Fourth Order Partial Differential Equations on General Geometries[*]

John B. Greer [†] **Andrea L. Bertozzi** [‡] **Guillermo Sapiro**[§]

Abstract

We extend a recently introduced method for numerically solving partial differential equations on implicit surfaces (Bertalmío, Cheng, Osher, and Sapiro 2001) to fourth order PDEs including the Cahn-Hilliard equation and a lubrication model for curved surfaces. By representing a surface in $\mathbb{R}^N$ as the level set of a smooth function, ϕ, we compute the PDE using only finite differences on a standard Cartesian mesh in $\mathbb{R}^N$. The higher order equations introduce a number of challenges that are of small concern when applying this method to first and second order PDEs. Many of these problems, such as time-stepping restrictions and large stencil sizes, are shared by standard fourth order equations in Euclidean domains, but others are caused by the extreme degeneracy of the PDEs that result from this method and the general geometry. We approach these difficulties by applying convexity splitting methods, ADI schemes, and iterative solvers. We discuss in detail the differences between computing these fourth order equations and computing the first and second order PDEs considered in earlier work. We explicitly derive schemes for the linear fourth order diffusion, the Cahn-Hilliard equation for phase transition in a binary alloy, and surface tension driven flows on complex geometries. Numerical examples validating our methods are presented for these flows for data on general surfaces.

1 Introduction

Partial differential equations (PDEs) defined on surfaces embedded in $\mathbb{R}^3$ arise in a wide range of applications, including fluid dynamics, biology (e.g., fluids on the lungs), materials science (e.g., ice formation), electromagnetism, image processing (e.g., images on manifolds and inverse problems such as EEG), computer graphics (e.g., water flowing on a surface), computer aided geometric design (e.g., special curves on surfaces), and pattern formation. The work in this paper is concerned with fourth order differential equations, which have interests in all the areas mentioned above. Examples of physical flows modeled by fourth order PDEs include ice formation [36, 37], fluids on lungs [21], brain warping [33, 53], and designing special curves on surfaces [22, 33]. In this paper we extend the work in [6] to these high order flows. We represent the surface with arbitrary geometry implicitly, as the level set of a smooth function defined in all of the embedding space $\mathbb{R}^3$, and rewrite the relevant equations in terms of Euclidean coordinates and derivatives of the level set function (see Section 2). This method has been used for solving first and second order equations on static, [6, 27, 31, 32], as well as dynamic, [2, 57], surfaces. In [6], the authors introduced the general framework and showed how to solve second order linear and nonlinear diffusions and reaction-diffusion equations on implicitly defined surfaces. In [27, 31], the authors solved the Eikonal equation on surfaces like those in [6] (while in the first paper the work is for triangulated surfaces, in the second implicit representations are used). In these works, static surfaces were considered. The authors of

*Work supported by the National Science Foundation and the Office of Naval Research.

†Courant Institute for Mathematical Sciences, New York University, New York, 10012, greer@cims.nyu.edu

‡Department of Mathematics, UCLA, Los Angeles, CA 90095, bertozzi@math.ucla.edu

§Electrical and Computer Engineering, University of Minnesota, Minneapolis, MN 55455, guille@ece.umn.edu

[2] and [57] solve second order diffusion equations on interfaces that are deforming subject to an extrinsic flow. As discussed in the above papers in detail, implicit representations provide a natural means for addressing these flows on arbitrary surfaces.

Solving PDEs and variational problems with polynomial meshes involves the non-trivial discretization of the equations in general polygonal grids, as well as the difficult numerical computation of other quantities like projections onto the discretized surface (when computing gradients and Laplacians for example). Although the use of triangulated surfaces is quite popular, there is still no consensus on how to compute simple differential characteristics such as tangents, normals, principal directions, and curvatures. On the other hand, it is commonly accepted that computing these objects for iso-surfaces (implicit representations) is simpler and more accurate and robust. This problem becomes even more significant when we not only have to compute these first and second order differential characteristics of the surface, but also have to use them to solve variational problems and PDEs for data defined on the surface. Formal analysis of finite difference schemes on non-Cartesian meshes is a new area [3, 9, 10, 46], whereas finite difference schemes on Cartesian meshes are better understood. Note also that working with polygonal representations is dimensionality dependent, and solving these equations for high dimensional (> 2) surfaces becomes even more challenging and significantly less studied. The work here developed is valid for all dimensions of interest. The computational cost of working with implicit representations is not higher than with meshes, since all the work is performed in a narrow band around the level-set(s) of interest.

The framework of implicit representations for solving PDEs on them, as introduced in [6, 32], enables us to perform all the computations on the Cartesian grid corresponding to the embedding function. These computations are, nevertheless, intrinsic to the surface. Advantages of using Cartesian grid instead of a triangulated mesh include the availability of well studied numerical techniques (that we will extend in this paper, see below) with accurate error measures, and the topological flexibility of the surface, all leading to simple, accurate, robust and elegant implementations. The approach is general (applicable to PDEs and variational problems beyond those derived in this paper) and dimensionality independent as well. We should note of course that the computational framework here developed is only valid for manifolds which can be represented in implicit form or as intersection of implicit forms. As mentioned above, several disciplines of sciences have numerous problems that can be embedded within the implicit framework.

In a number of applications, surfaces are already given in implicit form, see for example [43, 49], therefore, the framework in this paper is not only simple and robust, but it is also natural in those applications. Moreover, in the state-of-the-art and most commonly used packages to obtain 3D models from range data and from segmented volumetric medical images, the algorithms output an implicit (e.g., distance) function (see for example graphics.stanford.edu/projects/mich/ and www.itk.org), and it is important to develop computational frameworks where the surface representation is dictated by the data and application, and not the other way around. On the other hand, not all surfaces (manifolds) are originally represented in implicit form. For generic surfaces, we need to apply an algorithm that transforms the given explicit representation into an implicit one. Although this is still a very active area of research, many very good algorithms have been developed; see for example [17, 23, 29, 59]. This translation needs to be done only once for any surface. For rendering, the volumetric data can be used directly, without the need for an intermediate mesh representation.

Once the surface is in implicit form, using the results and the basic “dictionary” provided in [6, 32], we can translate PDEs and variational problems based on intrinsic characteristics of the manifold, into PDEs and variational problems that depend on the implicit manifold and the embedding space. This translation is done in a systematic and generic fashion.

In this paper we consider fourth order equations on static surfaces. Although future work will undoubtedly extend these methods to solve related equations on dynamically changing surfaces, the jump from second (as in [6, 32]) to fourth order equations on static surfaces is a significant computational challenge unto itself. The computation of fourth order diffusions in flat Euclidean space is far less understood than computation of second order diffusions. The literature on numerics for fourth order PDEs is an active area of research [5, 11, 19, 20, 51, 56, 60]. Some work on solving equations with fourth order diffusions on surfaces has been done for particular examples (see [37]), but methods for solving general fourth order PDEs on arbitrary smooth surfaces remains untouched due to the very complicated nature of these high order equations. We present a significant initial step in this direction.

Many difficulties arise when computing any fourth order diffusions. The dynamics depend highly on the smoothness of initial data. Boundary conditions are often difficult to implement; fourth order equations require prescribing two boundary conditions in contrast to the one needed for second order diffusions. Time stepping is perhaps the crucial matter for fourth order diffusions. Stability requirements restrict explicit time steps for fourth order diffusions to be on the order of h^4, where h is the grid size. Compare that to second order diffusions, where time steps are only restricted to be $O(h^2)$. The time step restriction prohibits explicit schemes in any meaningful simulation of fourth order equations. On the other hand, implicit schemes require the inversion of a linear system of equations that is typically very large for fourth order equations. Furthermore, the literature on solving fourth order diffusions in arbitrary smooth domains is virtually nonexistent. To date, a number of tools have proved successful for fourth order diffusions occurring in areas such as thin film fluid flow and materials science, including ADI, spectral, and finite element methods, as well as convexity splitting [5, 11, 19, 20, 51, 56, 60]. We address each of these methods, and explain how they come into play with our own problem when addressing general geometries.

Although our work shares many features with level set methods used for solving surface diffusion [8, 51] and Willmore flow [15], there are significant differences. Both dynamics are driven by fourth order equations, but unlike the problems we consider here, in surface diffusion and Willmore flow the surface is given by the PDE dynamics. Our equations are fundamentally different as the PDE is in some sense given by the surface. In the case of level set methods for surface diffusion, the dynamics quickly smooth the surface while damping regions of high curvature. In our case, those regions of high curvature remain unchanged and stay an integral part of the PDEs that we solve. Our computational methods also differ from [8], [15], and [51]. Unlike the work in [8], we use implicit times stepping schemes. Also, since we solve only in a band around the surface to speed-up the computation, we can not directly use the FFT method discussed in [51]. However we use operator splitting schemes much like those in [51]. The authors of [15] use finite element methods, whereas we use finite difference schemes.

In the next section we first provide the basic structure of the problem, explain its challenges, and describe the particular equations we will address. The general organization of the paper is detailed after this.

