Available online at www.sciencedirect.com
ScienceDirect
Journal of Computational Physics 227 (2008) 1943–1961
JOURNAL OF COMPUTATIONAL PHYSICS
www.elsevier.com/locate/jcp

A simple embedding method for solving partial differential equations on surfaces

Steven J. Ruuth [a,*,1], Barry Merriman [b]

[a] Department of Mathematics, Simon Fraser University, Burnaby, British Columbia, Canada V5A 1S6
[b] University of California, Los Angeles, CA, United States

Received 13 October 2006; received in revised form 12 June 2007; accepted 3 October 2007
Available online 22 October 2007

Abstract

It is increasingly common to encounter partial differential equations (PDEs) posed on surfaces, and standard numerical methods are not available for such novel situations. Herein, we develop a simple method for the numerical solution of such equations which embeds the problem within a Cartesian analog of the original equation, posed on the entire space containing the surface. This allows the immediate use of familiar finite difference methods for the discretization and numerical solution. The particular simplicity of our approach results from using the closest point operator to extend the problem from the surface to the surrounding space. The resulting method is quite general in scope, and in particular allows for boundary conditions at surface boundaries, and immediately generalizes beyond surfaces embedded in $\mathbb{R}^3$, to objects of any dimension embedded in any $\mathbb{R}^n$. The procedure is also computationally efficient, since the computation is naturally only carried out on a grid near the surface of interest. We present the motivation and the details of the method, illustrate its numerical convergence properties for model problems and also illustrate its application to several complex model equations.

Keywords: Closest point representations; Partial differential equations; Implicit surfaces; Finite difference schemes; Manifolds; Laplace–Beltrami operator

1. Introduction

Many applications in the natural and applied sciences require the solutions of partial differential equations (PDEs) on surfaces or more general manifolds. Examples of such application areas arise in biological systems, image processing, medical imaging, mathematical physics, fluid dynamics and computer graphics. For example, applications in image processing include the generation of textures [27,28], the visualization of vector

* Corresponding author.
E-mail addresses: sruuth@sfu.ca (S.J. Ruuth), barrym@ucla.edu (B. Merriman).
[1] The work of this author was partially supported by a grant from NSERC Canada.

0021-9991/$ - see front matter © 2007 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2007.10.009

fields [7] and weathering [8], while applications in fluid dynamics include flows and solidification on surfaces [17,18], and the problem of evolving surfactants on interfaces [29].

A popular approach to solving PDEs on surfaces is to impose a smooth coordinate system or parameterization on the surface, express differential operators within these coordinates, and then discretize the resulting equations. See, for example [10] for a tutorial and survey of methods for parameterizing surfaces. However, the required coordinates can be complicated or impractical to construct, and the coordinates may introduce artificial singularities, such as at the poles in spherical coordinates. Indeed, as pointed out in [10], "parameterizations almost always introduce distortion in either angles or some region". Also, equations that are simple when written using intrinsic derivatives, such as surface diffusion, become substantially more complex when written in a coordinate system, involving nonconstant coefficients and more derivative terms.

Another common approach to solving partial differential equations on surfaces is to solve the PDE directly on a triangulation of the surface. This approach can be effective for certain classes of equations, however, as a general technique it leads to a number of difficulties. These are discussed in [3,4]. In particular, this approach leads to nontrivial discretization procedures for the differential operators, as well as difficulties in accurately computing geometric primitives, such as surface normals and curvature [3]. In addition, convergence of numerical schemes on triangulated surfaces remains less well understood than methods on Cartesian grids [11].

An alternative approach to treating PDEs on surfaces is to embed the surface differential equations of interest within differential equations posed on all of $\mathbb{R}^3$, so that the solution of the embedding equations, when restricted to the surface, provides the solution to the original problem. With such an approach, the ultimate goal is to develop a method that allows the treatment of general, complex surface geometry, while still retaining the simplicity that comes from working in standard Cartesian coordinates. With this in mind, Bertalmío et al. [3] introduced an embedding method for solving variational problems and the resulting Euler–Lagrange evolution PDEs on surfaces. In their approach, the underlying surface is represented as the level-set of a higher dimensional function and the evolution corresponding to the surface PDEs is carried out via PDEs that are posed on all of $\mathbb{R}^3$. This leads to equations that can be discretized and solved using Cartesian grid methods. A further improvement to this method was proposed by Greer [11]. In his approach, the evolution equation is modified away from the surface to maintain greater regularity of the solution near the surface of interest. See also [1] for related work on the finite element approximation of elliptic partial differential equations on implicit surfaces via level-set methods.

Embedding methods based on level-set methods have a number of limitations, however. Most obviously, these methods do not immediately allow for open surfaces with boundaries, or filamentary objects of codimension-two or higher, although it is in principle possible to represent such objects by introducing additional level-set functions. Another limitation is that these methods result in an embedding PDE posed on all of space, and complications arise when they are solved in a restricted computational band around the surface. Such "narrow banding" requires the imposition of appropriate boundary conditions, and how to best impose these conditions is not generally understood. For example, even using the regularity improvements proposed by Greer [11], a degradation of the order of convergence is observed when banding is used in diffusive problems. At a more technical level, level-set based methods also either lead to degenerate diffusion equations or require the use of an additional diffusion step when applied to parabolic equations [11].

Similar to other embedding methods, the approach we present here discretizes the partial differential equations using a fixed Cartesian grid in the embedding space. However, our method is based on the use of a closest point representation of the surface (cf. [16]) rather than a level-set representation. In conjunction with this change in representation, we abandon the concept of solving an embedding PDE for all time, and instead use the embedding PDE to advance the solution near the surface for one time step (or one stage of a Runge–Kutta method). This leads to a new method for solving PDEs on surfaces which has great simplicity, as well as additional desirable features. Most importantly, the embedding PDE is the obvious analog of the surface PDE, and involves only the standard Cartesian differential operators. In addition, the method can treat open surfaces and is not limited to objects of codimension-one. It also naturally allows the computation to be done on a grid defined in a narrow band near the surface without any degradation in the order of accuracy and without imposing artificial boundary conditions. To distinguish our approach from other embedding procedures, and to emphasize its essential reliance on the closest point representation, we refer to the method as the *closest point method*. As an aside, note that a special case of this procedure was introduced in our recent

paper on the diffusion-generated motion of curves on surfaces [16]. It was in this context that the general potential of the approach was first realized, although it occurs only as a special, simple procedure for approximating in-surface curvature motion according to the diffusion-generated motion algorithm.

The paper unfolds as follows. Section 2 gives the method, describes its implementation and provides an analysis. In Section 3, we give a number of two-dimensional convergence studies to validate the method. Section 4 considers a variety of three-dimensional convergence tests and provides some examples that are relevant to biology: a Fitzhugh–Nagumo and a morphogenesis simulation. In Section 5, we give a summary and discuss some of our ongoing projects in the subject. Finally, Appendix A concludes with a discussion on the computation of geometric quantities defined on the surface.

2. The closest point method

In this section, we describe the method as well as its motivation, analysis and implementation. We begin by discussing the motivation for the method and its surface representation.

2.1. The closest point representation

As part of our method, we need to rapidly and accurately extend functions defined on surfaces to a neighborhood of the surface. For maximal simplicity, this extension will be chosen according to the principle that the embedding PDE should be the natural extension of the original, i.e., we form the embedding PDE by replacing intrinsic surface gradients with standard gradients on $\mathbb{R}^3$. Clearly, both evolutions cannot agree for long times, but if we select a suitable extension the evolution of the embedding PDE will be accurate initially, which will be sufficient to update the solution in time. Thus, we will select an operator, E, that extends functions defined on the surface to functions defined on all of $\mathbb{R}^3$ in such a way that the natural extension of the surface PDE to all of space generates the required rates of change.

