with the same point sets ($X = Z$) for both Algorithms 1 and 2. Having said that, oversampling in Algorithm 1 can still result in better accuracy; see Example 4.1. The situations are quite different when imposing discontinuous ICs on surfaces with high curvatures, in which oversampling is essential. In our concluding simulations in Examples 4.4 and 4.5, we will demonstrate the robustness of our proposed methods.

Example 4.1. *An evolving curve in* $\mathbb{R}^2$.

In the first example, we compare Algorithm 1 with two time-dependent FEMs [8, 15] for solving (1) on two moving surfaces. We use the same diffusivity $\varepsilon = 1$ and the source terms $f(\boldsymbol{x},t)$ by the symbolic mean by providing the same exact solutions $u^*(\boldsymbol{x},t)$ in the FEM papers. The smoothness order is set to be $\mu = 4$.

In [15], weak formulations of (1) were discretized in some narrow bands containing the evolving surface, and an implicit temporal scheme was used to



(a) numerical solutions (b) error functions

Figure 3: Example 4.1. The numerical solutions and error functions (color) relative to $u^*(\boldsymbol{x}, t)$ for $t \in [0, 0.5]$, obtained by oversampled Algorithm 1 with $h_Z = 2^{-5}\sqrt{2}$ with $(n_X, n_Z) = (214, 143)$, using the restricted Whittle-Matérn-Sobolev kernels of smoothness order 4, for solving a convective-diffusion equation on a moving curve.

solve (1), which yields an *unfitted finite element method*. We aim to examine the same experiment as in [15, Section 6.3] with the use of Algorithm 1, imposed on the following evolving curve,

$$\mathcal{S}(\boldsymbol{x}, t) = \frac{x_1^2}{1 + 0.25\sin(2\pi t)} + x_2^2 - 1 = 0 \quad \text{for } t \in [0, 0.5].$$

First, we test the Algorithm 1 with oversampling. In Figure 3, the numerical solutions and their error functions are plotted by color on curves at some time points from $t = 0$ to 0.5, when using $h = 2^{-5}\sqrt{2}$. Table 1 shows the corresponding $L^\infty\big([0, T]; L^2(\mathcal{S}(t))\big)$ errors under several different oversampling settings of $n_X > n_Z$ with the corresponding time step $\triangle t = h_Z/4$. With the increasing numbers of both centers and collocation points, the $L^\infty\big([0, T]; L^2(\mathcal{S}(t))\big)$ errors decrease. We then compare results in Table 1 with those listed in Table 2 obtained by the corresponding exactly determined settings $X = Z$. We find that using $h_X < h_Z$ can improve accuracy slightly. Under such smoothness order for the kernel, oversampling or not both yield the expected second-order $L^\infty\big([0, T]; L^2(\mathcal{S}(t))\big)$ convergence. Notably, even for settings with very few data points, say $n_Z = 5$, we still observe accuracy of 10^{-2} in $L^\infty\big([0, T]; L^2(\mathcal{S}(t))\big)$ norms that outperforms the FEM mehtod.

For comparison with the results of the unfitted FEM as found in [15], Ta-

Table 1: Example 4.1. The $L^\infty\big([0,T];L^2(\mathcal{S}(t))\big)$ errors and the corresponding estimated order of convergence (eoc) by (17) with respect to h_Z, obtained by Algorithm 1 with overdetermined formulas ($n_X > n_Z$), under the same other settings as in Figure 3.

h_Z	n_Z	n_X	$L^\infty\big([0,T];L^2(\mathcal{S}(t))\big)$	eoc
$\sqrt{2}$	5	7	2.31823e-02	–
$2^{-1}\sqrt{2}$	9	14	2.29316e-03	3.33762
$2^{-2}\sqrt{2}$	18	27	5.44931e-04	2.07319
$2^{-3}\sqrt{2}$	26	54	1.34549e-04	2.01794

Table 2: Example 4.1. The $L^\infty\big([0,T];\mathcal{H}^2(\mathcal{S}(t))\big)$ and $L^\infty\big([0,T];L^2(\mathcal{S}(t))\big)$ errors and the corresponding convergence rates (eoc) with respect to h, obtained by Algorithm 1 with exactly determined formulas and the unfitted FEM [15], under the same other settings as in Figure 3.