2 Basics, Challenges, and Goals

In order to adequately describe the challenge of computing higher order equations on surfaces, we first summarize the ideas developed in [2, 6, 57], and then describe where those methods must be modified. Let S be a given smooth closed curve or surface in $\mathbb{R}^N$, where N is assumed to be two when S is a curve, or three, when S is a surface. Let $\phi : \mathbb{R}^N \to \mathbb{R}$ be a smooth function whose zero level set is given by S with $\phi < 0$ inside S and $\phi > 0$ outside S. Thus $\frac{\nabla\phi}{|\nabla\phi|}$ gives the outer pointing normal to S. Letting I denote the identity matrix, define

$$\mathrm{P} := \mathrm{I} - \frac{\nabla\phi \otimes \nabla\phi}{|\nabla\phi|^2} \tag{1}$$

so that $\mathrm{P}(x_0)v(x_0)$ gives the projection of a vector v at the point $x_0 \in \mathbb{R}^N$ onto the tangent plane to the surface

$$S = \{x \in \mathbb{R}^3 : \phi(x) = \phi(x_0)\}.$$

Suppose u is a smooth function defined on S. We use $\nabla_S u$ to denote the gradient of u intrinsic to the surface S. This is simply the projection onto S of the gradient of (the extended) u in the embedded space [50]. We similarly use $\Delta_S f$ to denote the Laplacian of u intrinsic to S; in other words, Δ_S is the Laplace-Beltrami operator on S. Now suppose u is a function defined on all of $\mathbb{R}^N$. We use ∇u and Δu to denote the the standard Euclidean gradient and Laplacian of u. We calculate ∇_S and Δ_S using P and extrinsic derivatives (i.e., derivatives in the Euclidean space containing S): for all points x on S,

$$\nabla_S u(x) = \mathrm{P}(x)\nabla u(x)\ |\nabla\phi(x)|. \tag{2}$$

Since ϕ and P are defined in some domain $\Omega \subset \mathbb{R}^3$ containing S, if u is defined in all of Ω, then equation (2) makes sense in all of Ω. At each $x \in \Omega$, $\nabla_S u(x)$ defined as above corresponds to the gradient of u intrinsic to the level set of ϕ through x. We compute the Laplace-Beltrami operator in the same manner:

$$\Delta_S u(x) = \frac{1}{|\nabla\phi(x)|}\nabla \cdot \left(\mathrm{P}(x)\nabla u(x)\ |\nabla\phi(x)|\right). \tag{3}$$

If ϕ is a signed distance function,

$$|\nabla\phi| = 1,$$

then (2) and (3) simplify to

$$\nabla_S u(x) = \mathrm{P}(x)\nabla u(x)$$

and

$$\Delta_S u(x) = \nabla \cdot \left(\mathrm{P}(x)\nabla u(x)\ \right).$$

It is thus desirable, though unnecessary, to define ϕ as a signed distance function.

2.1 Example: The Heat Equation

Consider the heat equation on a given surface S. This equation arises from the gradient descent [6] of the energy

$$E(u) = \int_S \frac{1}{2}|\nabla_S u|^2 ds, \tag{4}$$

which is given by

$$u_t = \nabla_S \cdot \nabla_S u = \Delta_S u. \tag{5}$$

We combine (5) with an initial condition,

$$u(y,0) = f(y) \text{ for } y \text{ on } S, \tag{6}$$

to derive the heat equation on the surface S. We assume ϕ is a distance function as discussed above and use (3) to write (5) in terms of Euclidean derivatives:

$$u_t = \nabla \cdot (\mathrm{P}\ \nabla u). \tag{7}$$

Equation (7) is defined in all of $\mathbb{R}^N$ and can therefore be computed on a Cartesian grid. Solving (7) in all of $\mathbb{R}^N$ is equivalent to solving (5) on each level set of ϕ. In particular, it will be necessary to define an initial condition u_0 on the entire computational domain (a band around S) that is equal to f on S. Equation (7) is discussed and computed in [2, 6, 57].

2.2 Example: Linear Fourth Order Diffusion

This paper centers on fourth order equations, the simplest example being linear fourth order diffusion,

$$\begin{aligned} u_t &= -\Delta_S \Delta_S u \\ u(x,0) &= u_0. \end{aligned} \tag{8}$$

Equation (8) is the gradient descent of the energy

$$E(u) = \frac{1}{2}\int_S |\Delta_S u|^2 .$$

If S is given by the level set of a distance function ϕ, then we can compute (8) in the Euclidean space containing S (and then use Cartesian numerics) by the equation

$$\begin{aligned} u_t &= -\nabla \cdot (\mathrm{P}\,\nabla(\nabla \cdot (\mathrm{P}\,\nabla u))) \\ u(x,0) &= u_0, \end{aligned} \tag{9}$$

where P is the projection operator given by (1).

2.3 Basic Observations and Challenges

Equation (9) shares a number of challenges with (7). These are problems that have been addressed for the second order problem, but take on additional features and challenges in the high order equation:

1. **Domain.** We first note a key element of equations (7) and (9) as compared with equations (5) and (8). The new PDEs are defined in all of $\mathbb{R}^N$. This is both the advantage of the method and one of its main challenges. Since the new PDEs are defined in Eulerian coordinates, we can use a Cartesian grid and apply the usual finite difference schemes for solving these surface equations. However since the PDEs are defined in all of $\mathbb{R}^N$, we have increased the problem by a dimension. We minimize the additional work by computing only in a band around the surface. Fixing $c > 0$ and considering a signed distance function $\phi(x)$, we compute only in the band $|\phi(\mathbf{x})| \le c$. Unfortunately, this requires computing on unusual domains with curved boundaries, while to date, most simulations of fourth order equations have been done on rectangular domains. Thus we must simultaneously develop methods for solving fourth order equations on Cartesian grids with complicated boundaries.

   We also need to choose appropriate initial data for (7) and (9). For the underlying problem, our initial data is only defined on the surface. We must extend these values to a function defined in the entire band.

2. **Degeneracy.** Consider equation (7). At each point $\mathbf{x} \in \mathbb{R}^N$, (7) defines a diffusion that is degenerate in one direction. There is no diffusion in the direction normal to S, so the equation is extremely anisotropic. Similar anisotropic second order diffusions have been thoroughly studied; consider for example anisotropic diffusions used for image processing. One might argue that the fourth order equation (9) is even more degenerate, since it is a higher order diffusion with absolutely no diffusion in one direction. To our knowledge, no such equation has been studied. The closest example we know of is surface diffusion.

   This degeneracy plays a central role in our choice of computational methods. It leads us to consider convexity splitting methods similar to those used in [57] to deal with the degeneracy of the second order problem. In Appendix B we look at an ADI method that has been previously suggested for fourth order diffusions [56]. We examine this scheme and show why the degeneracy of the fourth order problem prohibits a direct application of this method. We also show how the second order degeneracy is better behaved, as it is amenable to the same approach. Although ADI can not be applied directly to equation (9), we do use it to invert the linear biharmonic operator in convexity splitting schemes that we discuss in Section 4.

3. **Boundary Conditions.** Suppose we are solving on the domain $S_c = \{x|\phi(x) \leq c\}$. No boundary conditions are required to solve the above PDEs in S_c, since no information is shared between level sets. The values of u on the boundaries $\phi = \pm c$ thus have no effect on u in the interior of the band. However, since we discretize on a Cartesian grid, any numerical solution will depend on the boundary, though this dependence should decrease as $h, \Delta t \to 0$. We also note that solving fourth order PDEs on bounded domains requires prescribing two boundary conditions; for example one might fix u and Δu, or $\nabla u \cdot \nu$ and $\nabla \Delta u \cdot \nu$ on the boundary (with normal ν).

Although we build upon previous work on equations like (7), we do not simply re-apply those methods. Equation (9) inherits all of the complications of other fourth order equations and has a few that are particular to its nature as a higher order surface PDE. These complications have not been addressed in the work on second order surface PDEs, and in fact many of them are central to current research on the computation of general higher order equations:

1. **Dependence on high derivatives.** This obvious difference between equations (7) and (9) presents many difficulties that were not apparent in the works discussed above. Equation (9) depends on fourth derivatives of not only u, but of ϕ as well. This high order dependence places severe restrictions on u and ϕ. Both must be smooth and accurate to a high enough degree that they possess smooth numerical derivatives. As we shall see, level set functions ϕ that are smooth enough to use in computations of second order PDEs might not be smooth enough for their fourth order counterparts. The initial condition u_0 must also be sufficiently smooth. This plays a particularly important role when extending the initial condition on S to a band containing S. See section 3.4 for more details.

2. **Stencil Size.** In one dimension, the standard centered difference discretization of the biharmonic operator has a stencil width of 5 grid points. In three dimensions, the stencil size is 33 grid points. Compare that to the 7 grid points in the standard discretization of the three dimensional Laplacian. The stencil size becomes especially problematic when solving PDEs on a surface. In all likelihood, all but one or two of these grid points will lie on neighboring contours of ϕ ($\phi \neq 0$), and not on the surface of interest ($\phi = 0$). Thus we must extend the initial data off of S carefully. We pick u_0 to be constant in directions perpendicular to S for this reason. Choosing u to be non-constant in this direction could significantly increase the discretization error.

   Stencil size also plays a central role in computation speed. Discretizing the Laplacian requires far fewer operations than discretizing the biharmonic. There is an even greater difference in the number of operations required between (7) and (9). Computing (7) requires at least one matrix multiplication by P at each time step, whereas equation (9) requires at least two such multiplications. We must take this need for heavy computation into consideration when considering the domain of computation, much more so than in the case of (7).

3. **Time stepping.** Explicit methods for fourth order diffusions have a time step restriction requiring the time step k to be on the order of h^4, where h denotes the grid cell width. This restriction is far worse than the stability requirement for second order diffusions. Due to this drawback of explicit schemes, we turn to implicit schemes that require the inversion of a linear system at each time step. Since fourth order equations depend on high derivatives, we desire a highly resolved grid for accuracy, however refined grids are detrimental to time stepping. Explicit schemes would be too

slow, while the necessary inversion for implicit schemes might be too computationally expensive. Time stepping remains a major challenge for solving fourth order equations.