Note that the requirement that the all-space equation is the natural extension of the intrinsic surface equation leads to a specific class of extension operators, namely those which extend function values to be constant along the directions normal to the surface. To illustrate this, consider the special case of the prototype Hamilton–Jacobi equation defined on the surface,

$$\frac{\partial u_S}{\partial t} + ||\nabla_S u_S|| = 0,$$

which we want to achieve for short times by treating the natural corresponding equation in all-space, $u_t + \|\nabla u\| = 0$. If we want both equations to agree on the surface, we need the extended function $E[u_S]$ to have only gradients along the surface directions, not normal to the surface. This means that $E[u_S]$ should be constant along the directions normal to the surface, and thus $E[u_S]$ must be the extension of u_S which is constant moving normal from the surface. While extending values constant normal to the surface may be an intuitively desirable property for the extension operator, this fundamental example shows it is in fact needed to obtain the simplest form of the embedding equation.

Selecting an extension operator which produces a constant normal extension defines the method in general terms. However, one ingredient which is critical to the simplicity of the resulting method is the means of actually constructing the constant normal extension. A particularly simple, accurate and efficient method for constant normal extension is to make use of the "closest point" representation of the surface. For any point x in $\mathbb{R}^3$, let $CP(x)$ denote the closest point to x in the surface S. If S is smooth, this function is well defined and smooth near the surface. (It will generally have discontinuities away from S, at points in space that are equidistant from multiple points on S. Such points do not interfere with the embedding PDE method here, which ultimately relies only on points near S, but they may place an upper limit on grid spacing in $\mathbb{R}^3$, in practice.) The closest point function provides an especially convenient representation of the surface for our purpose because the constant normal extension operator can be defined in terms of it, simply as composition of functions, according to

$$E[u_S](x) = u_S(CP(x))$$

for any u_S defined on S. Thus, what would otherwise be a construction process is reduced to standard evaluation, which greatly streamlines both the practical implementation and the theoretical analysis of the method. As a function, CP is simply a map from $\mathbb{R}^3$ to $\mathbb{R}^3$ that returns values lying in S.

Note that representing the underlying surface using the closest point function gives advantages that extend beyond having a simple, fast and accurate extension process. With a closest point representation we enjoy the flexibility to represent both open and closed surfaces as well as surfaces without an orientation, such as a Möbius strip or a Klein bottle. Also, curves or "filaments"—objects of codimension-two or higher—are naturally accommodated in this representation, as are composite objects such as collections of surfaces, curves and points. Thus, in general, the formalism places no limitations on the topology, geometry or dimensionality of the object on which the PDE is posed.

There are a number of possible techniques to determine the closest point function if it is not already given to us as a part of the problem. In practice, for simple surfaces such as the sphere or torus, the preferred approach is to express the closest point function analytically. This is the approach taken for the circular, spherical and toroidal surfaces considered in our numerical examples. For parameterized surfaces, the closest point function can be accurately computed by standard numerical optimization techniques. See Sections 3.2, 3.3 and 4.4 for examples involving an ellipse and a helix, and [16] for examples involving an ellipsoid and a Möbius strip. Finally, if the surface is in a triangulated form we can either use direct methods, or efficiently determine the closest point at the grid nodes using a variety of sophisticated algorithms, such as the tree-based algorithms of Strain [25] or the public domain closest point transform of Mauch [15]. We remark that the closest point representation for a particular shape need only to be tabulated once, since interpolation can be used to map a well-resolved closest point representation to a desired computational grid. This implies that less efficient (but simple) techniques for computing the closest point will often be effective, such as searching through a list of triangles that define a triangulated surface, and evaluating the distance to each triangle analytically. In any case, the closest point function is ultimately stored on the underlying computational grid used to discretize the PDE of interest, which will be a uniform, Cartesian grid defined in a band around the surface. See Fig. 1 for an example in two dimensions.

2.2. The closest point method

We now present the closest point method for evolving PDEs on surfaces.

To initialize the method we carry out several steps:

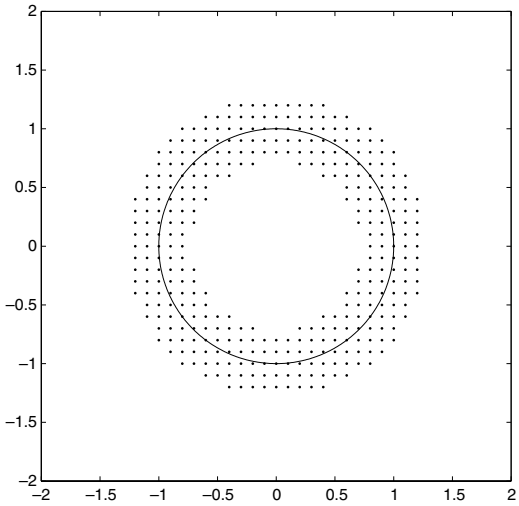

Fig. 1. Underlying computational grid for the unit circle. This example corresponds to a mesh spacing of $\Delta x = 0.1$ and degree-two interpolation polynomials with a five point differencing stencil.

- If it is not already given, a closest point representation of the surface, $CP(x)$, is constructed according to the discussion of Section 2.1.
- A computational domain, Ω_c, is chosen. As is discussed in Section 2.5, the computational domain will typically consist of a band around the surface.
- The embedding of the surface PDE is found in the usual Cartesian coordinates of $\mathbb{R}^3$ by replacing surface gradients by standard gradients in $\mathbb{R}^3$. This leads to a simpler PDE than previous embedding methods [3,4,11] since now there is no projection matrix involved. See Sections 4 and 5 for a variety of examples illustrating this step.
- The solution variable is initialized by extending the initial surface data on to the computational domain using the closest point function.

The closest point method then proceeds by alternating the following two steps:

1. Extend the solution off the surface to the computational domain using the closest point function, i.e., replace u by $u(CP)$ for each grid node on the computational domain.
2. Compute the solution to the embedding PDE using standard finite differences on a Cartesian mesh in the computational domain for one time step. If a Runge–Kutta method is selected for the time evolution, a closest point extension should be carried out after each stage, so that all quantities are evaluated at their closest point values.

At any time step, the approximation of the surface PDE is given by the solution of the embedding PDE at the surface itself.

We remark that the closest point extension is an interpolation step, and the order of the interpolation should be sufficiently high that interpolation errors do not dominate the solution. Thus, for a qth-order differencing scheme and a problem involving up to order-r derivatives the interpolation order should be order $q + r$ (or higher). In our experiments, we often select order $q + r + 1$ to give interpolation errors that are smaller than other effects. Since the PDEs in this paper all give smooth solutions, polynomial interpolation is used throughout the paper. Specifically, interpolation in multi-d is carried out using (one-dimensional) Lagrange interpolating polynomials in a dimension-by-dimension fashion. For nonsmooth solutions, ENO [23] and WENO [13] based interpolation are expected to give better results since they are designed to avoid interpolating across nonsmooth features.

We illustrate the main ideas of the algorithm with two examples. A justification of the method appears in the following section.

Example 1. Consider a general prototype for a PDE describing physical processes on a surface, that allows for reaction, advection and diffusive transport terms, written in nonconservation form as

$$\frac{\partial u_S}{\partial t} = F(x, u_S, \nabla_S u_S, \nabla_S^2 u_S)$$

where ∇_S is the gradient intrinsic to the surface S, and ∇_S^2 is the surface Laplacian, or Laplace–Beltrami operator on the surface. Assuming a forward Euler time discretization, which may be a sub-step in a more accurate time evolution, the corresponding surface evolution problem reads

$$u_S^{n+1} = u_S^n + \Delta t \cdot F(x, u_S^n, \nabla_S u_S^n, \nabla_S^2 u_S^n).$$

We do not treat this surface equation. Instead, we evolve the corresponding equation in the embedding space,

$$u^{n+1} = u^n(CP) + \Delta t \cdot F(CP, u^n(CP), \nabla u^n(CP), \nabla^2 u^n(CP))$$

where ∇ and ∇^2 are the standard Cartesian derivative operators in $\mathbb{R}^3$, which are to be discretized on a regular grid in $\mathbb{R}^3$. Thus to obtain the update of surface function, we simply update the corresponding three-dimensional problem, using entirely standard discrete operators on a regular grid to evaluate the right-hand side. The updated surface function is represented on the grid nodes. Based on these nodal values, an interpolation step is carried out to obtain the values of u at the required surface points. This yields the arguments appearing

in the right-hand side in the subsequent step. Note that equations written in conservation form can be handled in the same manner by applying standard methods for discretizing equations in conservation form.