h	n	Algorithm 1 with $X=Z$ ($\triangle t = h/4$)				Unfitted FEMs [15] ($\triangle t = 2h^2$)	
		$L^\infty\big([0,T];\mathcal{H}^2(\mathcal{S}(t))\big)$	eoc	$L^\infty\big([0,T];L^2(\mathcal{S}(t))\big)$	eoc	$L^\infty\big([0,T];L^2(\mathcal{S}(t))\big)$	eoc
$\sqrt{2}$	5	1.14643e-1	–	3.85031e-2	–	–	–
$2^{-1}\sqrt{2}$	9	8.38177e-3	3.77376	2.60321e-3	3.88661	1.15457e-1	–
$2^{-2}\sqrt{2}$	18	2.00492e-3	2.06371	5.47333e-4	2.24980	3.25344e-2	1.82732
$2^{-3}\sqrt{2}$	26	5.23047e-4	1.93853	1.34576e-4	2.02400	8.64172e-3	1.91258
$2^{-4}\sqrt{2}$	72	1.30694e-4	2.00075	3.36858e-5	1.99820	2.13241e-3	2.01883
$2^{-5}\sqrt{2}$	143	3.31576e-5	1.97878	8.44284e-6	1.99634	5.42960e-4	1.97357

ble 2 also lists the $L^\infty\big([0,T];L^2(\mathcal{S}(t))\big)$ errors with $\triangle t = 2h^2$, as well as the estimates in $L^\infty\big([0,T];\mathcal{H}^2(\mathcal{S}(t))\big)$ error of our Algorithm 1 with $X = Z$. It also presents convergence orders as computed by (17). Algorithm 1 achieves higher-order accuracy in $L^\infty\big([0,T];L^2(\mathcal{S}(t))\big)$ and $L^\infty\big([0,T];\mathcal{H}^2(\mathcal{S}(t))\big)$ norms, using a relatively large $\triangle t = h/4$. The magnitudes of our corresponding $L^\infty\big([0,T];L^2(\mathcal{S}(t))\big)$ errors are lower by a factor of about 100 relative to the unfitted FEM with many more iterations ($\triangle t = 2h^2$).

Example 4.2. *An evolving surface in* $\mathbb{R}^3$.

This is a 3D version of the previous example, but we focus on the exactly determined case with $X^m = Z^m$ at all time steps. We use kernels with the smoothness order 4. In [8], two Eulerian FEMs (the cGdG and cGcG methods) employed Galerkin methods both in time and space with narrow bands to solve (1). The test problem in [8, Section 11.3] is posed on a moving

Table 3: Example 4.2. The corresponding $L^\infty\big([0,T];\mathcal{H}^2(\mathcal{S}(t))\big)$ and $L^\infty\big([0,T];\mathcal{H}^1(\mathcal{S}(t))\big)$ errors and the estimated order of convergence (eoc) by (17) using $h=\triangle t$, compared with the $L^\infty\big([0,T];\mathcal{H}^1(\mathcal{S}(t))\big)$ results in [8, Table 4], under the same settings as in Figure 4.

		Algorithm 1				cGdG		cGcG	
h	n	$L^\infty([0,T];\mathcal{H}^2(\mathcal{S}(t)))$	eoc	$L^\infty([0,T];\mathcal{H}^1(\mathcal{S}(t)))$	eoc	$L^\infty([0,T];\mathcal{H}^1(\mathcal{S}(t)))$	eoc	$L^\infty([0,T];\mathcal{H}^1(\mathcal{S}(t)))$	eoc
3/4	36	5.5926e-3	–	3.7442e-3	–	–	–	–	–
1/2	78	1.9828e-3	2.557	9.2059e-4	3.460	0.574872	–	1.12954	–
1/4	312	4.3551e-4	2.187	1.8276e-4	2.333	0.303749	0.920	0.550169	1.04
1/8	1206	9.6060e-5	2.181	4.0135e-5	2.187	0.157731	0.945	0.249556	1.14
1/16	4836	2.3407e-5	2.037	9.7715e-6	2.038	0.0803239	0.974	0.112138	1.15

surface given by

$$\mathcal{S}(\boldsymbol{x},t)=\Big(\frac{x_1}{1+0.25\sin(t)}\Big)^2+x_2^2+x_3^2-1=0 \quad \text{ for } t\in[0,4].$$

We solve the same problem by Algorithm 1 with $\triangle t=1/8$. Based on the initial points with $h:=h_{Z^0}=\triangle t$, we move each point by point-to-point motion based on the analytical formula in polar coordinates, that is, the fill distance h_{Z^m} slightly varies in time periodically in this experiment. In Figure 4, we present a few snapshots of the evolved surfaces and numerical solutions by colormap.

To be compared with [8, Figures 13-14], Figure 5 illustrates the convergence profile of Algorithm 1 in $L^\infty\big([0,T];L^2(\mathcal{S}(t))\big)$ errors. First, Figure 5(a) is obtained by three fixed h-values as $\triangle t$ varies. It can be seen that errors for $h=1/4$ are almost as accurate as those for $h=1/8$, under the same $\triangle t$.