2.4 Overview of computational examples

We consider three fundamental fourth order problems in this paper: linear fourth order diffusion,

$$u_t = -\Delta_S \Delta_S u,$$

the Cahn-Hilliard equation for phase transitions in a binary alloy,

$$u_t = \Delta_S \left(-\varepsilon^2 \Delta_S\, u + u^3 - u\right),$$

and a PDE derived in [47] for the motion of thin films driven on a surface by gravity and surface tension,

$$\begin{aligned} \frac{\partial \zeta}{\partial t} &= -\frac{1}{3}\nabla_S \cdot \left[u^2 \zeta \nabla_S \tilde{\kappa} - \frac{1}{2} u^4 \left(\kappa \mathbf{I} - \mathrm{K}\right) \cdot \nabla \kappa\right] \\ &\quad -\frac{1}{3} Bo \nabla_S \cdot \left[u^3 \hat{\mathbf{g}}_{\mathbf{s}} - u^4 \left(\kappa \mathrm{I} + \frac{1}{2}\mathrm{K}\right) \cdot \hat{\mathbf{g}}_{\mathbf{s}} + \hat{g}_n u^3 \nabla_S u\right]. \end{aligned} \tag{10}$$

In equation (10), k_1 and k_2 are the principle curvatures of the substrate, $\kappa = k_1 + k_2$, $\tilde{\kappa}$ is curvature of the free surface, $\zeta = u - \frac{1}{2}\kappa u^2 + \frac{1}{3} k_1 k_2 u^3$, and K is the curvature tensor for the curved substrate. The Bond number Bo quantifies the relative strength of gravity to surface tension. The vector $\hat{\mathbf{g}}_{\mathbf{s}}$ is the component of gravity tangent to the surface S, and $\hat{g}_n$ is the magnitude of the component of gravity normal to S.

For these problems, we address each of the issues discussed in Section 2.3, describe the methods we used to deal with those issues, and mention some areas of possible future research. We compare with second order problems throughout the discussion. In Section 3 we discuss the groundwork for implementation, including data structures, boundary conditions, extension of the initial data off of S, and intermittent re-extension of the surface data. Although these elements have been discussed previously in [2, 6, 57], they take on new features and challenges with the fourth order problem. Sections 4 and 6 concern the main challenge: time stepping of the fourth order highly degenerate diffusions. We discuss both ADI and convexity splitting methods that have been successful for other fourth order equations and consider their relation to our problem. The new challenges here include the extreme degeneracy of these surface equations and the unusual computational domains.

The remaining sections demonstrate the generality of our methods by applying them to each of the above equations. In Section 5 we consider linear fourth order diffusion (9) on the unit circle in $\mathbb{R}^2$ and the unit sphere in $\mathbb{R}^2$. We give pseudocode describing the basic steps of the method. We also check convergence in the case of fourth order diffusion on a circle, for which we know exact solutions. Section 7 concerns the Cahn-Hilliard equation. We give a numerical scheme and discuss changes to the algorithm given for the linear problem. These changes are minor, since the highest order term is still linear. Finally, in Section 8 we apply these methods to a fully nonlinear model for thin film fluid flow on surfaces. We consider flows driven by gravity and flows driven by curvature alone. To demonstrate the effects caused by the curvature terms, we compare the model that takes curvature into consideration with a simpler model that ignores curvature effects.

3 General Implementation Issues

We first describe methods for the general implementation of PDEs on implicit surfaces, including data structures, extension of initial data off of the surface, boundary conditions, and modifications of the methods in [2, 6, 57] that are required to compute higher order PDEs. These are the essential building blocks for the numerical computations in sections 5-8. A number of these tools can be used to solve many different PDEs on a variety of geometries.

We assume that we are given a level set function ϕ that determines S by $S = \{x|\phi(x) = 0\}$ with $\phi(x) > 0$ for points x outside of S and $\phi(x) < 0$ for x inside S. Works such as [43, 45, 52] and the papers mentioned in the introduction discuss means of producing such an implicit representation (e.g. via highly accurate WENO schemes). For fourth order equations, very high accuracy is required for the level set function since we must take fourth order derivatives of ϕ as well as of u. As noted in Section 2, choosing ϕ to be a signed distance function simplifi es the PDEs.

3.1 Surface Complexity

The examples in this paper fall into two categories of implicit surfaces. The fi rst are “simple” surfaces, with small curvature, in which we have an analytical expression for the level set function. Such examples include ellipses, the unit circle in $\mathbb{R}^2$, and the torus and sphere in $\mathbb{R}^3$. The second category are complex surfaces defi ned computationally by a dataset, with regions of high curvature. Our example of such a surface is the Stanford Bunny [30]. In this second case, we found that the level set function from the original Stanford Bunny data is not smooth enough for fourth order differences. To address that problem, we slightly smoothed the bunny by applying a few steps of the heat equation to the level set function. This results in smoothing high derivatives of ϕ while retaining the essential bunny shape. Such smoothing is not required for solving second order equations, and might be avoided for high order fbws by using local mesh refi nement or adaptive grids.

Surface complexity also affects the choice of time stepping method. The geometry of the bunny has regions of very high curvature that are not completely resolved by our 3D grid with $159 \times 161 \times 129$ points. The fi nite difference discretization of the projected PDE introduces terms with third and fourth derivatives of ϕ that produce a time scale in the calculation on the order of $\frac{1}{h^3}$. A stable method requires resolving this time scale, thus introducing a time step limitation due to the geometry of the surface, as opposed to the underlying scheme. When smaller time-steps are required by the surface geometry, we found ADI schemes to be more effi cient than iterative solvers. We discuss this further in sections 4 and 6.

3.2 Computational Domain: A Band Containing S

To reduce computational costs, we restrict computation to a band around the surface,

$$S_c = \{|\phi(\mathbf{x})| \leq c\}, \quad c > 0.$$

The value of the parameter c selected generally depends on the size of the numerical stencil and the complexity of the working surface. This restriction to a band is similar in spirit to the local level set method [1, 45]. For example, when computing linear fourth order diffusion on the unit circle, we could compute (9) in a square containing the circle ($[-2,2] \times [-2,2]$ for example), but instead it is more effi cient to choose the annulus

$$1 - c \leq \sqrt{x^2 + y^2} \leq c + 1$$

as the computational domain (see Fig. 1).

3.3 Data Structures

Throughout we rely only on the implicit structure of arrays. We follow the example of the local level set algorithm described in [45]. We fi rst defi ne an array that stores the values of ϕ within some box containing S. For our discussion we assume an N-dimensional box with M grid-points on each side, so ϕ is stored in

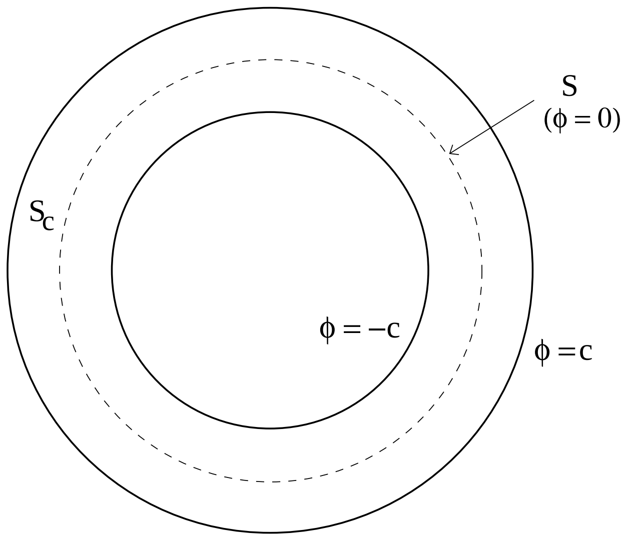


Figure 1: S denotes the surface corresponding to the zero level set of ϕ, and S_c is a band around that surface used for the computations.

an array with M^N elements. We then make a list of all points that are within the band S_c. For simplicity, we also store the values of u and any other functions necessary for solving the PDE in arrays of length M^N. Particular entries of each array correspond to particular grid-points in the box containing S, however, to minimize memory use, one could use only one array of length M^N and at grid-points lying in the band, store pointers to the necessary function values. By using array storage we access each grid-point's nearest neighbors without any searching. This is useful for PDEs discretized by finite differences. To illustrate how this would work in practice, for our simple example of the annulus in R^2, we define ϕ in all of $\Omega = [-2,2] \times [-2,2]$ on a Cartesian grid with M^2 grid-points and cell width $h = 4/(M-1)$. We mark the grid-points $x_{i,j}$ satisfying $|\phi(x_{i,j}| \leq c$. For simplicity we also define u on the entire grid, but we compute using only the values of u within the band.

3.4 Extension of Initial Surface Data

Since u_0 is assumed to be defined only on the surface S, we must extend u_0 to the other level sets of ϕ within the band. We extend u_0 by requiring

$$\nabla u_0 \cdot \nabla \phi = 0. \tag{11}$$

In other words, we require u_0 to be constant perpendicular to the surface. In our simple example of the annulus, condition (11) states that in polar coordinates (r, θ), u is a function of θ alone.

One way of forcing (11) is to solve

$$u_t + H_h(\phi) \frac{\nabla \phi}{|\nabla \phi|} \cdot \nabla u = 0 \tag{12}$$

to steady state within the band, where $H_h(s)$ is a smoothed version of the Heaviside function depending on the gridsize h,

$$H_h(s) = \frac{s}{\sqrt{s^2 + h^2}}.$$

Since $H(0) = 0$, values on S remain unchanged while the information is propagated both inside (where $H(\phi) < 0$) and outside ($H(\phi) > 0$) the surface. Discussion of this method, including the selection of H_h, can be found in [43] and [45]. For second order equations on surfaces, equation (12) can be solved using simple first order up-winding. However this is inadequate for our purposes, as such simple integration results in an extended u that has very noisy high derivatives. We compute (12) using fifth-order WENO [25, 26] for spatial derivatives and integrate in time with second order Runge-Kutta.

3.5 Re-extension of Surface Data

We also note that u must be intermittently re-extended off of the surface, so the method used to compute (12) must be both accurate and efficient, further suggesting the use of a high order WENO scheme. The frequency of re-extension depends on time steps used in the integration of the surface PDE. If large time steps are taken, we re-extend each time step by solving (12) to steady state. This can often be done in one or two iterations of the numerical scheme for (12). When taking small time steps, we only re-extend every few steps as needed. This need to re-extend surface data is also discussed in [57]. In the simple example of an annular domain, the re-extension is a means of adjusting the values of u off of S so that it depends only on θ. In general, doing a re-extension enables us to minimize effects on surface values of u by values of u off of the surface.