Example 2. For applications in image processing and geometry, processes defined by contour shortening result in nonlinear, gradient-dependent diffusion equations that can be used for segmentation, noise-removal or computation of geodesics. A suitable prototype for these gradient-dependent diffusion equations posed on a surface is

$$\frac{\partial u_S}{\partial t} = F\left(x, u_S, ||\nabla_S u_S||, \nabla_S \cdot \frac{\nabla_S u_S}{||\nabla_S u_S||}\right).$$

Consider a forward Euler time discretization, which again may be a sub-step in a more accurate time evolution. Then, the surface update equation is formally

$$u_S^{n+1} = u_S^n + \Delta t \cdot F\left(x, u_S^n, ||\nabla_S u_S^n||, \nabla_S \cdot \frac{\nabla_S u_S^n}{||\nabla_S u_S^n||}\right).$$

To solve, we extend the initial conditions on to the computational domain using $u^0 \equiv u_S^0(CP)$ and solve using the standard spatial discretization of the corresponding embedding problem

$$u^{n+1} = u^n(CP) + \Delta t \cdot F\left(CP, u^n, ||\nabla u^n||, \nabla \cdot \frac{\nabla u^n(CP)}{||\nabla u^n(CP)||}\right).$$

Numerical experiments for the special case of curvature-driven motion of curves on surfaces are presented in Section 4.3 (this same application is studied in [6] using level-set methods). Note that general nonlinear, gradient-dependent diffusion terms can be handled in a similar manner, as described in the following section.

2.3. Analysis of the closest point method

We now give an analysis of the closest point method. We will not attempt the rigorous convergence theory, but rather we shall simply show that the method is formally consistent with the original surface PDE.

First, let ∇_S and $\nabla_S \cdot$ denote the intrinsic surface gradient and divergence operators, which are well defined for any quantities defined on the surface. Rather than work with abstract tangent vectors to the surface this section makes use of the corresponding vectors tangent to the surface, as embedded in $\mathbb{R}^3$. Thus, for example, $\nabla_S u(x)$ denotes a vector in $\mathbb{R}^3$ lying in the plane in $\mathbb{R}^3$ that is tangent to S at $x \in S$. Having made this convention, we can proceed with the analysis by stating two fundamental properties

1. Suppose u is any function defined on $\mathbb{R}^3$ that is constant along the directions normal to the surface. Then, at the surface,
$$\nabla u = \nabla_S u.$$
2. For any vector field v on $\mathbb{R}^3$ that is tangent at S, and also tangent at all surfaces displaced by a fixed distance from S (i.e., all surfaces defined as level-sets of the distance function to S), then at the surface
$$\nabla \cdot v = \nabla_S \cdot v.$$

Intuitively, these are quite obvious statements. The first condition says that a function that is constant in the normal direction only varies along the surface, while the second condition says that a flux that is everywhere directed along the surface can only spread out within the surface directions.

Now, in particular, if u is a function defined on the surface, $u(CP)$ is constant along the directions normal to the surface, so the first principle implies

$$\nabla u(CP) = \nabla_S u.$$

Moreover, $\nabla u(CP)$ is always tangent to the level-sets of the distance function, so applying the second principle gives

$$\nabla \cdot (\nabla u(CP)) = \nabla_S \cdot (\nabla_S u)$$

which is the result for the surface Laplacian. More generally, we may consider a surface diffusion operator $\nabla_S \cdot (a(x)\nabla_S u)$, where the diffusion coefficient depends on position. In this case, $v = a(CP)\nabla(u(CP))$ is an extension of $a\nabla_S u$ to $\mathbb{R}^3$ that is tangent to the level surfaces, which implies

$$\nabla_S \cdot (a\nabla_S u) = \nabla \cdot (a(CP)\nabla u(CP)).$$

Generalizing to more complicated diffusion relations, if the coefficient is $a = a(x, u, \nabla_S u)$, it still follows that $v = a(CP, u(CP), \nabla u(CP))\nabla(u(CP))$ is a tangential extension off the surface. Thus,

$$\nabla_S \cdot (a\nabla_S u) = \nabla \cdot (a(CP, u(CP), \nabla u(CP))\nabla u(CP)).$$

These same results would hold if the diffusion "coefficient" a was actually a diffusion matrix, as well.

For the most general diffusion form, suppose $f(x, u, \nabla_S u)$ is an arbitrary "diffusive flux", i.e. a tangent vector field $f(x, u, \nabla_S u)$ that has any functional dependence on position, x, a scalar value, u, and a tangent vector, $\nabla_S u$. Then $v = f(CP, u(CP), \nabla u(CP))$ is an extension of this field that is tangent to all level surfaces, and so again by the general principle we have

$$\nabla_S \cdot (f(x, u, \nabla_S u)) = \nabla \cdot (f(CP, u(CP), \nabla u(CP))) \tag{1}$$

which covers a very broad class of second-order operators, including the level-set equation for curvature motion of contours on the surface and other nonlinear diffusion models described in our second example above. Thus, we find that by extending quantities on to the computational domain and by replacing the first- or second-order surface derivatives by the standard derivatives in $\mathbb{R}^3$ we obtain precisely the desired evolution on the surface itself for a very broad assortment of second-order PDEs.

2.4. Extension to general PDEs

While the method as presented covers a great variety of PDEs of common interest, the basic formalism outlined above is not consistent for general equations with derivatives of more than second-order, or even for the most general possible second-order equation that involves the full second derivative matrix (Hessian). However, for completely general equations, a minor modification given at the end of this section provides a consistent formalism. This comes at the cost of a slightly more complicated procedure, but one that still preserves the essential goal of solving the equation using only the analogous equation on $\mathbb{R}^3$, and discretizations of standard differential operators on uniform grids.

To treat high-order derivatives, one could adopt the approach of other embedding methods [3,11] and introduce an operator which projects on to the tangent plane defined by the local level-set of the signed distance function ϕ, e.g.,

$$P = I - \nabla\phi \otimes \nabla\phi.$$

By inserting sufficient projections P into the derivative expressions we can obtain any desired intrinsic surface derivatives. However, this approach has the disadvantage that it introduces explicit projection operators, as well as derivatives of the projection operator at higher orders. This adds a degree of complication to the approach, and also may result in degenerate equations which are relatively poorly behaved. For example, application of the projection methodology to the surface Laplacian problem leads to a degenerate diffusion equation,

$$\frac{\partial u}{\partial t} = \nabla \cdot (P\nabla u),$$

an equation whose treatment requires more care than a standard diffusion equation [11].

Alternatively, it is possible to handle equations of higher order simply by successively re-extending the gradients themselves. That is, we replace the terms u,$\nabla_S u$, $\nabla_S\nabla_S u$ successively as follows: u is replaced by $u(CP)$, and $\nabla u(CP)$ replaces $\nabla_S u$. This is then extended as $(\nabla u(CP))(CP)$, and $\nabla_S\nabla_S u$ is replaced by $\nabla((\nabla u(CP))(CP))$. Further high derivatives are evaluated by extending a lower derivative (which is defined on all space) using a closest point extension, and then taking the all-space gradient of this extended function. This has the net effect

of trading off all the "outer" application of projections and their derivatives, which is somewhat complicated, for "inner" application of the just the *CP* operator itself, which is trivial. Thus, formally we can embed the higher order operator

$$F(u, \nabla_S u, \nabla_S \nabla_S u, \ldots)$$

as

$$F(u(CP), \nabla u(CP), \nabla((\nabla u(CP))(CP)), \ldots)$$

and otherwise the procedure remains unchanged. Note, however, that we have already seen that a simplifying case arises for powers of the Laplace–Beltrami operator, since the Laplacian is extended correctly by the basic closest point extension. For example, the surface biharmonic operator is given by

$$(\nabla_S^2)^2 u = \nabla^2((\nabla^2 u(CP))(CP))$$

and similarly for higher powers, so that only half as many closest point re-extensions are required for this special, but important, class of operators. The same is true for the more general diffusive flux operator (1) so that we only need to re-extend iterates of this second-order operator, rather than all first-order derivatives involved.