Our comparison with the cGdG and cGcG approaches in [8] is as follows. With respect to time, our convergence rates and those of the two FEMs are all around order 2 due to CN. In terms of accuracy, we obtain errors less than 10^{-5} by using $h=1/4$, while their best FEM result shown is approximately 10^{-4} using a much smaller $h=1/64$. For spatial convergence, we fix a several $\triangle t$ and let h vary to obtain Figure 5(b). Generally speaking, our numerical solutions are more accurate based on [8, Figures 13-14], which used smaller h-values for each $\triangle t$. Our method shows a rate of spatial convergence of 6.4, whereas that in [8] is around 2.

Table 3 further lists the $L^\infty\big([0,T];\mathcal{H}^2(\mathcal{S}(t))\big)$ and $L^\infty\big([0,T];\mathcal{H}^1(\mathcal{S}(t))\big)$ estimates obtained by Algorithm 1, and the $L^\infty\big([0,T];\mathcal{H}^1(\mathcal{S}(t))\big)$ results of two Eulerian FEMs in [8] using various $h=\triangle t$. Note that Algorithm 1 remains second order while the FEM methods drop to first order.

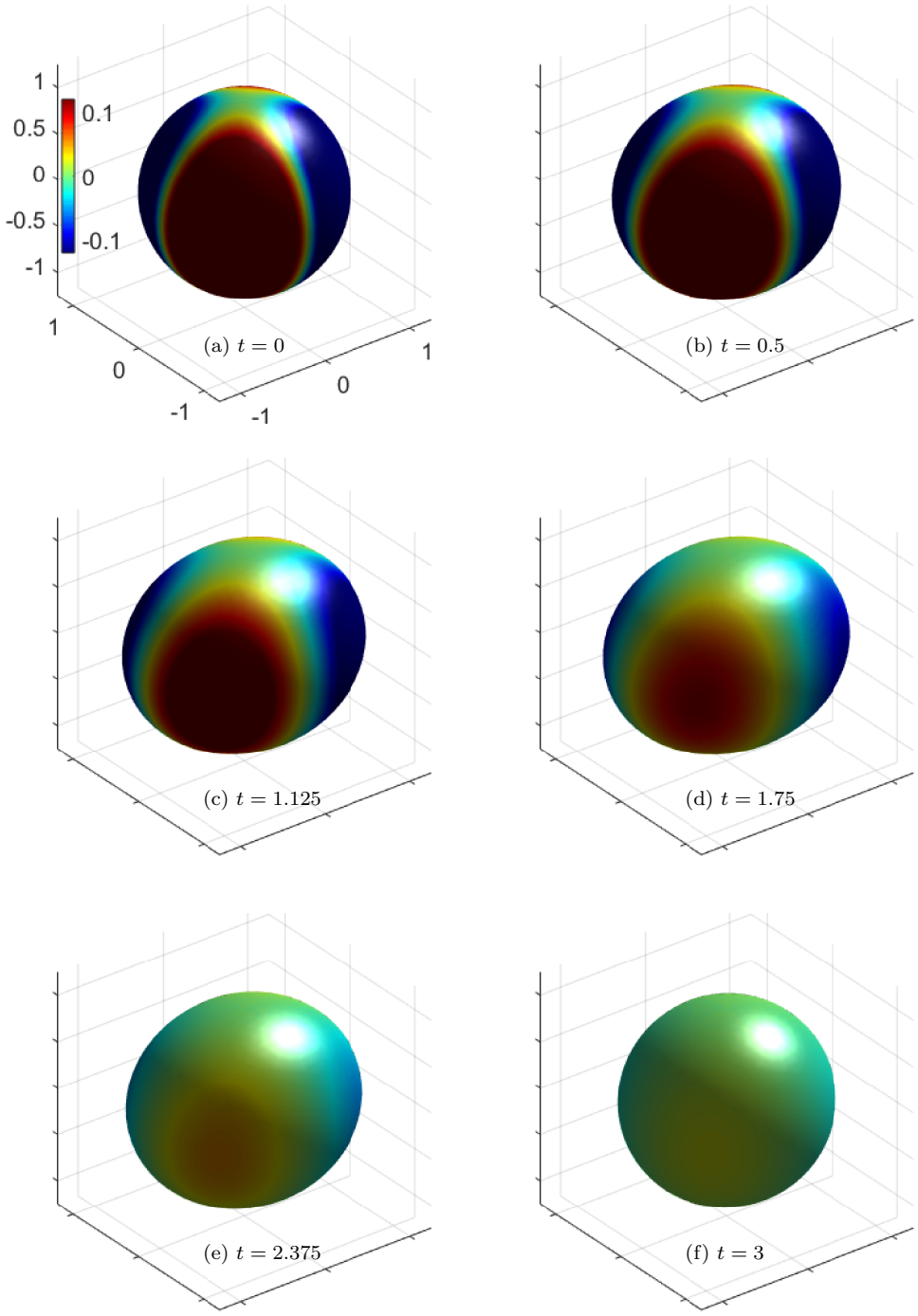
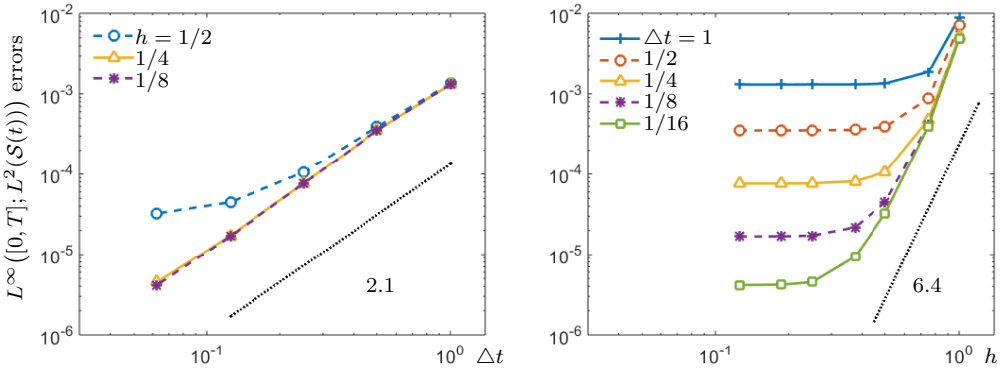