3.6 Boundary Conditions

Consider, for example, equation (9) defined only in the band $S_c = \{x \,|\, |\phi(x)| \leq c\}$. The boundary of S_c consists of the disjoint curves $\phi = \pm c$ (see Figure 1). Since the boundary is given by level sets of ϕ, information is not transferred from the boundary, and thus boundary conditions are not needed to solve the PDE. However, when the PDE (9) is discretized using finite differences on a Cartesian grid in S_c, because the grid directions do not run parallel to the boundary of the annulus, we must impose numerical boundary conditions for computation which then affect the solution. However for short times and a large enough band size, these boundary conditions should have minimal effect on the value of the solution at the surface itself. We have some choice in the selection of the numerical boundary conditions on the edge of the band, and in this paper we choose the boundary conditions that are easiest to calculate.

Consider solving the second order problem, equation (7), in S_c. Since we extend u off of S in such a way that $\nabla u \cdot \nabla\phi = 0$, Neumann conditions would be the natural boundary conditions to apply. However for the analogous fourth order problem, Neumann conditions are trickier to apply, especially on curved domains. Instead, for that problem, we pick Dirichlet boundary conditions, which are easier to implement for the fourth order problem. Moreover, Dirichlet boundary conditions assure us of a symmetric matrix when we invert the linear biharmonic operator, thus allowing us to use the Jacobian Conjugate Gradient method. We prescribe u and $\Delta_S u$ on the boundary, which is more natural in computations than prescribing u and Δu on the boundary – otherwise we will be required to compute $\Delta_S u$ from Δu. Our choice for $\Delta_S u$ on the boundary actually results from extending the initial condition far enough past the boundary to compute a finite difference approximation of $\Delta_S u$.

The numerical values of u near the band boundary are often strongly affected by the boundary conditions. We significantly reduce this effect by regularly re-extensions of u off of S and re-initializing the Dirichlet conditions based on this re-extension. By re-extending u up to the boundary, we will have a more appropriate boundary condition of u for the next time-step. By re-extending u at least one grid cell beyond the boundary, we can compute more appropriate values of $\Delta_S u$ for the next time step. In the special case of the annular geometry of in Fig. 1, fixing u and $\Delta_S u$ at the boundary is equivalent to fixing u and $u_{\theta\theta}$ in polar coordinates.

3.7 Visualization

We use MATLAB to visualize the PDE dynamics in all of our examples. The MATLAB routine *isosurface* provides an easy means of interpolating values of ϕ to produce the surface S and rendering a color map of u on S.

4 Time Stepping

A number of choices are available for time integration of the finite difference schemes, each with its own advantages and disadvantages. Explicit schemes are often used for second order diffusion equations since they are easy to implement; to date they are the most commonly used schemes for these surface problems [2, 6]. For fourth order diffusions, however, stability considerations restrict explicit time steps to be on the order of h^4. This severe restriction prohibits calculating on fine meshes by explicit schemes. We thus turn to implicit schemes which have no time step restriction for linear fourth order diffusion, though solving the linear system may require extensive computation. Inverting the matrices for solving fourth order diffusions in three dimensions is computationally expensive due to the 33 point stencil of the biharmonic operator. Nonlinear systems are even more difficult to solve, as they require repeated Newton iterations, and time steps may be restricted by convergence of the Newton method.

All the above issues come into play when picking a method for time stepping. We seek a method that is effective for strongly degenerate PDEs while being easy to implement and robust to changes in both the PDE and computational domain. These goals lead us to convexity splitting methods, which have proved successful for the Cahn-Hilliard equation [16], degenerate fourth order diffusions including models for thin films [18], and for surface diffusion [51], and for second order diffusions on surfaces [57]. Our convexity splitting schemes require the inversion of a linear system at each time step. We consider two methods for inverting these systems: the conjugate gradient method and an Alternating Direction Implicit (ADI) method. Each method is advantageous for particular surfaces, as will be discussed below.

In this section, we first describe convexity splitting methods, then demonstrate their application to linear diffusion with Jacobian Conjugate Gradient for solving the linear systems. For now we restrict our discussion to "simple" surfaces with relatively small curvature. In section 6 we turn to more complicated surfaces, and discuss how we use ADI methods to invert the linear systems arising from solving PDEs on these surfaces.

4.1 Convexity Splitting

Given a gradient flow PDE of the form

$$u_t = F(u),$$

we split it into two parts

$$u_t = F_1(u) + F_2(u), \tag{13}$$

where F_1 has a strictly convex energy and F_2 has a strictly concave energy. Note this decomposition is non-unique. We update each time step by integrating $F_1(u)$ implicitly and $F_2(u)$ explicitly; for example, we might use the scheme

$$u^{n+1} = u^n + k(F_1(u^{n+1}) + F_2(u^n)), \tag{14}$$

where u^n denotes the value of u after n time steps of size k. Scheme (14) is unconditionally stable in most cases, and it has an $O(k)$ error [16]. For gradient flows, this low time stepping accuracy is not an issue, since the objective is to reach a local minimum of the energy. Although (14) may be unconditionally stable, larger time steps do not necessarily bring the solution to this minimum any faster, rather there is typically an optimal finite k for obtaining a local minimum of the gradient flow's energy.

This same method may be applied to problems that are not a gradient descent; for example, consider the equation

$$u_t = -\nabla \cdot (a(\mathbf{x}) \nabla \Delta u).$$

Instead of computing this equation directly, we rewrite it as

$$u_t = -C\Delta\Delta u + \nabla \cdot ((C - a(\mathbf{x})) \nabla \Delta u). \tag{15}$$

If C is chosen large enough, then we can set

$$F_1(u) = -C\Delta\Delta u$$

and

$$F_2(u) = \nabla \cdot ((C - a(\mathbf{x})) \nabla \Delta u),$$

to derive the unconditionally stable scheme

$$\frac{u^{n+1} - u^n}{k} = -C\Delta\Delta u^{n+1} + \nabla \cdot ((C - a(\mathbf{x})) \nabla \Delta u^n). \tag{16}$$

The same method has been used for other fourth order PDEs, including surface diffusion [51], the Cahn-Hilliard equation [16, 54], and Hele-Shaw fbw [18]. Xu and Zhao used the same idea to compute second order diffusion on surfaces in [57]. We describe exactly how C is chosen for each of our numerical examples in later sections. The different equations we consider each require different choices.

Our main challenge is to solve (16) quickly with a method applicable to non-rectangular domains. Since they computed in rectangular domains, the authors of [18] and [51] were able to use Fast Fourier Transform (FFT) methods effectively. We experimented with similar methods for our particular problem, and found computing in a three dimensional box with FFT to be slower than using the methods we present here for computing in a narrow band around the surface. Depending on the complexity of the surface under consideration, we use one of two different methods for the linear inversion – Conjugate Gradient or Aternate Direction Implicit.

4.2 Example: Linear Fourth Order Diffusion

We fi rst demonstrate convexity splitting methods applied to linear fourth order diffusion on surfaces,

$$u_t = -\Delta_S^2 u. \tag{17}$$

Assume $|\nabla\phi| = 1$ and let P denote the projection matrix of Section 2, so that (17) can be rewritten as

$$u_t = -\nabla \cdot (\mathrm{P}\nabla\Delta_S u).$$

Noting that $\mathrm{P} = \mathrm{I} - \mathrm{N}$ where N is a matrix projecting onto the surface normal, $\nabla\phi$, we see

$$\Delta_S^2 u = \Delta^2 u - \nabla \cdot (\mathrm{N}\nabla\Delta_S u). \tag{18}$$

Equation (18) also results from setting $C = 1$ in (15). At each time step, we integrate the fi rst term on the right of (18) implicitly and the remaining term explicitly:

$$\frac{u^{n+1} - u^n}{k} = -\Delta^2 u^{n+1} + \nabla \cdot (\mathrm{N}\nabla\Delta_S u^n). \tag{19}$$

To simplify (19), we use an approach suggested in [56]. Let $L = (I + k\Delta^2)$, subtract Lu^n from both sides of (19), and group terms to get

$$L(u^{n+1} - u^n) = -k\Delta^2 u^n + k\nabla \cdot (\mathrm{N}\nabla \Delta_S u^n)\,. \tag{20}$$

We define $v^{n+1} = u^{n+1} - u^n$ and notice the right side of (20) is $-k\Delta_S^2 u^n$ to derive the $O(k)$ scheme,

$$Lv^{n+1} = -k\Delta_S^2 u^n. \tag{21}$$

When solving with Dirichlet boundary conditions, $v \equiv 0$ on the boundary, thus simplifying L. Boundary conditions only affect the explicit term, $-\Delta_S^2 u^n$, which suggests fixing $\Delta_S u$ on the boundary, instead of Δu. Note that solving the original form (19) requires using the boundary values of both Δu and $\Delta_S u$: the boundary values of Δu are required to define L near the boundary and $\Delta_S u$ is needed to compute the explicit term near the boundary. We discuss the discretizations of all the spatial operators in Appendix A. As the reader will notice, we use standard finite difference stencils.

4.3 Iterative Solver: Conjugate Gradient Method

On domains with Dirichlet boundary conditions, the standard stencil for $\Delta^2 u$ is symmetric, so we can solve (21) using the Conjugate Gradient Method. We use the Conjugate Gradient solver provided by ITPACK [28]. Unfortunately a large number of iterations may be required at each time step, especially for the typically large computations in three space dimensions. In cases where time steps are not limited by stability, the large number of iterations is not prohibitive, and this method is a tremendous improvement over explicit schemes. However, if the time steps taken must be very small (as we found to be the case for surfaces with regions of high curvature), this method provides little, if any, improvement over explicit methods, which require no iteration at each time step.

5 Numerical Examples: Linear Diffusion on Simple Surfaces

In this section we present the first example, and the simplest one, linear fourth order diffusion on simple surfaces. Our basic computational framework follows the following algorithm:

1. Store values of the signed distance function ϕ on a Cartesian grid in a rectangular domain containing S.
2. Mark grid points x_{ij} (array entries) satisfying $|\phi(x_{ij})| \leq c$ for the user-defined bandwidth c.
3. Store an initial value u_0 that satisfies $u_0 = f$ on S.
4. Evolve u_0 by solving (12) until (11) is satisfied in some domain containing the band (far enough to compute $\Delta_S u_0$ on the band's boundary).
5. Compute values of $\Delta_S u_0$ on the band's boundary.
6. Do while $t^n < t_{Max}$
   (a) Solve for u^{n+1} in (21) using Jacobi Conjugate Gradient method.
   (b) Solve (12) to re-extend values of u off of surface past band boundary.
   (c) Recompute $\Delta_S u$ on boundary.