2.5. Banding

Within the class of embedding methods, the closest point method is distinct in a number of respects. For example, it represents the surface using the closest point function. The closest point method also differs in that it makes use of the obvious and familiar Cartesian analog of the underlying surface PDEs instead of using projection operators when forming the embedding PDEs. Another distinction between the closest point method and other embedding methods relates to how the embedding PDEs are used and how this influences the development of practical algorithms based on narrow banding. We now elaborate on this last point.

To obtain efficient algorithms, any embedding method should treat the embedding PDE on a narrow band

$$\Omega_c = \{x : ||x - CP(x)||_2 \leqslant \lambda\}$$

surrounding the surface, where λ is the bandwidth. All previous embedding methods treat an underlying embedding PDE which is defined on all of space and whose solutions, when restricted to the surface, solve the original surface PDE for all times t. Limiting the computation to a narrow band complicates the solution considerably for a number of reasons:

- Solving the embedding PDE on the band requires the imposition of artificial boundary conditions at the boundaries of the computational band since the embedding PDE is defined throughout space and time. The selection of these boundary conditions is not well understood, and may lead to a degradation of the order of accuracy when the bandwidth varies according to the mesh spacing.
- The choice of the bandwidth, λ, is unclear, and is not justified by analytical arguments. Even imperically, the optimal choice of bandwidth remains unclear.
- To improve regularity and limit the effects of the artificial boundaries, methods must be introduced to propagate quantities off the surface on to the band.

The evolution strategy for the closest point method is fundamentally different. The embedding PDE only agrees with the underlying surface PDE when the values off the surface correspond to a constant normal extension of the surface data. Clearly, the necessity of such special data implies that the embedding PDE cannot give a solution to the underlying surface flow for all times, t. On the other hand, the embedding PDE is initialized using a constant normal extension and, according to the analysis of Section 2.3, this implies that the embedding PDE and the surface PDE agree at the surface initially. With explicit time-stepping, this is all that is required for consistency since the subsequent closest point extension step gives a constant normal extension of the surface data that is suitable for the next step of the algorithm. This clear separation of the evolution at the surface, and extension throughout space greatly simplifies narrow banding because it does not introduce

artificial boundaries (while still automatically enforcing the regularity of the solution). Indeed, in a banded calculation with explicit time-stepping, the closest point method gives *exactly* the same result at the surface as an all-space calculation provided the bandwidth is sufficiently wide. We now provide details on selecting such a bandwidth λ.

Suppose that we are working in d-dimensions. Our interpolation will be carried out dimension-by-dimension using one-dimensional Lagrange interpolating polynomials of degree-p with interpolation nodes chosen in the most symmetric way possible around the interpolation point (see, e.g., Fig. 2). Other nonsymmetric interpolation techniques (e.g., ENO and WENO) may also be used and will typically lead to somewhat larger computational bands. To obtain a bound on the bandwidth, we consider an arbitrary point x lying on the surface. Each polynomial interpolation requires that the $(p+1)^d$ nodal values arising in the interpolation stencil have been evolved accurately. Each nodal value will depend on its neighbors, according to the differencing stencil which is used in the PDE evolution step. Calling the set of all such neighbors $\mathcal{N}$, it is clear that a bound on the bandwidth is given by the maximum Euclidean distance from x to a grid node belonging to $\mathcal{N}$.

This calculation is most easily illustrated by an example. Suppose that we work with degree-three interpolation polynomials ($p = 3$) and that the standard five-point Laplacian is used in two-dimensions ($d = 2$). As is shown in Fig. 2, no grid node in the stencil lies more than a distance $\sqrt{2^2 + 2^2}\Delta x$ from x. But the value at this grid node will depend on its four closest neighbors, so the relevant band must also include those points. This leads to a bandwidth of $\sqrt{2^2 + 3^2}\Delta x$. More generally, carrying out this calculation in d-dimensions for the second-order centered difference Laplacian and gradient operators considered in this paper leads to the bandwidth value

$$\lambda = \sqrt{(d-1)\left(\frac{p+1}{2}\right)^2 + \left(1 + \frac{p+1}{2}\right)^2}\,\Delta x.$$

Sharp bounds on the bandwidth for other differencing or interpolation stencils may be determined in a similar manner.

In practice, it is straightforward to code narrow banding. Similar to a global computation, we maintain a uniform mesh of grid nodes in the embedding space to store the extended function and the closest point function. However, an indexing array stores the indices corresponding to all the nodes within the band. Thus, a banded calculation is carried out by limiting calculations to those nodes appearing in the indexing array. Note that in the computations appearing in this paper, the cost of evolving the embedding PDE is less than the interpolation cost, so we may estimate the overall cost per step of the algorithm to be the number of points in the band times the cost of each interpolation. One may carry out the polynomial interpolation dimension-by-dimension using one-dimensional barycentric Lagrange interpolation [2]. This leads to a total cost of $O(p^d)$ operations per interpolation. Alternatively, the well-known Newton divided difference form [5] may be used to

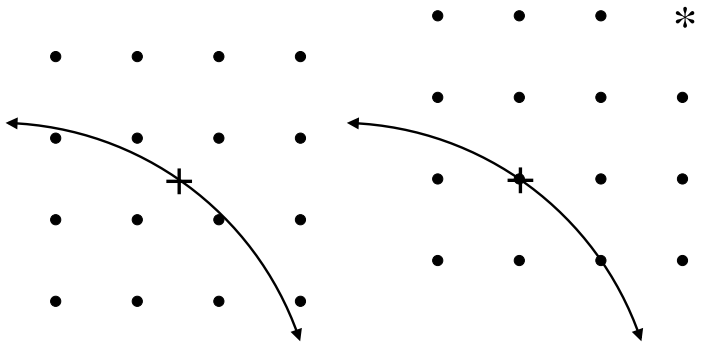

Fig. 2. Interpolation stencil corresponding to degree-three interpolation polynomials for a point "+". Left: From a bandwidth perspective, the grid is optimal for interpolating at the point in question since no stencil node lies more than a distance $1.5\sqrt{2}\Delta x$ from the interpolation point. Right: For this grid, the most symmetrical interpolation stencil will have some grid point "$*$" a distance $2\sqrt{2}\Delta x$ from the interpolation point. The bandwidth will be determined by the distance from the interpolation point to the nearest neighbors of "$*$".

evaluate the interpolating polynomials, but this is more expensive and gives a total of $O(p^{d+1})$ operations per interpolation.

3. Numerical experiments in 2D

We now provide some studies of numerical convergence in two dimensions. In all of our examples analytical solutions are determined from the corresponding one-dimensional periodic systems.

For simplicity, second-order centered differences are used to carry out spatial discretizations and all time-stepping is carried out explicitly (using forward Euler or the third-order TVD Runge–Kutta method). All computations are carried out on a uniform grid defined on the relevant computational band. See Section 2.5 for details on this localization technique.

3.1. Diffusion equation

Consider first diffusion on the unit circle. Following [11], an initial profile

$$u_S(\theta, 0) = \sin(\theta)$$

is assigned, which implies that the solution at any time t is given by

$$u_S(\theta, t) = \exp(-t)\sin(\theta).$$

We apply the closest point method to the problem using an analytical representation of the closest point function. Time-stepping is carried out using forward Euler with a time step-size $\Delta t = 0.1\Delta x^2$ and all interpolations are accomplished with degree-four interpolation polynomials.

The relative errors in the result at the final time $t = 1$ were computed on the circle using the max-norm for a variety of Δx-values. These results are reported in Table 1.