Figure 4: Example 4.2. Snapshots of numerical solutions (color) at several times, obtained by Algorithm 1, using the kernel of smoothness order 4 and $h = \triangle t = 1/8$, on a moving surface.



(a) convergence for $\triangle t$ (b) convergence for h

Figure 5: Example 4.2. The $L^\infty\big([0,T];L^2(\mathcal{S}(t))\big)$ error profiles with different h and $\triangle t$, obtained by our proposed Algorithm 1, under the same settings as in Figure 4.

Example 4.3. *Mass conservation on an expanding sphere.*

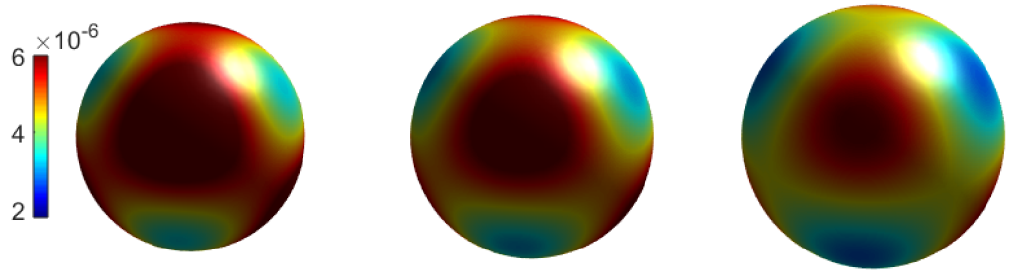
Aiming to verify the mass conservation law in (2), we solve PDE (1) without the source term, i.e., $f = 0$. We choose a small diffusion coefficient $\varepsilon = 10^{-3}$ to approximate the viscosity solutions to (1). The tested surface is defined by

$$\mathcal{S}(\boldsymbol{x}, t) = x_1^2 + x_2^2 + x_3^2 - (e^{t/5})^2 = 0 \quad \text{for } t \in [0, 1].$$

We set the initial solution to be $u^*(\boldsymbol{x}, 0) = 0.5 + x_1x_2x_3$.

We use $X = Z$ in both Algorithms 1 and 2 (Algorithm 2a and b are practically identical here) to solve this problem with the kernels of order 4 and $h = \triangle t = 0.02$. Solutions of two algorithms agree up to 10^{-6} approximately; their difference on $\mathcal{S}(t)$ is shown in Figure 6 for a particular parameter setup.

Now we can see Figure 7 for the masses of the solutions $\int_{\mathcal{S}(t)} u \, \mathrm{d}\mathcal{S}$ over time, which are approximated by $\sum_i^{n_X} u(\boldsymbol{x}_i)\triangle\mathcal{S}_i$ at X based on some triangle mesh $\mathcal{T}$. It is easy to see that $u(\boldsymbol{x}_i)\triangle\mathcal{S}_i = u(\boldsymbol{x}_i)\mathrm{Area}(\mathcal{T}_i)/3$ where $\mathcal{T}_i$ is the triangle containing $\boldsymbol{x}_i$. For Figure 7(a), it is clear that both algorithms conserve mass as desired. When we subtract the exact mass computed using $u^*(\boldsymbol{x}, 0)$ to yield errors in Figure 7(b), we can see the different mass-conservation behaviour of the proposed algorithms. The analytically exact Algorithm 1 shows gradually increasing error in time, which is typical for accumulative error. On the other hand, the nearly-constant error curve of



(a) $t = 0.3$ (b) $t = 0.6$ (c) $t = 1$

Figure 6: Example 4.3. Difference functions of solutions obtained by Algorithms 1 and 2a, under $h = \triangle t = 0.02$, for solving a convective-diffusion equation on a sphere expanding along outer normal directions.