We now present computational results for (19) on both the unit circle in $\mathbb{R}^2$ and the unit sphere in $\mathbb{R}^3$. We remind the reader that discretizations of all spatial derivatives are described in Appendix A.

5.1 Diffusion on the Unit Circle

We compare our computations with an exact solution of (17) on the unit circle. In this case, solving (17) is exactly the same as solving

$$u_t = -u_{ssss} \tag{22}$$
$$u(s,0) = f(s) \tag{23}$$

on $[0,2\pi]$ with periodic boundary conditions. Equation (22) can be solved exactly using separation of variables. We choose $f(s) = \cos(4s)$, giving the exact solution

$$u(x,t) = \cos(4s)e^{-64t}.$$

We use conjugate gradient method to invert the implicit term. We compare our results with the exact value of u at times $t = 0.001$ and $t = 0.01$. Although these times are small, the half life of (17) for this initial data is approximately $t = 0.01$. We define the level set function on the box $[-2,2]\times[-2,2]$ using grid sizes of $h = \frac{4}{M-1}$ where $M = 40, 80, 160,$ and 320. We compute in a band of width 3/8 in these calculations. Since our scheme is $O(k)$, we use a time-step of $k = \frac{1}{100}h^2$. We must choose a small constant of proportionality to get adequate results. We expect this is due to the high degeneracy of the equation. We must choose a small constant of proportionality to get adequate results. We expect this is due to the high degeneracy of the equation. The error in the long-time calculation is of course improved by careful re-extension, although we report here the results without it in order to better identify the contributions of the different components of our scheme. Note that for $M = 320$, over 300 time steps are necessary to reach $t = 0.01$. Without re-extension of data off of the surface, the Dirichlet boundary conditions increase the solution's error. Figure 2 demonstrates an error of $O(h^2)$ in the short-time calculation and $O(h)$ in the long-time calculation. This decrease in accuracy is caused by the choice of Dirichlet Boundary conditions. We found that we maintain $O(h^2)$ accuracy by re-extending the initial data after every five time steps. To calculate the error, we interpolated to find the values of u on the circle and then used the l^∞ norm of the difference of these values from the exact solution.

5.2 Diffusion on the Unit Sphere

Figure 3 shows linear fourth order diffusion on the unit sphere. Implementation requires only slight changes from the example of a circle. We only need to change the level set function and use finite differences in three dimensions. This easy adaptability to higher dimensions is typical of level set methods. In addition, we may easily compute on surfaces such as ellipsoids or tori by changing only the level set function.

In our example, the level set function ϕ isdefined on $[-2,2]\times[-2,2]\times[-2,2]$ with $100\times100\times100$ grid points. The initial condition is defined using polar coordinates by $u_0(\rho,\theta,\beta) = \sin(3\theta)\sin(7\beta)$. We take time steps $k = 5h^2$ where h is the grid cell width. Figure 3 displays the solution at $t = 0$, $t = 0.2$, and $t = 0.9$. We use a bandwidth of ten grid cells (five cells off the surface in each direction), and re-extend the initial data every four time steps by solving (12) to steady state. See Appendix A for the spatial discretizations of the operators in (21).

6 ADI and Complex Geometry Surfaces

In Section 5, we solved linear diffusion on two simple geometries: the unit circle, and the unit sphere. For these examples, Conjugate Gradient Method provided an efficient means of inverting the linear systems in our implicit schemes. Unfortunately, as we discovered experimentally, the Conjugate Gradient Method

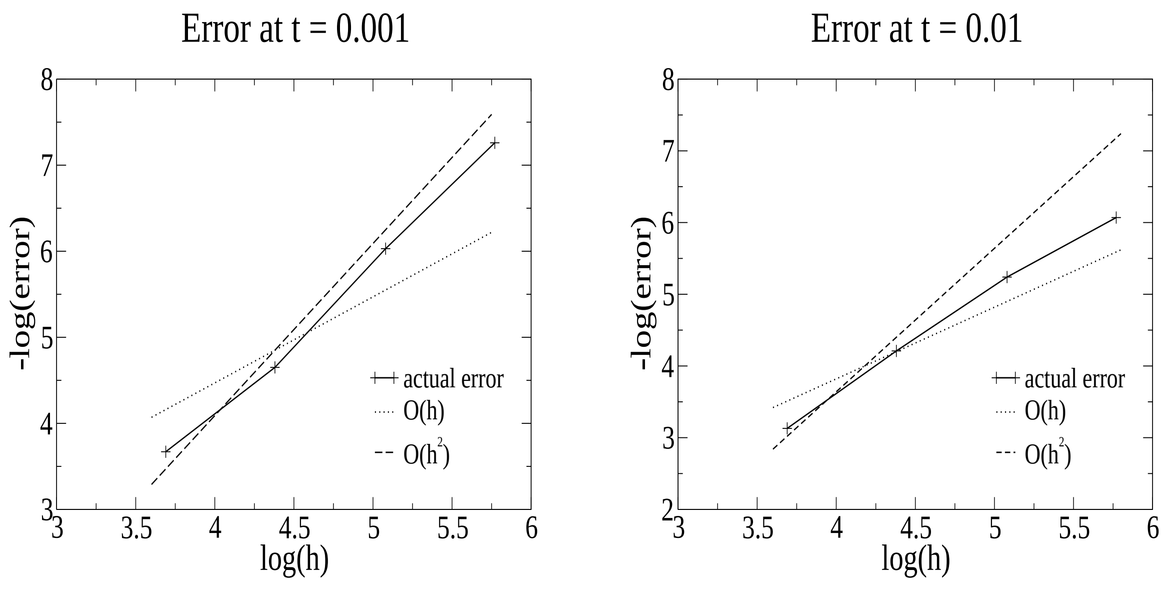


Figure 2: Error plot for linear fourth order diffusion on the unit circle, without re-extension.

often requires a large number of iterations. This becomes prohibitive when forced to take small time steps, as we found to be the case for solving PDEs on surfaces with regions of high curvature (see Section 3.1). For these surfaces, we turn to Alternate Direction Implicit (ADI) schemes, which do not require repeated iterations. ADI schemes involve the inversion of only banded matrices, which require only $O(M)$ operations for an M-by-M matrix. ADI schemes have been used for both second and fourth order PDEs [12, 13, 14, 56, 58]. For example, assume we are solving the heat equation on a rectangle with an implicit scheme:

$$\frac{u^{n+1}-u^n}{k} = u_{xx}^{n+1} + u_{yy}^{n+1}. \tag{24}$$

The following is an $O(k)$ approximation of (24):

$$(\mathrm{I}-k\partial_x^2)(\mathrm{I}-k\partial_y^2)u^{n+1} = u^n. \tag{25}$$

Unlike (24), equation (25) only requires the inversion of tridiagonal matrices.

Linear fourth order diffusion,

$$u_t = -\Delta^2 u, \tag{26}$$

is more difficult to implement with ADI, as Δ^2 includes a cross-term, $2\partial_x^2\partial_y^2$. In [56], Witelski and Bowen suggest an ADI scheme in which the mixed derivative term is computed explicitly. They showed that

$$(\mathrm{I}+k\partial_x^4)(\mathrm{I}+k\partial_y^4)u^{n+1} = (\mathrm{I}-2\mathrm{k}\partial_\mathrm{x}^2\partial_\mathrm{y}^2)\mathrm{u}^\mathrm{n} \tag{27}$$

is an unconditionally stable $O(k)$ scheme for (26). Higher order accurate ADI schemes following from the same idea are discussed in [56].

Unfortunately, applying an ADI scheme directly to the degenerate diffusions we consider here yields a scheme with a stability restriction that is no better than for explicit schemes. The interested reader will find a discussion of this in Appendix B. In order to improve upon the stability restrictions of explicit schemes, we combine ADI with convexity splitting.

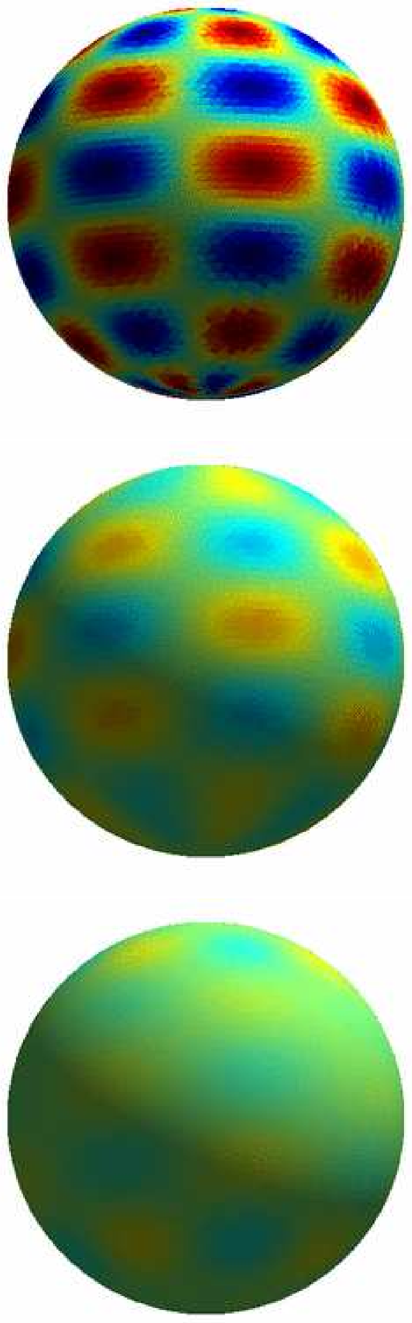


Figure 3: Linear fourth order diffusion on the sphere after 0, 4, and 18 time steps. Red denotes $u = 1$ and blue denotes $u = -1$.