As expected from the order of the spatial discretization, these results give a second-order error in the value of u. We remark that the errors and convergence rates represent an improvement over those reported for a recent embedding method when computing on a band that adapts to the mesh size. See [11] for the results using that level-set based method.

3.2. Advection equation

Our second test evolves a smooth initial profile on the ellipse

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1, \quad a = 0.75, \quad b = 1.25 \tag{2}$$

according to the advection equation,

$$\frac{\partial u_S}{\partial t} + \frac{\partial u_S}{\partial s} = 0,$$

where s is the arclength along the curve.

Table 1
Max-norm relative errors for the heat equation on a circle

Δx	Error	Conv. rate
0.2	1.03e−02	
0.1	2.51e−03	2.04
0.05	6.26e−04	2.00
0.025	1.54e−04	2.02
0.0125	3.84e−05	2.00
0.00625	9.63e−06	2.00

The embedding partial differential equation is

$$\frac{\partial u}{\partial t} + T(x,y)\cdot\nabla u = 0,$$

where the velocity

$$T(x,y) = V(CP(x,y)) \tag{3}$$

is an extension of the value defined on the ellipse itself

$$V((x,y)) \equiv \frac{(-\frac{y}{b^2},\frac{x}{a^2})}{\sqrt{\frac{y^2}{b^4}+\frac{x^2}{a^4}}}.$$

The closest point representation of the ellipse is precomputed on the underlying grid (to double precision) using Newton's method applied to the derivative of the square of the distance function.

We consider the initial profile

$$u_S(s,0) = \cos^2(2\pi s/L),$$

where s is arclength and $L \approx 6.38174971583483$ is the perimeter of the ellipse (cf. [11]). Our computations measure the max-norm of the difference between our computed solution and the exact solution,

$$u_S(s,t) = \cos^2(2\pi(s-t)/L)$$

for several different grid spacings to estimate the convergence rate.

Because second-order central differences are used, we must use a time-stepping scheme that includes the imaginary axis near the origin to achieve linear stability. For this reason, we use the popular third-order TVD Runge–Kutta scheme [23,22]. In all calculations, the time step-size is set according to $\Delta t = 0.5\Delta x$ and we use third-order interpolation polynomials to carry out interpolations. Note that simple second-order central differences are effective for this smooth problem. If upwinding is required for the embedding PDE, however, it may make sense to also use an interpolation step which is nonsymmetric. An example of such an interpolation appears in [14], where a WENO-based interpolation procedure and standard WENO methods for Hamilton–Jacobi PDEs are used to solve level set equations on surfaces.

The relative errors at time $t = 1$ for a number of experiments are reported in Table 2, below.

These results clearly demonstrate second-order convergence in the value of u. Thus, both the closest point method and the recent level-set method of Greer [11] give second-order convergence when the bandwidth is adapted to the mesh-size. A direct comparison of the errors cannot be made in this example since we have treated an ellipse rather than a circle.

3.3. Advection–diffusion equation

We conclude our examples in $\mathbb{R}^2$ with the evolution of a smooth initial profile on the ellipse (2) according to the advection–diffusion equation,

$$\frac{\partial u_S}{\partial t} + \frac{\partial u_S}{\partial s} = \frac{\partial^2 u_S}{\partial s^2},$$

where s is the arclength along the curve.

Table 2
Max-norm relative errors for the advection equation on an ellipse

Δx	Error	Conv. rate
0.2	3.88e−02	
0.1	6.89e−03	2.49
0.05	1.72e−04	2.00
0.025	4.30e−04	2.00
0.0125	1.07e−04	2.00
0.00625	2.68e−05	2.00

The embedding partial differential equation is

$$\frac{\partial u}{\partial t} + T(x,y)\cdot\nabla u = \nabla^2 u$$

with velocity $T(x, y)$ as defined in (3). In this example, we consider the initial profile

$$u_S(s,0) = \sin^2(2\pi s/L),$$

where s is arclength and L is the perimeter of the ellipse (cf. [11]). Our computations measure the max-norm of the difference between our computed solution and the exact solution,

$$u_S(s,t) = \exp(-4t)\sin^2(4\pi(s-t)/L)$$

for several different grid spacings to estimate the convergence rate. To ensure stability, time-stepping is carried out using forward Euler with a time step-size $\Delta t = 0.1\Delta x^2$. All interpolations are carried out using degree-four polynomials.

The relative errors arising at time $t = 1$ for a number of experiments are reported in Table 3.

This convergence test also indicates a second-order convergence in the value of u. Similar to the case of diffusion, the convergence rates represent an improvement over those reported for a recent level-set based method [11] when computing on a band that adapts to the mesh size.

4. Numerical experiments in 3D

We now examine the numerical behavior of the method in three dimensions. Where analytical solutions exist, numerical convergence studies are carried out.

Similar to the previous section, second-order centered differences are used to carry out spatial discretizations and all time-stepping is carried out explicitly (using forward Euler or the third-order TVD Runge–Kutta method). All computations are carried out on a band around the surface.

4.1. Diffusion equation

Consider diffusion on the unit sphere. Following [11], we assign initial conditions

$$u_S(\theta,\eta,0) = \cos(\eta)$$

in spherical coordinates (r, θ, η). As pointed out in [11], the corresponding initial value problem has exact solution

$$u_S(\theta,\eta,t) = \exp(-2t)\cos(\eta).$$

The evolution is carried out using an analytical representation for the closest point function. Time-stepping is carried out using forward Euler with a time step-size $\Delta t = 0.1\Delta x^2$ and degree-four polynomials are used to carry out interpolations. Calculating the max-norm relative error of the numerical result at the final time $t = 1$ for several Δx-values gives the results reported in Table 4.

Table 3
Max-norm relative errors for the advection–diffusion equation on an ellipse

Δx	Error	Conv. rate
0.1	4.72e−02	
0.05	3.85e−03	3.62
0.025	1.09e−03	1.82
0.0125	2.96e−04	1.88
0.00625	7.48e−05	1.98
0.003125	1.88e−05	2.00
0.0015625	4.69e−06	2.00

Table 4
Max-norm relative errors for the heat equation on a sphere

Δx	Error	Conv. rate
0.2	7.49e−03	
0.1	2.14e−03	1.81
0.05	5.19e−04	2.05
0.025	1.30e−04	2.00
0.0125	4.38e−05	2.00

This convergence test also indicates a second-order convergence in the value of u. Similar to the two-dimensional case, the errors and convergence rates represent an improvement over those reported for a recent level-set based method applied to a band that adapts to the mesh size [11].

4.2. Advection equation

To study the numerical convergence for advection on a surface, we consider the example provided in [11]. Specifically, we evolve on a torus defined by

$$\left(\left(\frac{1}{2}\cos(\eta)+1\right)\cos(\theta),\left(\frac{1}{2}\cos(\eta)+1\right)\sin(\theta),\frac{1}{2}\sin(\eta)\right),\quad -\pi\leqslant\theta,\quad \eta<\pi$$

according to the advection equation $\frac{\partial u_S}{\partial t}+\frac{\partial u_S}{\partial \eta}=0$. The initial conditions are set equal to

$$u_S(\theta,\eta,0)=f(\eta),$$

where f is the smooth function of period 2π defined by

$$f(\eta)=\begin{cases} g\left(\frac{\eta+\pi}{\pi}\right) & -\pi\leqslant\eta\leqslant 0\\ g\left(\frac{\eta-\pi}{\pi}\right) & 0<\eta<\pi\end{cases}\quad \text{with } g(x)=\frac{\exp\left(\frac{1}{x-1}\right)-\exp\left(-\frac{1}{x}\right)}{\exp\left(-\frac{1}{x}\right)+\exp\left(\frac{1}{x-1}\right)}.$$

Computing using second-order centered differences, the third-order TVD Runge–Kutta scheme and degree-three interpolation polynomials with a time step-size $\Delta t=0.5\Delta x$ and varying Δx gives us a sequence of solutions. The corresponding max-norm relative errors at time $t=1$ are reported in Table 5.