Algorithm 2 in Figure 7(b) suggests that the error in numerical mass comes from the approximation steps in the formulation.

Example 4.4. *Continuous initial conditions on a point cloud with high curvatures.*

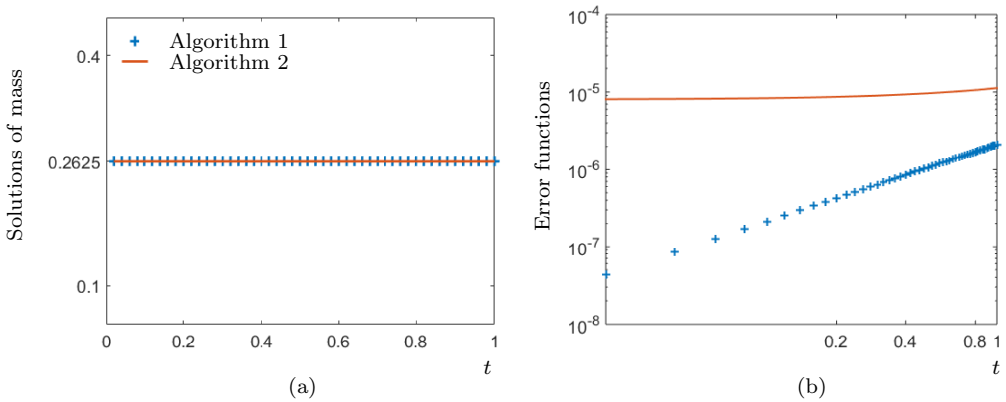
The last two examples aim to prepare for problems on merging surfaces. In both cases, we consider the mean-curvature motion of kissing-spheres, as in Figure 2, which has high initial curvature at the contact point. Here, we use the FDM proposed in [24] to obtain point clouds in $\mathbb{R}^3$ at each discrete time. Without analytic formula for $\mathcal{S}(t)$, we can only use Algorithm 2.

In order to test the effect of geometry, we first impose a continuous IC

$$u^*(\boldsymbol{x}, 0) = \sin(x + z)$$

on the surface. The arrows in Figure 8 show mean-curvature velocities and hence the directions of motion. The diffusion coefficient and time step size are set as $\varepsilon = 1$ and $\triangle t = 0.001$ respectively. First, we solve (1) with $f = 0$ by using the exactly determined formulas with $(n_X = n_Z = 1196)$. Figure 9 presents two numerical results at $t = 0.05$ and 0.2. From Figure 9(a), some oscillations near the region with high curvature can be observed. Once diffusion starts, the oscillatory solution smooths out to the whole surface, see Figure 9(b). We postpone the comparison with oversampling to the next example.

Example 4.5. *A discontinuous initial condition imposed on point clouds evolving by mean-curvature motion.*



(a) (b)

Figure 7: Example 4.3. (a) Solutions of mass and (b) error functions between exact mass and them, obtained by Algorithms 1 and 2 respectively, under the same settings as in Figure 6.

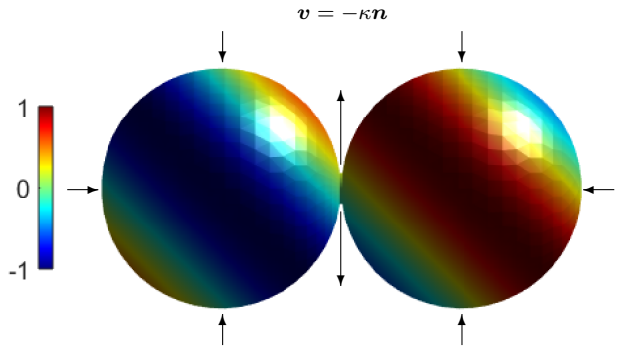
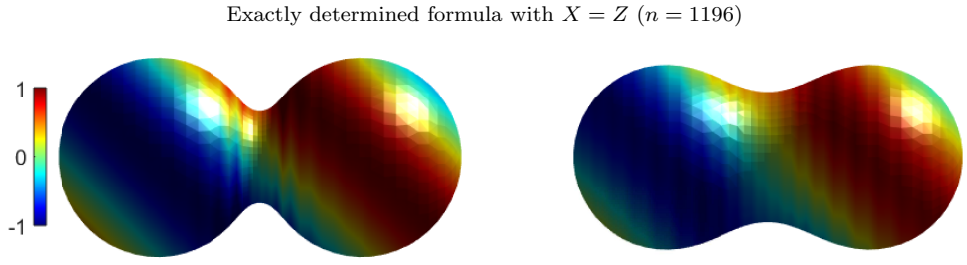


Figure 8: Example 4.4. A continuous IC (color) on a surface that evolves by mean curvature motion (arrows are the velocities $\boldsymbol{v}$).