6.1 Example: Linear Fourth Order Diffusion

As a first example, consider solving (19) with an ADI method. Rewrite (19) as

$$(I + k\Delta^2)u^{n+1} = u^n + k\nabla \cdot (\mathrm{N}\nabla\Delta_S u^n)\,. \tag{28}$$

Letting $L_x = (I + k\partial_x^4)$ and $L_y = (I + k\partial_y^4)$, we introduce $O(k)$ errors by modifying (28) to

$$L_x L_y u^{n+1} = (I - 2k\partial_x^2\partial_y^2)u^n + k\nabla \cdot (\mathrm{N}\nabla\Delta_S u^n)\,. \tag{29}$$

Now subtract $L_x L_y u^n$ from both sides and define $v^{n+1} - v^n$ to derive the compact scheme

$$L_x L_y v^{n+1} = -\Delta_S^2 u^n. \tag{30}$$

Unlike Locally One-Dimensional (LOD) schemes, this method easily extends to three dimensions, producing an unconditionally stable scheme,

$$L_x L_y L_z v^{n+1} = -\Delta_S^2 u^n, \tag{31}$$

with $L_z = (I + k\partial_z^4)$. This easy application to three dimensions is obviously important for the problems we consider. However ADI schemes have a disadvantage in non-rectangular domains such as the narrow band ($|\phi(x)| \leq c$) we consider. This is best seen by rewriting (27) as

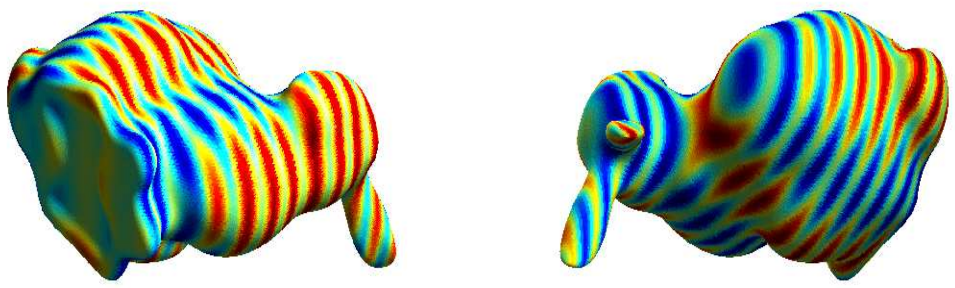
$$\begin{aligned} w &= L_y u^{n+1} \\ L_x w &= (I - 2k\partial_x^2\partial_y^2)u^n. \end{aligned} \tag{32}$$

At the half time step we solve for w, which does not have the same boundary conditions as u. Solving for these boundary conditions on non-rectangular domains is a tricky problem that becomes more difficult for complicated domains – domains like the bands we wish to compute in. Instead we approximate the boundary condition by fixing $w = u$ on the boundary, as suggested in [58]. Unfortunately this method is not unconditionally stable as we discovered in our numerical experiments. Even with this restriction, however, we found this ADI method to be more efficient than Conjugate Gradient Method for solving PDEs on surfaces with complicated geometries.

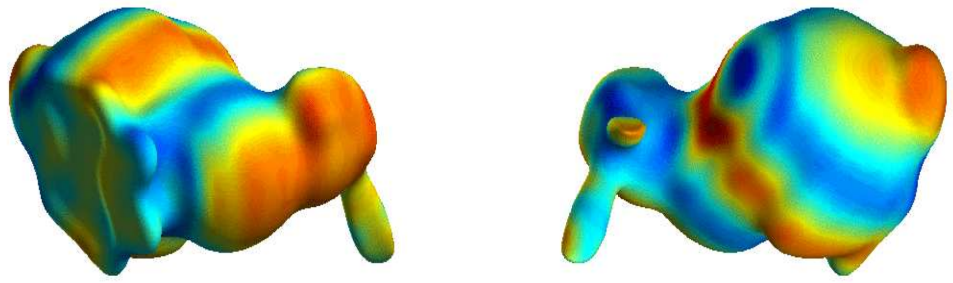
6.2 Numerical Example: Linear Fourth Order Diffusion on the Stanford Bunny

The bunny data for this example is taken from [30]. Due to the existence of regions of high curvature on the bunny, we first run the standard heat equation in $\mathbb{R}^3$ on the bunny's level set function for a very short time period. This smoothes any singularities in the level set function for the bunny while maintaining its essential structure, as Figure 4 shows. We found this surface smoothing to be unnecessary for solving second order equations on the bunny – it is required for higher order equations at the grid resolution with which we work. After smoothing ϕ, we can either use this new ϕ as is, or we may keep only its zero level set then reinitialize ϕ so that it is again a distance function.

We follow the algorithm in Section 5 with only two modifications. As just discussed, there is an added preprocessing step where the level set function for the bunny is smoothed, and instead of solving our numerical scheme with the Conjugate Gradient method at each time step, we use the ADI scheme (31). Figure 4 shows fourth order linear diffusion on the surface of the bunny. We use a cubic grid with cell width $h = \frac{1}{4}$. There are $159 \times 161 \times 129$ grid cells. We use a bandwidth of about 10 grid cells. The time step is $k = \frac{1}{250}h^3$. When computing with an explicit scheme, we needed $k = \frac{1}{500}h^4$, so there is still significant improvement over the explicit method. We define the initial condition in all of R^3 by $u_0(x,y,z) = \frac{1}{2}(cos(\frac{5}{2}x) + sin(\frac{2}{5}y))$. We then fix u on the bunny and use the extension procedure to change the values of u off of the surface so that the initial condition satisfies $\nabla u_0 \cdot \nabla\phi = 0$. We re-extend every four time steps.



$t = 0$



$t = 0.2$

Figure 4: Linear fourth order diffusion. Each row shows two views of the surface at the same point in time.

7 Cahn-Hilliard Equation

We now consider a classical phase-field model for spinodal decomposition of alloys, the Cahn-Hilliard equation [41]. Such models describe coarsening dynamics such as the phase separation following a quench from a disordered to an ordered phase. Computer simulations play an important role in the characterization of late-stage coarsening processes. Recent efforts have focused on developing numerical methods for the Cahn-Hilliard equation in Euclidean geometries using finite element methods [4] and psuedospectral methods [54]. Here we develop a numerical method for solving the Cahn-Hilliard equation on a general surface, using the implicit representation approach.

We solve

$$u_t = \Delta_S \left(-\varepsilon^2 \Delta_S\, u + u^3 - u\right), \tag{33}$$

for which $u = 1$ and $u = -1$ are both stable steady states. In our examples we use an initial condition of $u = 0$ plus a very small zero mean perturbation. The solution u quickly separates the surface S into two regions S_+ and S_- where u takes on values of 1 and -1 respectively. The remaining points of S lie on the interface of width $O(\varepsilon)$ between these two regions. In later stages, u undergoes spinodal decomposition; S_+ and S_- change shape so that the length of the interface between the two regions decreases while maintaining the area of each region (and the mean value of u). This coarsening of S_+ and S_- slows with time.

In free space, the Cahn-Hilliard equation is a diffuse interface model for Mullins-Sekerka dynamics [40, 44], and a version of the Cahn-Hilliard equation with degenerate mobility is a diffuse interface model for Hele-Shaw [18]. For flows on surfaces, we might expect to be able to use these models as diffuse interface models of curve evolution on complicated geometries. This idea is further suggested by the simulations shown in figures 5 and 6.

Equation (33) is slightly more complicated than (17) due to the nonlinear second order term. We pick $F_1(u) = -\varepsilon^2\Delta^2 u$ in (13) to derive the scheme

$$\frac{u^{n+1} - u^n}{k} = -\varepsilon^2\Delta^2 u^{n+1} + \varepsilon^2\nabla\cdot(N\nabla\Delta_S u^n) + \Delta_S\left((u^n)^3 - u^n\right), \tag{34}$$

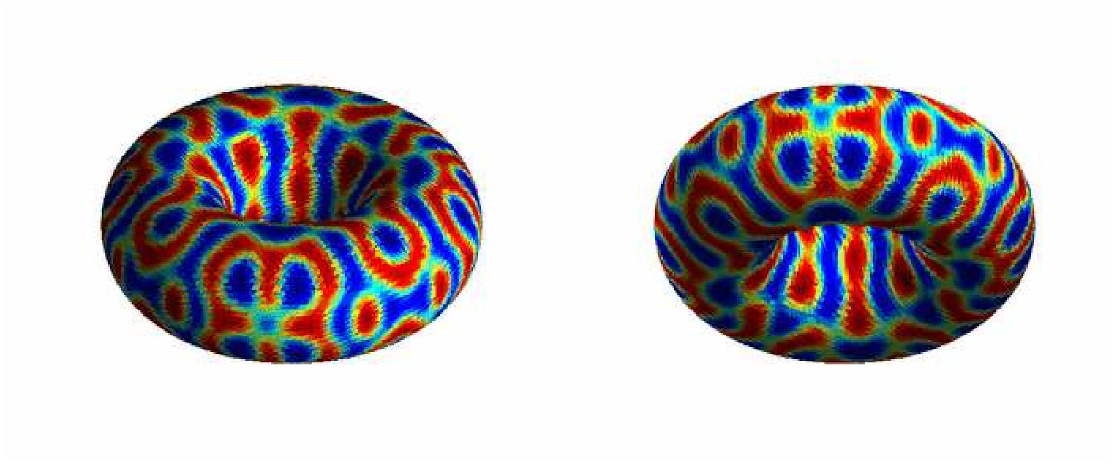
which has a stability requirement of $k = O(h^2)$. We define $v^{n+1} = u^{n+1} - u^n$ and $L = I + k\varepsilon^2\Delta^2$ to simplify (34) as was done in Section 4.2:

$$Lv^{n+1} = \Delta_S(-\varepsilon^2\Delta_S u^n + (u^n)^3 - u^n). \tag{35}$$

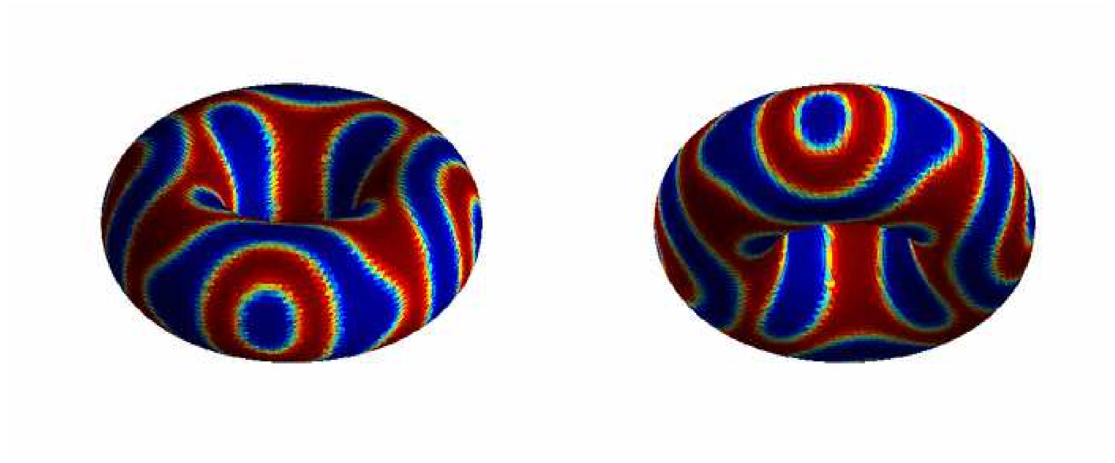
Note that our method does not split the energy for (33) into convex and concave parts. Doing so requires inverting an operator involving $\Delta_S u^{n+1}$, and we found this to be prohibitively slow in practice.

We first solve (33) on a torus. The surface does not seem to affect the stability of (35), so we take large time steps (relative to explicit schemes) and solve the linear systems with Jacobi Conjugate Gradient, as discussed in Section 4.3. We choose $\varepsilon = 2h$ and fix the initial condition to be $u_0 = 0 + \eta$, where η is a small random perturbation across the surface. Our choice of ε is similar to that used in [54] for Cahn-Hilliard with constant mobility and in [18] for Cahn-Hilliard with degenerate mobility. Our results have transition layers approximately four to six grid cells thick. We compute on a $120 \times 120 \times 120$ grid with cell width $h = \frac{1}{30}$. The time step taken is $k = \frac{1}{10}h^2$. We observe the spinodal decomposition expected; see Figure 5.

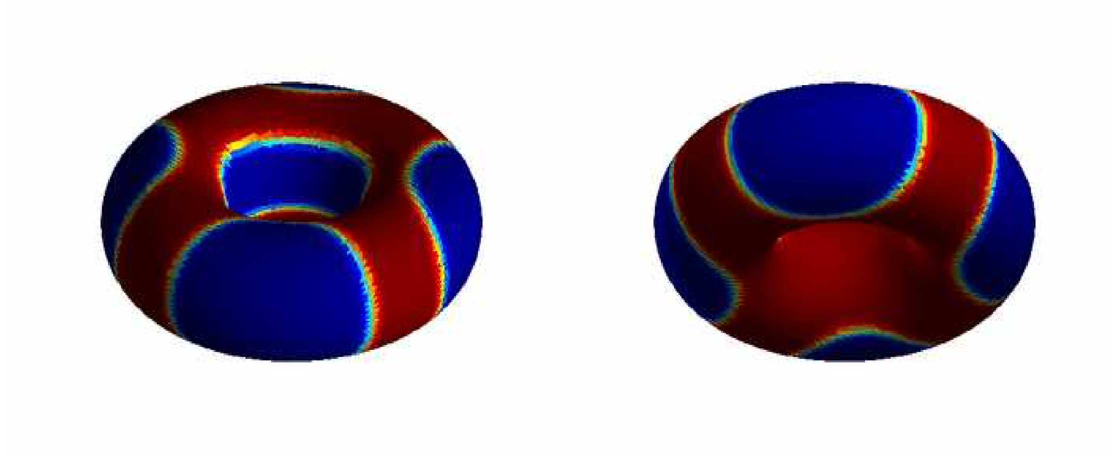
Figure 6 shows spinodal decomposition in the Cahn-Hilliard equation on the smoothed bunny. We use ADI to compute (35) as in Section 6. We are constrained to take the same time step as in the linear problem, $k = O(h^3)$. Again $\varepsilon = 2h$ and $u_0 = 0 + \eta$, where η is a small zero mean perturbation.



$t = 0.12$

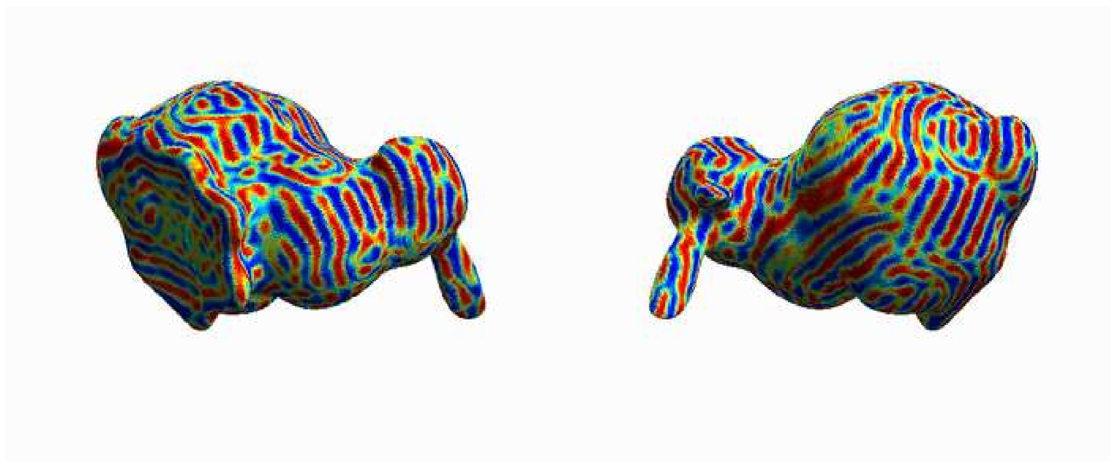


$t = 0.45$

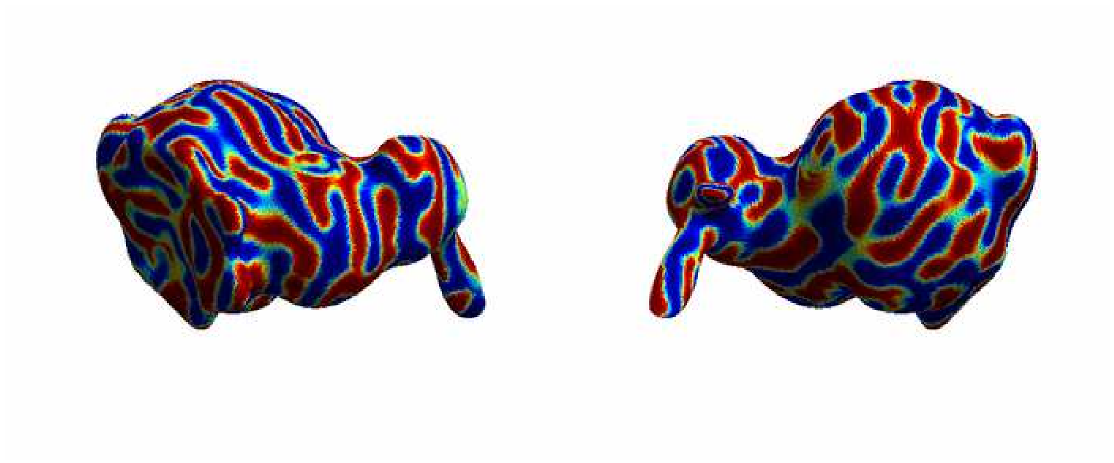


$t = 3.5$

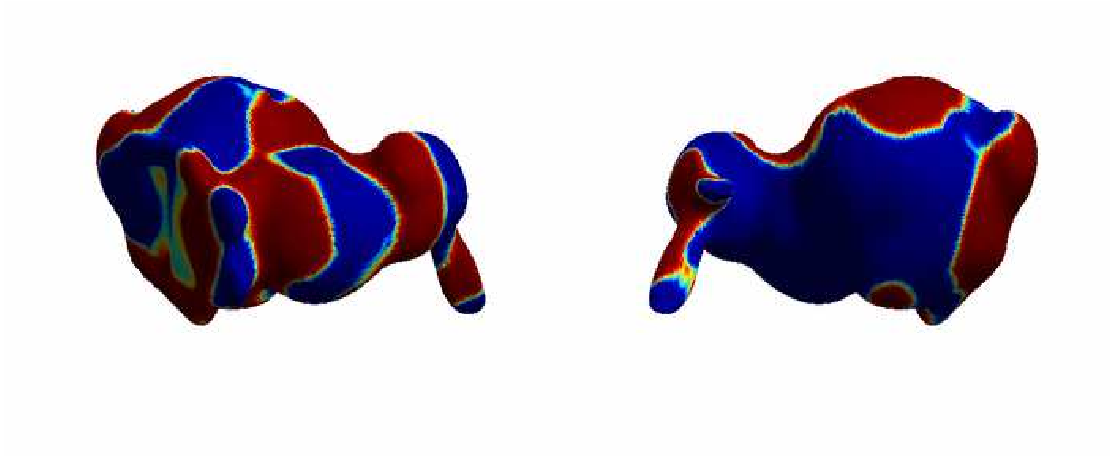
Figure 5: Spinodal decomposition in the Cahn-Hilliard equation. Initial condition is $u = 0$ (slightly perturbed), red denotes $u = 1$ and blue denotes $u = -1$. Each row shows two views of the surface at the same point in time.



$t = 3$



$t = 20$



$t = 950$

Figure 6: Spinodal decomposition in the Cahn-Hilliard equation. Initial condition is $u = 0$ (slightly perturbed), red denotes $u = 1$ and blue denotes $u = -1$. Each row shows two views of the surface at the same point in time.