This convergence test also indicates a second-order convergence in the value of u. Thus, both the closest point method and the recent level set method of Greer [11] give second-order when the bandwidth is adapted to the mesh-size. The errors generated by both methods for a particular mesh size are also very similar.

4.3. Curvature motion

Consider next the motion of a circular interface on a sphere evolving according to in-surface curvature motion. By symmetry, the interface remains a circle as it collapses. Moreover, it is straightforward to determine the state of the system at any time t since the radius of the collapsing circle is governed by an ODE system which can be solved to high precision with standard ODE methods.

Table 5
Max-norm relative errors for the advection equation on a torus

Δx	Error	Conv. rate
0.1	3.82e−02	
0.05	1.00e−02	1.93
0.025	2.44e−03	2.04
0.0125	6.57e−04	1.89
0.00625	1.62e−04	2.02

We represent the evolving contour by the zero level of the function ϕ which is initially set equal to $(2/3) - \sqrt{y^2 + z^2}$. To achieve curvature motion, we wish to evolve ϕ according to the embedding of the surface PDE,

$$\frac{\partial \phi_S}{\partial t} - \left(\nabla_S \cdot \frac{\nabla_S \phi_S}{||\nabla_S \phi_S||}\right) ||\nabla_S \phi_S|| = 0.$$

In our formulation, we therefore proceed simply by evolving the three-dimensional level-set equation,

$$\frac{\partial \phi}{\partial t} - \left(\nabla \cdot \frac{\nabla \phi}{||\nabla \phi||}\right) ||\nabla \phi|| = 0.$$

where $\kappa = (\nabla \cdot \frac{\nabla \phi}{||\nabla \phi||})$ is the mean curvature of the local level-set in $\mathbb{R}^3$. Similar to our other examples, the required initial conditions are obtained by extending the initial conditions on to the computational band using the closest point extension.

We select the sphere radius to be 1 and the initial radius of the circle to be 2/3. To discretize in time, forward Euler is used with a time discretization parameter $\Delta t = 0.1\Delta x^2$. Degree-four polynomials are used to carry out all interpolations. Convergence is studied by comparing the numerical radius at time $t = 0.1$ against the exact result (0.56695935668549) for various Δt. The relative errors in the final radius from a number of experiments are reported in Table 6.

This convergence test indicates a second-order convergence in the value of the circle radius.

4.4. Diffusion on a filament

Our final convergence test considers diffusion of a smooth initial profile on a helical curve

$$(x, y, z) = (\sin(2\pi s), \cos(2\pi s), 2s - 1) \tag{4}$$

where $0 \leqslant s \leqslant 1$ and homogeneous Neumann and Dirichlet conditions are imposed at the endpoints $s = 0$ and $s = 1$, respectively. Note this example involves a codimensional-two object with boundaries, so it is much more naturally treated using a closest point representation than a level-set representation. The closest point representation of the helix is precomputed on the underlying grid (to double precision) using Newton's method applied to the derivative of the square of the distance function.

To examine the numerical convergence, we consider the initial profile

$$u_S(s, 0) = \cos(0.5\pi s).$$

Our computations measure the max-norm of the difference between our computed solution and the exact solution,

$$u_S(s, t) = \exp\left(-\left(\frac{\pi}{2L}\right)^2 t\right) \cos(0.5\pi s)$$

where $L = \sqrt{1 + \pi^2}$ is the length of the helix.

No special treatment is required at the homogeneous Neumann boundary since the value at that endpoint is naturally extended throughout space at each step using the closest point function. At the homogeneous Dirichlet condition the treatment is also straightforward: instead of propagating out the numerical value at that endpoint, we propagate out the prescribed boundary value (in this case $u_S(1, t) = 0$).

Table 6
Max-norm relative errors for curvature motion on the sphere

Δx	Error	Conv. rate
0.05	3.35e−04	
0.025	8.22e−05	2.03
0.0125	2.05e−05	2.01
0.00625	5.11e−06	2.00

Table 7
Max-norm relative errors for diffusion on a helix with boundary conditions

Δx	Error	Conv. rate
0.2	1.53e−02	
0.1	7.64e−03	1.01
0.05	3.82e−03	1.00
0.025	1.91e−03	1.00
0.0125	9.54e−04	1.00

Time stepping is carried out to time $t = 1$ using forward Euler with $\Delta t = 0.1\Delta x^2$. Evaluating the max-norm relative errors for several different grid spacings gives the numerical convergence rate results presented in Table 7. Our interpolations are carried out using degree-three polynomials since higher orders did not significantly influence the errors.

As we can see from the table, the introduction of simple boundary conditions gives a consistent calculation, however, the result is only first-order accurate. When boundary conditions are enforced in this manner, the extended function is continuous but certain derivatives may be discontinuous near the boundaries. (In this example, discontinuities in the first derivatives occur on a planar region which is orthogonal to the filament at the Dirichlet boundary, $s = 1$. Discontinuities in the second derivatives occur on a planar region which is orthogonal to the filament at the homogeneous Neumann boundary, $s = 0$.) This loss of regularity contributes to a loss of accuracy in the interpolation procedure and the discretization of the embedding PDE.

The development of methods that gives an improved treatment of boundary conditions is of strong interest to us and is a part of our ongoing investigations.

4.5. Reaction diffusion systems

We conclude our numerical experiments by providing some applications to reaction diffusion systems.

Fig. 3 gives an example of a spiral wave evolving on a sphere, as computed by our approach. The simulated system in this example is the well-known Fitzhugh–Nagumo equations [9]

$$\frac{\partial u_S}{\partial t} = (a - u_S)(u_S - 1)u_S - v_S + v\nabla_S^2 u_S, \tag{5}$$

$$\frac{\partial v_S}{\partial t} = \epsilon(\beta u_S - v_S), \tag{6}$$

where u_S is the excitation variable, $\epsilon = 0.01$, $a = 0.1$, $\beta = 0.5$ and $v = 0.0001$. To obtain an attractive spiral wave, we set our initial conditions according to

$$(u_S, v_S) = \begin{cases} (1,0) & \text{if } x > 0, y > 0, z > 0, \\ (0,1) & \text{if } x < 0, y > 0, z > 0, \\ (0,0) & \text{otherwise.} \end{cases}$$

This simulation set $\Delta t = 0.0390625$ and $\Delta x = 0.00625$. Doubling these discretization step sizes gave very similar results.

Fig. 4 gives an example of a Turing pattern formation model [26] evolving on the surface of a U-shaped tube, as computed by our approach. The simulated system in this example is the Schnakenberg system [21],

$$\frac{\partial u_S}{\partial t} = \gamma(a - u_S + u_S^2 v_S) + \nabla_S^2 u_S, \tag{7}$$

$$\frac{\partial v_S}{\partial t} = \gamma(b - u_S^2 v_S) + v\nabla_S^2 v_S, \tag{8}$$

where u_S is the activator and v_S is the inhibitor. To perturb the system away from equilibrium, we set the initial conditions at each point (x, y, z) on the surface according to