(a) $t = 0.05$ (b) 0.2

Figure 9: Example 4.4. Numerical solutions at $t = 0.05$ and 0.2, obtained by Algorithm 2 with exactly determined formulas. We take $\triangle t = 0.001$ and the kernels of smoothness order 2.

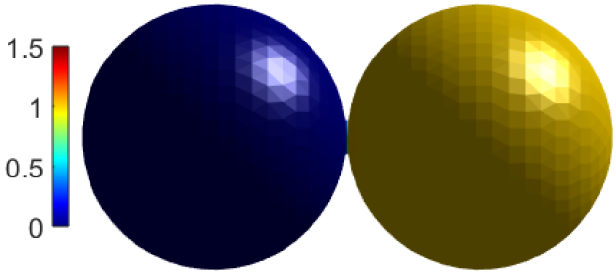


Figure 10: Example 4.5. Discontinuous initial conditions for a PDE posed on evolving surface under curvature motion.

We continue the study on the initial high-curvature surface in Example 4.4 but use a discontinuous IC defined by

$$u^*(\boldsymbol{x}, 0) = 0 \ (x_1 < 0) + 1 \ (x_1 \geq 0),$$

as shown in Figure 10. In the following, we will examine the robustness of Algorithms 2a and 2b.

The main results were shown in Figure 11. Because of the discontinuity, it makes sense to include some common regularization techniques into our comparison. We apply a regularized RBF interpolant [25] to interpolate the IC; then, we update it in time with $X = Z$. The employed regularization parameter $p = 0.2488$ is chosen by L-curves in $L^2(\mathcal{S}(t))$ errors [26]. From Figure 11(a)-(b), this common-sense setup yields undesirable solutions and we clear see the oscillatory solutions propagating towards the whole surface over time.

Now we focus on the results from Algorithms 2a and b shown in Figures 11(c)-(d) and (e)-(h) respectively. As Algorithm 2b is a computationally more expensive method, we use it with fewer Z nodes to compare fairly. To sum up, Algorithm 2a still show the spurious oscillations initially that will be corrected and the solutions get smoother with increasing time. Solutions of Algorithm 2b, in contrast, are physically correct without oscillations.

Before example ends, we want to make sure that Algorithm 2 really is honestly more suitable than regularization. We now apply regularization at each time, instead of to the initial time only, hoping to avoid oscillatory solutions. Various regularization parameters p were used, all of which yield non-oscillatory smooth *wrong* solutions. We make this conclusion based on

mass conservation in Figure 12.

5. Conclusions

We propose some numerical algorithms based on extrinsic kernel-based meshless techniques for solving parabolic PDEs on evolving surfaces. The proposed algorithms can be implemented either by an analytic projection or a pseudospectral approximation for differential operators on surfaces. Extension into embedding spaces is unnecessary.

The analytic approach (Algorithm 1) can achieve high-order accuracy and convergence in space when normal vectors of the surface can be differentiated analytically. In other cases, the approximated meshless method (Algorithm 2) becomes handy and it can handle PDEs defined on point clouds. Another issue of study is the oversampling strategy, viz., using denser collocation points than trial centers, to yield overdetermined matrix systems to be solved by least-squares. For smooth problems (including initial conditions, surfaces, and solutions), the need of oversampling is not obvious. We demonstrate that, when the initial condition is discontinuous and when the surface has high curvature, oversampling is essential for getting physically correct solutions.

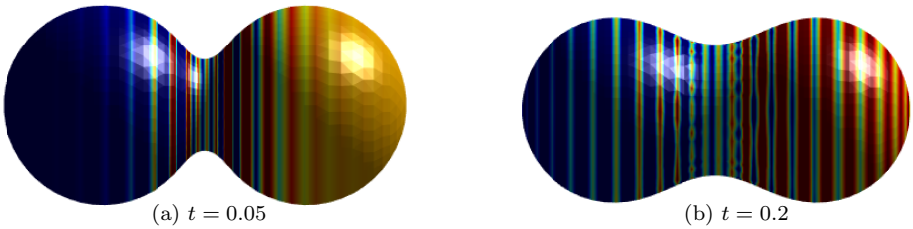
References

[1] C. Eilks, C. M. Elliott, Numerical simulation of dealloying by surface dissolution via the evolving surface finite element method, Journal of Computational Physics 227 (23) (2008) 9727–9741 (2008).

[2] R. Barreira, C. M. Elliott, A. Madzvamuse, The surface finite element method for pattern formation on evolving biological surfaces, Journal of Mathematical Biology 63 (6) (2011) 1095–1119 (2011).

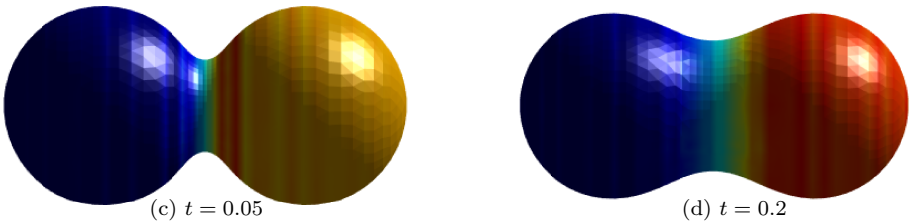
[3] C. M. Elliott, B. Stinner, Modeling and computation of two phase geometric biomembranes using surface finite elements, Journal of Computational Physics 229 (18) (2010) 6585–6612 (2010).

[4] C. M. Elliott, B. Stinner, C. Venkataraman, Modelling cell motility and chemotaxis with evolving surface finite elements, Journal of the Royal Society, Interface 9 (76) (2012) 3027–3044 (2012).



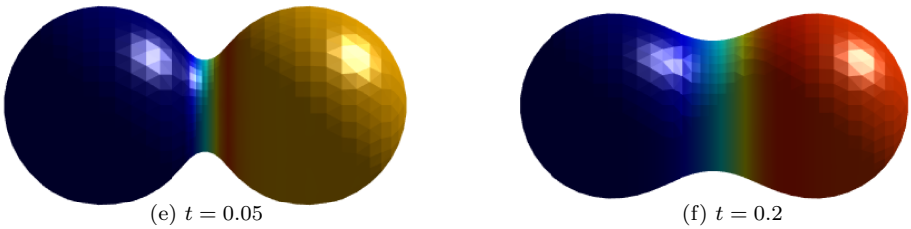
(a) $t = 0.05$ (b) $t = 0.2$

Exactly determined formula with IC regularization based on $X = Z$ ($n = 1196$)

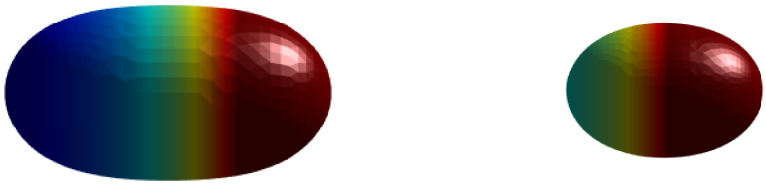


(c) $t = 0.05$ (d) $t = 0.2$

Overdetermined formula: Algorithm 2a with $(n_X, n_Z) = (1736, 1196)$



(e) $t = 0.05$ (f) $t = 0.2$



(g) $t = 1$ (h) $t = 1.5$

Overdetermined formula: Algorithm 2b with $(n_X, n_Z) = (1080, 710)$

Figure 11: Example 4.5. Numerical solutions at various t, obtained by the exactly determined formulations with IC regularization with a regularization parameter $p = 0.2488$ in (a)-(b), our Algorithm 2a in (c)-(d) and Algorithm 2b in (e)-(h) with different point sets for X and Z, under the same settings as in Figure 9.

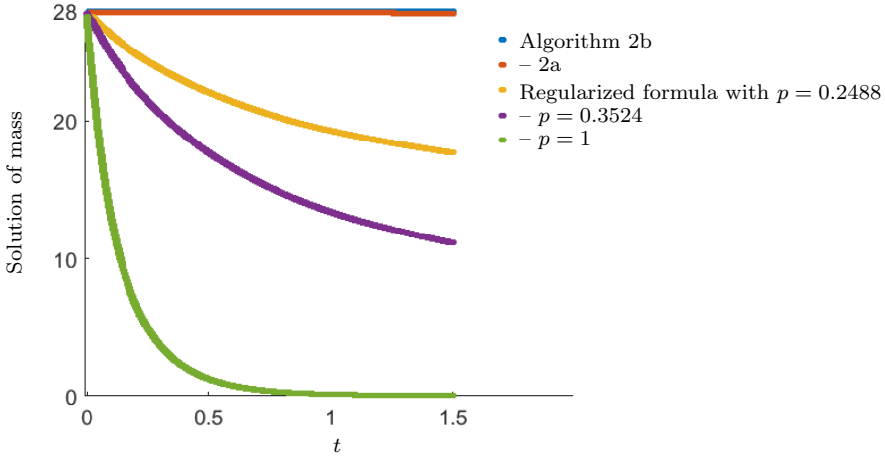


Figure 12: Example 4.5. Numerical solutions of mass from $t = 0$ to 1.5, obtained by Algorithm 2b and Algorithm 2a with $(n_X, n_Z) = (1736, 1196)$, as well as the regularized formula with $n = 1196$ using three different regularization parameters $p = 0.2488,\ 0.3524$, and 1 at all times, under the same settings as in Figure 11.