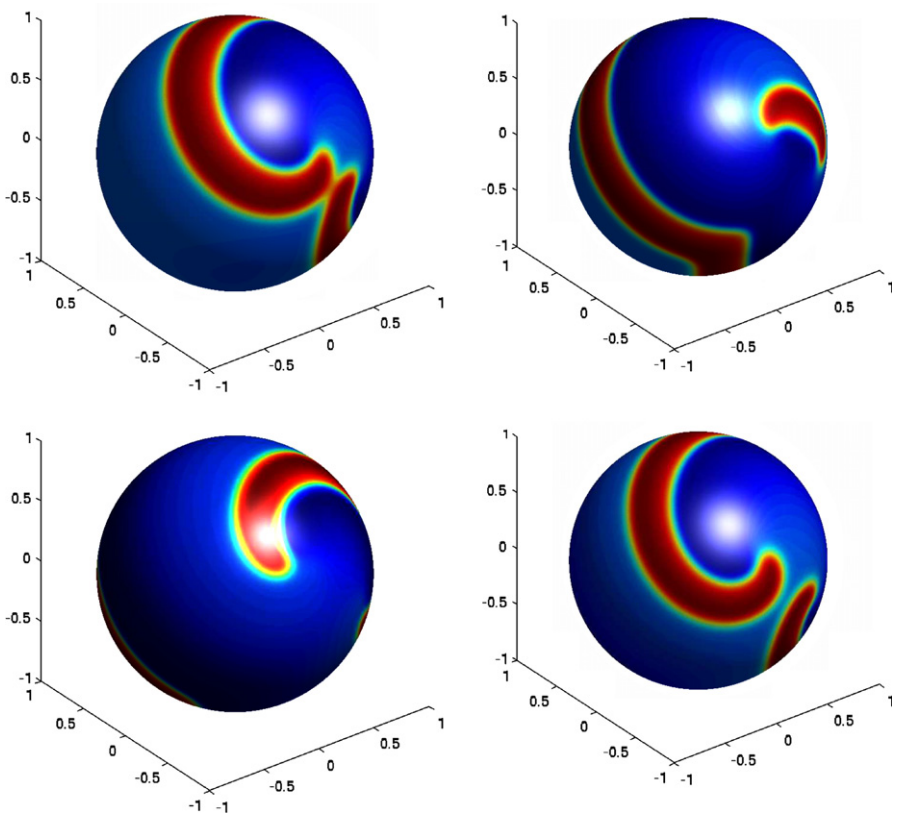

Fig. 3. Fitzhugh–Nagumo equation evolving on a sphere. The excitation variable u is displayed at times $t = 375$, 437.5, 500 and 562.5. This simulation takes $\Delta t = 0.0390625$, $\Delta x = 0.00625$ and uses an analytical representation of the closest point to the sphere. Forward Euler time-stepping and degree-four interpolating polynomials are used throughout the calculation.

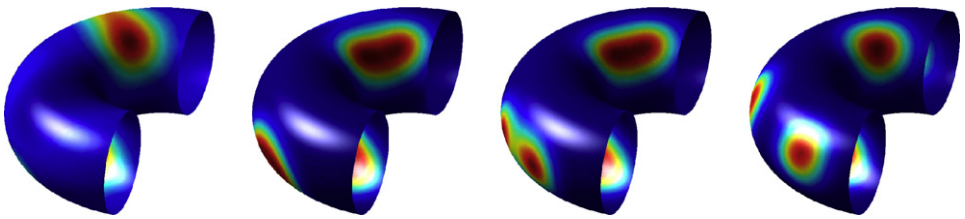

Fig. 4. Schnakenberg system evolving on a U-shaped tube. The activator u is displayed at times $t = 0.01$, 0.03, 0.05, 0.13. This simulation takes $\Delta t = 1.5625\text{e}{-07}$, $\Delta x = 1/160$ and uses an analytical representation of the closest point to the tube. Forward Euler time-stepping and degree-three interpolating polynomials are used throughout the calculation.

$$u_S(x,y,z) = a + b + \sum_{i=1}^{5} \frac{1}{20i} \sin(2\pi i x)\sin(2\pi i y)\sin(2\pi i z),$$

$$v_S(x,y,z) = \frac{a}{(a+b)^2} + \sum_{i=1}^{5} \frac{1}{20i} \cos(2\pi i x)\cos(2\pi i y)\cos(2\pi i z).$$

In this calculation steady patterns are sought, so the free parameters are set according to some values appearing in [19]: $\gamma = 500$, $a = -.048113$, $b = 1.202813$ and $v = 120$. This simulation took $\Delta t = 1.5625\text{e}{-07}$ and $\Delta x = 1/160$, and we note that doubling these discretization step sizes produced similar results. Because this problem is numerically stiff, implicit time-stepping would be highly desirable. A focus of our current work is the design of efficient methods for treating implicit time-discretizations using the closest point method.

5. Summary and future work

In this work, we present the closest point method, which is a new embedding method for solving partial differential equations on surfaces. The method is designed to make solving PDEs on surfaces as close as possible to the familiar process of solving PDEs in $\mathbb{R}^3$, in effect hiding all the geometric complexities. Central to the method is the choice of the closest point representation of the surface. This representation naturally gives an extension step which leads to embedding PDEs that are simply the surface PDEs with surface gradients replaced by standard gradients in $\mathbb{R}^3$. This representation also gives the flexibility to treat open surfaces, surfaces without orientations, objects of codimension-two or higher or collections of objects with varying codimension. We further remark that it is straightforward to compute on a narrow band around the surface using the closest point method and that (unlike other embedding methods) banding can be carried out without any degradation in the accuracy of the underlying discretization. The net result is that the method is remarkably simple: instead of treating a surface PDE it treats the corresponding PDE in the embedding space using standard numerical methods on uniform Cartesian grids. This means that existing software for three-dimensional flows can be modified to carry out surface flows with the minimal programming effort.

A variety of numerical experiments were carried out to validate the method. It was found that the numerical convergence rates agreed with the convergence rates of the underlying spatial discretizations. The errors produced by the method for a given mesh width were similar or better than those reported for a recent embedding method [11].

There are many opportunities for further development of the closest point method. In particular, we are investigating the use of ENO and WENO based interpolation to give methods suitable for nonsmooth flows (e.g. [14]). The generalization of the method to implicit time-stepping and elliptic equations on surfaces is also of strong interest to us, since many flows are too stiff to be conveniently treated by explicit methods (e.g. [12]). Formally, the analysis developed in Section 2.3 still applies to such equations, however, the solution of the corresponding nonlinear equations will be complicated by the closest point operator. Another area of research interest is the improved treatment of boundary conditions for open surfaces. In this paper, first-order accuracy was obtained using a straightforward extension of boundary data but we anticipate more sophisticated methods should yield more accurate results.

The study of more general flows or flows on more general objects is also of interest. For example, the treatment of third- and higher-order PDEs should be further investigated. While such applications can be solved using multiple applications of the closest point mapping, we have not yet investigated this class of problems numerically. Other potential targets for future work include solving PDEs on surfaces with edges/corners, or even point clouds of data, as the general closest point method formalism still applies even when the underlying notion of a PDE on the object is no longer clearly defined.

While this paper focuses on the important case of static surfaces, the treatment of moving surfaces also appears in applications (e.g., [29]) and is another interesting topic for investigation. In such applications, it is important to note an advantage that level-set representations have over closest point representations: level-set representations of surfaces can be evolved using well-known and robust discretizations of the level-set equation, whereas the evolution of closest point representations is less well understood (cf. [24,20]). This advantage is particularly pronounced in flows involving surfaces that merge or break apart, since level-set methods treat such problems naturally and automatically.

Finally, we conclude by noting that the closest point method's simplicity and flexibility make it an excellent candidate for treating areas of application in the natural and applied sciences. See [27,28,7,8,17,18] and the references in [3,11] for a sample of some application areas that treat PDEs on surfaces.

Appendix A. Calculation of surface normal and curvature

The surface normal and curvature are fundamental geometric properties of the surface, and are of interest for a variety of purposes. In the context of solving surface PDEs, our main concern is that such quantities could occur explicitly within the PDE, for example, in the form of a curvature-dependent reaction rate in a reaction diffusion equation.

If these geometric quantities are available as a given function on the surface, our general formalism would immediately apply. However, it is more likely that they will actually need to be computed from the surface itself, prior to any extension. Thus, we provide a convenient way to compute these quantities directly from the closest point representation of the surface, $CP(x)$.

First, consider constructing a normal vector field, N. In terms of the closest point function, this is quite simple. Given any point x in $\mathbb{R}^3$, the vector $x - CP(x)$ extends from the surface at $CP(x)$ to the point x, in a direction normal to the surface. Thus, normalizing this vector field

$$N(x) = \frac{x - CP(x)}{||x - CP(x)||}$$

provides a suitable extension of the normal *where the direction points away from the surface*. In some applications, the fact that the vector field N changes direction discontinuously at the surface may preclude the immediate use of this formula. For example, the discontinuity at the surface is undesirable in the curvature formulas below. For such situations, we need to make a choice of one of the two available directions at the surface as the preferred direction, and reverse the direction of N on the other side. This choice can be encapsulated in a sign function $s(x)$ defined near the surface that is $+1$ on one side of surface, and -1 on the other. This is in effect a choice of "outward normal direction", or orientation, for the surface, which is always needed to define the overall sign of the curvature, independent from the issues at hand. Given such a sign function, then the vector field

$$n(x) = s(x)N(x) \tag{9}$$

is a suitable extension of a unit normal field on the surface to all of space. Note this is not strictly defined *at* the surface, since $N(x)$ is not defined there. At such points, however, $N(x)$ can be assigned its limiting value, approaching from off the surface.

The sign function $s(x)$ must somehow be constructed independently, as it is not determined by the surface itself, or the closest point function. For example, if the surface is closed, with a well-defined inside region, the indicator function of this inner region can be used to define s. Or, if a signed distance function is available, the sign from that can be adopted for s.

For general curvature flows we may compute curvature from an extension of the unit normal vector field. Specifically, let n be a unit vector field defined on $\mathbb{R}^3$ that reduces to a unit normal vector field along the surface. Then the *mean curvature* of the surface, κ, is given by the divergence of this vector field

$$\kappa = \nabla \cdot n$$

at the surface, and this relation provides a convenient embedding of the mean curvature off the surface as well. More generally, all properties of the curvature can be obtained from the curvature matrix K, which can be computed as the total derivative of the normal vector field

$$K = \nabla n$$

at the surface, and similarly using this formula to extend K off the surface. The matrix K has n itself as a trivial eigenvector, with eigenvalue 0, reflecting the constant length of n. The nontrivial eigenvectors of this matrix are tangent to the surface and define the directions of principal (maximal and minimal) curvature, and the corresponding eigenvalues κ_1 and κ_2 are the principal curvatures at the point in question. The mean curvature is the sum of these principal curvatures, or, equivalently, the trace of the matrix K, which yields the divergence formula given previously.

Alternatively, certain curvature-dependent quantities can be obtained from the closest point function itself. For example, it is easily shown [20] that at the surface the *vector mean curvature* is given by the Laplacian of the closest point function, i.e.,

$$-\kappa n = \nabla^2 CP$$

for any point on the surface S. This vector-valued quantity corresponds to the velocity of the surface under gradient descent on the surface area, or equivalently, it is the first variational derivative of the surface area, and thus is a quantity of particular importance. From this, the mean curvature can be determined by

$$\kappa = -\left(\nabla^2 CP\right) \cdot n,$$

where n is a given choice of the unit normal.

References

[1] M. Berger, Finite Element Approximation of Elliptic Partial Differential Equations on Implicit Surfaces, CAM Report 05-46, University of California, Los Angeles, 2005.
[2] J.-P. Berrut, L.N. Trefethen, Barycentric Lagrange interpolation, SIAM Review 46 (3) (2004) 501–517.
[3] M. Bertalmío, L.T. Cheng, S. Osher, G. Sapiro, Variational problems and partial differential equations on implicit surfaces, Journal of Computational Physics 174 (2001) 759–780.
[4] M. Bertalmío, F. Memoli, L.T. Cheng, G. Sapiro, S. Osher, Variational problems and partial differential equations on implicit surfaces: bye bye triangulated surfaces? in: S. Osher, N. Paragios (Eds.), Geometric Level Set Methods in Imaging, Vision, and Graphics, Springer, New York, 2003, pp. 381–398.
[5] Richard L. Burden, J. Douglas Faires, Numerical Analysis, seventh ed., Brooks/Cole, 2001.
[6] L.-T. Cheng, P. Burchard, B. Merriman, S. Osher, Motion of curves constrained on surfaces using a level-set approach, J. Comput. Phys. 175 (2) (2002) 602–644.
[7] U. Diewald, T. Preufer, M. Rumpf, Anisotropic diffusion in vector field visualization on Euclidean domains and surfaces, IEEE Tran. Vis. Comput Graph. 6 (2000) 139–149.
[8] J. Dorsey, P. Hanrahan, Digital materials and virtual weathering, Sci. Am. 282 (2) (2000) 282–289.
[9] R. FitzHugh, Fitzhugh–Nagumo simplified cardiac action potential model, Biophys. J. 1 (1961) 445–466.
[10] M.S. Floater, K. Hormann, Surface parameterization: a tutorial and survey, in: N.A. Dodgson, M.S. Floater, M.A. Sabin (Eds.), Advances in Multiresolution for Geometric Modelling, Heidelberg, 2005, pp. 157–186.
[11] J.B. Greer, An improvement of a recent Eulerian method for solving PDEs on general geometries, J. Sci. Comput. 29 (3) (2006) 321–352.
[12] J.B. Greer, A.L. Bertozzi, G. Sapiro, Fourth order partial differential equations on general geometries, J. Comput. Phys. 216 (1) (2006) 216–246.
[13] G.-S. Jiang, C.-W. Shu, Efficient implementation of weighted ENO schemes, J. Comput. Phys. 126 (1996) 202–228.
[14] C.B. Macdonald, S.J. Ruuth, Level set equations on surfaces via the closest point method, submitted for publication.
[15] S. Mauch, Efficient Algorithms for Solving Static Hamilton Jacobi Equations, PhD Thesis, California Institute of Technology, Pasedena, 2003.
[16] B. Merriman, S.J. Ruuth, Diffusion generated motion of curves on surfaces, J. Comput. Phys. 225 (2) (2007) 2267–2282.
[17] T.G. Myers, J.P.F. Charpin, A mathematical model for atmospheric ice accretion and water flow on a cold surface, Int. J. Heat Mass Transf. 47 (25) (2004) 5483–5500.
[18] T.G. Myers, J.P.F. Charpin, S.J. Chapman, The flow and solidification of a thin fluid film on an arbitrary three-dimensional surface, Phys. Fluids 14 (8) (2002) 2788–2803.
[19] S.J. Ruuth, Implicit–explicit methods for reaction–diffusion problems in pattern-formation, J. Math. Biol. 34 (2) (1995) 148–176.
[20] S.J. Ruuth, B. Merriman, S. Osher, A fixed grid method for capturing the motion of self-intersecting interfaces and related PDEs, J. Comput. Phys. 163 (2000) 1–21.
[21] J. Schnakenberg, Simple chemical-reaction systems with limit-cycle behavior, J. Theor. Biol. 81 (3) (1979) 389–400.
[22] Chi-Wang. Shu, Total-variation-diminishing time discretizations, SIAM J. Sci. Statist. Comput. 9 (6) (1988) 1073–1084.
[23] Chi-Wang. Shu, Stanley. Osher, Efficient implementation of essentially nonoscillatory shock-capturing schemes, J. Comput. Phys. 77 (2) (1988) 439–471.
[24] J. Steinhoff, M. Fan, L. Wang, A new Eulerian method for the computation of propagating short acoustic and electromagnetic pulses, J. Comput. Phys. 157 (2) (2000) 683–706.
[25] J. Strain, Fast tree-based redistancing for level set computations, J. Comput. Phys. 152 (1999) 648–666.
[26] A.M. Turing, The chemical basis of morphogenesis, Roy. Soc. Lond. Philos. Trans. Ser. B 237 (1952) 37–72.
[27] G. Turk, Generating textures on arbitrary surfaces using reaction–diffusion, Comput. Graph. 25 (4) (1991) 289–298.
[28] A. Witkin, M. Kass, Reaction–diffusion textures, Comput. Graph. 25 (4) (1991) 299–308.
[29] J. Xu, H.-K. Zhao, An Eulerian formulation for solving partial differential equations along a moving interface, J. Sci. Comput. 19 (1–3) (2003) 573–594.