[5] H. A. Stone, A simple derivation of the time-dependent convective-diffusion equation for surfactant transport along a deforming interface, Physics of Fluids A: Fluid Dynamics 2 (1) (1990) 111–112 (1990).

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

[7] G. Dziuk, C. M. Elliott, An Eulerian approach to transport and diffusion on evolving implicit surfaces, Computing and Visualization in Science 13 (1) (2010) 17–28 (2010).

[8] J. Grande, Eulerian finite element methods for parabolic equations on moving surfaces, SIAM Journal on Scientific Computing 36 (2) (2014) B248–B271 (2014).

[9] G. Dziuk, Finite elements for the Beltrami operator on arbitrary surfaces, Springer Berlin Heidelberg, 1988 (1988).

[10] S. J. Ruuth, B. Merriman, A simple embedding method for solving partial differential equations on surfaces, Journal of Computational Physics 227 (3) (2008) 1943–1961 (2008).

[11] C. Piret, The orthogonal gradients method: A radial basis functions method for solving partial differential equations on arbitrary surfaces, Journal of Computational Physics 231 (14) (2012) 4662–4675 (2012).

[12] K. C. Cheung, L. Ling, A kernel-based embedding method and convergence analysis for surfaces PDEs, SIAM Journal on Scientific Computing 40 (1) (2018) A266–A287 (2018).

[13] T. März, C. B. Macdonald, Calculus on surfaces with general closest point functions, SIAM Journal on Numerical Analysis 50 (6) (2012) 145–189 (2012).

[14] T. März, P. Rockstroh, S. J. Ruuth, An embedding technique for the solution of reaction-diffusion equations on algebraic surfaces with isolated singularities, Journal of Mathematical Analysis and Applications 436 (2) (2016) 911–943 (2016).

[15] K. Deckelnick, C. M. Elliott, T. Ranner, Unfitted finite element methods using bulk meshes for surface partial differential equations, SIAM Journal on Numerical Analysis 52 (4) (2014) 2137–2162 (2014).

[16] C. M. Elliott, B. Stinner, V. Styles, R. Welford, Numerical computation of advection and diffusion on evolving diffuse interfaces, IMA Journal of Numerical Analysis 31 (3) (2010) 786–812 (2010).

[17] M. Chen, L. Ling, Extrinsic meshless collocation methods for PDEs on manifolds, SIAM Journal on Numerical Analysis (In press 2020).

[18] K. C. Cheung, L. Ling, R. Schaback, H^2-convergence of least-squares kernel collocation methods, SIAM Journal on Numerical Analysis 56 (1) (2018) 614–633 (2018).

[19] E. J. Fuselier, G. B. Wright, Scattered data interpolant on embedded submanifolds with restricted positive definite kernels: Sobolev error estimates, SIAM Journal on Numerical Analysis 50 (3) (2012) 1753–1776 (2012).

[20] N. Flyer, G. B. Wright, A radial basis function method for the shallow water equations on a sphere, in: Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences, The Royal Society, 2009, pp. rspa–2009 (2009).

[21] B. Fornberg, T. A. Driscoll, G. Wright, R. Charles, Observations on the behavior of radial basis function approximations near boundaries, Computers & Mathematics with Applications 43 (3-5) (2002) 473–490 (2002).

[22] E. Marchandise, C. Piret, J. F. Remacle, CAD and mesh repair with radial basis functions, Journal of Computational Physics 231 (5) (2012) 2376–2387 (2012).

[23] E. J. Fuselier, G. B. Wright, A high-order kernel method for diffusion and reaction-diffusion equations on surfaces, Journal of Scientific Computing 56 (3) (2013) 535–565 (2013).

[24] L. Bronsard, B. T. Wetton, A numerical method for tracking curve networks moving with curvature motion, Journal of Computational Physics 120 (1) (1995) 66–87 (1995).

[25] G. E. Fasshauer, Meshfree Approximation Methods with MATLAB, Vol. 6, World Scientific Publishing Company, 2007 (2007).

[26] P. C. Hansen, The L-curve and its use in the numerical treatment of inverse problems, in: Computational Inverse Problems in Electrocardiology, ed. P. Johnston, Advances in Computational Bioengineering, WIT Press, 2000, pp. 119–142 (2000).