8 Thin Film Fluid Flow

An interesting class of problems described by nonlinear fourth order diffusion equations are thin film flows involving a layer of viscous liquid on a solid surface. Surface tension forces lead to the fourth order motion, due to curvature affects on the air-liquid interface of the film [7, 34, 42]. In the case of such flows on curved surfaces, the underlying curvature of the substrate plays a role in the motion of the film, with flow out of regions of high curvature. Recently there has been some interest in the fluids community in deriving equations of motion for such flows on general surfaces. The most comprehensive paper to date on this problem is that of Roy, Roberts and Simpson [47], building on previous work of Schwartz and Weidner [48]. Several authors have developed numerical methods for thin films on surfaces using coordinates on the surface. This work includes flows on cylinders [24, 48, 55] and more general surfaces [37], as well as icing of airplane wings [39, 35, 36, 38]. We develop a numerical method for thin films on general surfaces using the implicit representation methodology.

Letting u denote the film thickness, Roy, Roberts, and Simpson derived the following for the dimensionless flow of a thin film in the absence of gravity [47]:

$$\frac{\partial \zeta}{\partial t} = -\frac{1}{3}\nabla_S \cdot \left[u^2 \zeta \nabla_S \tilde{\kappa} - \frac{1}{2} u^4 (\kappa \mathrm{I} - \mathrm{K}) \cdot \nabla_S \kappa \right]. \tag{36}$$

K is the curvature tensor for the curved substrate. The quantities k_1 and k_2 are the principle curvatures of the substrate, and $\kappa = k_1 + k_2$ is twice the mean curvature. $\tilde{\kappa}$ is curvature of the free surface, which we approximate as

$$\tilde{\kappa} = \kappa + (k_1^2 + k_2^2) + \Delta_S u.$$

The variable ζ is the amount of fluid above a surface patch, which we approximate by $\zeta = u - \frac{1}{2}\kappa u^2 + \frac{1}{3}k_1 k_2 u^3$.

Using the framework of our level set function ϕ, we can easily compute κ and K, as well as the quantities k_1, k_2, and $k_1^2 + k_2^2$. We note that K is the Jacobian of the Gauss Map of the substrate. In terms of the implicit representation, the Gauss Map is $\frac{\nabla\phi}{|\nabla\phi|}$. The trace of K gives $-\kappa$, and the determinant of $\mathrm{A} = \mathrm{K} + \frac{\nabla\phi \otimes \nabla\phi}{|\nabla\phi|^2}$ gives the Gauss curvature, $k_1 k_2$. We finally note that

$$k_1^2 + k_2^2 = \kappa^2 - 2k_1 k_2.$$

This model differs from both linear diffusion and the Cahn-Hilliard equation in that the highest order term of (36) is nonlinear. Our convexity splitting must take this nonlinearity into account. To derive the evolution scheme, we carry out the differentiation on the left of (36) to get an evolution equation for u :

$$\frac{\partial u}{\partial t} = -\frac{1}{3\zeta'}\nabla_S \cdot \left[u^2 \zeta \nabla_S \tilde{\kappa} - \frac{1}{2} u^4 (\kappa \mathrm{I} - \mathrm{K}) \cdot \nabla_S \kappa \right], \tag{37}$$

where $\zeta' = 1 - \kappa u + \kappa_1 \kappa_2 u^2$. Following our earlier examples, we pick an appropriate C and evolve

$$\frac{u^{n+1} - u^n}{k} = -C^n \Delta\Delta u^{n+1} + C^n \Delta\Delta u^n - F(u^n), \tag{38}$$

where $F(u^n)$ denotes the right hand side of (37). The dependence of C on the time step reflects the dependence of the highest order term of (37) on u^n. Noting that the highest order term of $F(u^n)$ is $\frac{1}{3\zeta'}u^2\zeta\Delta_S^2 u^n$, we see that choosing $C^n < \frac{1}{3\zeta'}u^2\zeta$ might cause the method to be unstable. On the other hand, picking C too large significantly slows the dynamics. So again using S_c to denote the computational band $|\phi_{i,j,k}| \leq c$, we pick

$$C^n = \max_{x_{i,j,k} \in S_c} \frac{1}{3(1 - \kappa u^n_{i,j,k} + \kappa_1 \kappa_2 (u^n_{i,j,k})^2)} \left(u^n_{i,j,k}\right)^2 \zeta^n_{i,j,k}. \tag{39}$$

8.1 Example: Thin Film on an Ellipse

The curvature dependent terms in (36) reflect fluid motion driven by curvature of the surface. Fluid builds up in regions of high negative curvature while leaving areas of high positive curvature. We consider an example discussed in [47] and study the flow of an initially constant layer of film on an ellipse. The authors of [47] discussed the plausible motion of the fluid, however they did not compute (36) for this example. We provide numerical evidence supporting their claims for the fluid dynamics.

We place a layer of fluid with constant thickness $u = 0.02$ on the outside of the ellipse

$$x^2 + 3y^2 = 2. \tag{40}$$

We computed on $[-2,2] \times [-2,2]$ with 200×200 grid points. We used a constant time step $k = 10h^2$ where h denotes the grid cell width. We used the algorithm in Section 5 with only one change: C is updated dynamically as described above. As seen in Figure 7, the fluid moves away from the ellipse's regions of high curvature near $y = 0$.

Now consider placing the same constant layer of fluid on the inside of the ellipse (40). Once the implicit function ϕ for the above problem is defined, we easily adjust our simulation to this case by mapping $\phi \to -\phi$. On the inside of the ellipse, the film builds up in the areas of high curvature. See Figure 8.

8.2 Sphere with Gravity: Fingering Instability

The following model, derived in [47], includes the effects of gravity:

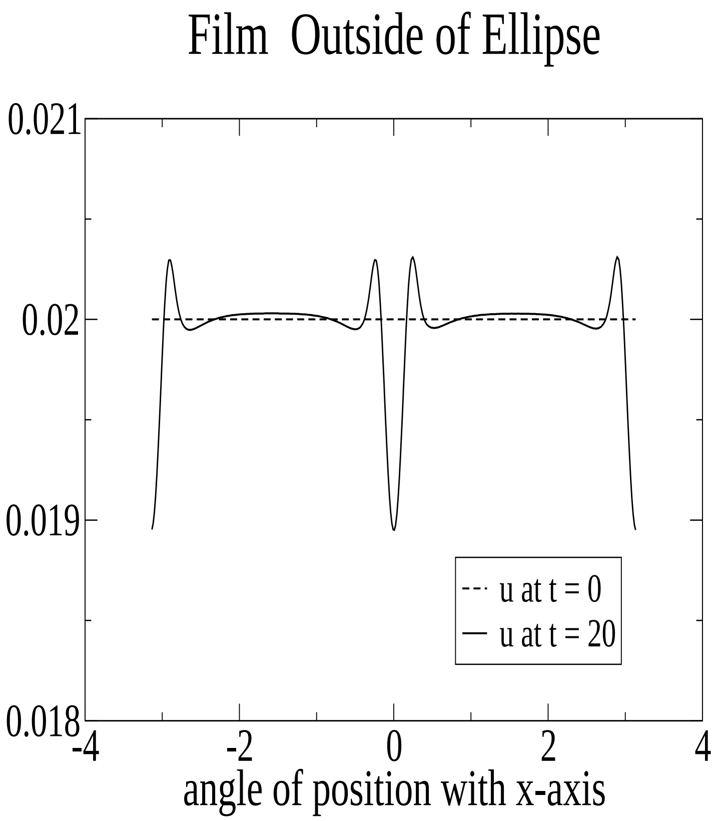
$$\begin{aligned} \frac{\partial \zeta}{\partial t} &= -\frac{1}{3}\nabla_S \cdot \left[u^2 \zeta \nabla_S \tilde{\kappa} - \frac{1}{2} u^4 (\kappa \mathbf{I} - \mathrm{K}) \cdot \nabla \kappa \right] \\ &\quad -\frac{1}{3} Bo \nabla_S \cdot \left[u^3 \hat{\mathbf{g}}_{\mathbf{s}} - u^4 \left(\kappa \mathrm{I} + \frac{1}{2}\mathrm{K} \right) \cdot \hat{\mathbf{g}}_{\mathbf{s}} + \hat{g}_n u^3 \nabla_S u \right]. \end{aligned} \tag{41}$$

The Bond number Bo quantifies the relative strength of gravity to surface tension, and it is given by

$$Bo = \rho g H^2 \sigma,$$

where ρ is the fluid density, g is acceleration due to gravity, σ is the surface tension, and H is the characteristic thickness of the film. The vector $\hat{\mathbf{g}}_{\mathbf{s}}$ is the component of gravity tangent to the surface S, and $\hat{g}_n$ is the magnitude of the component of gravity normal to S. In our example, $\hat{g} = -(0,0,1)$, so $\hat{g}_s = \mathrm{P}\hat{g}$ and $\hat{g}_n = |\hat{g} - \hat{g}_s|$.

We place a ring of fluid with sinusoidally varying height near the top of the sphere. The remainder of the sphere has a precursor layer of thickness $u = 5 \times 10^{-3}$. Gravity drives the fluid to the bottom of the sphere while the non-constant thickness causes a fingering effect. We used up-winding to compute the gravity terms in (41). We computed in $[-2,2] \times [-2,2] \times [-2,2]$ on a $128 \times 128 \times 128$ grid. We used a constant time step $k = \frac{1}{2}h^2$ where h denotes the grid spacing. We compute in a band that is about ten grid cells wide and re-extend data off of the surface after every five time steps. The Bond number is $Bo = 100$ in this simulation. The results are given in figures 9, 10, and 11. Note that the fluid drips down the sphere and eventually begins to collect at the bottom of the sphere. Equation (41) is derived from lubrication theory and thus does not capture later time dynamics in which fluid might drop off the bottom of the sphere. Although the computation is performed on a spherical surface, the method is very general and can easily be applied to surfaces that lack the symmetry of the sphere.



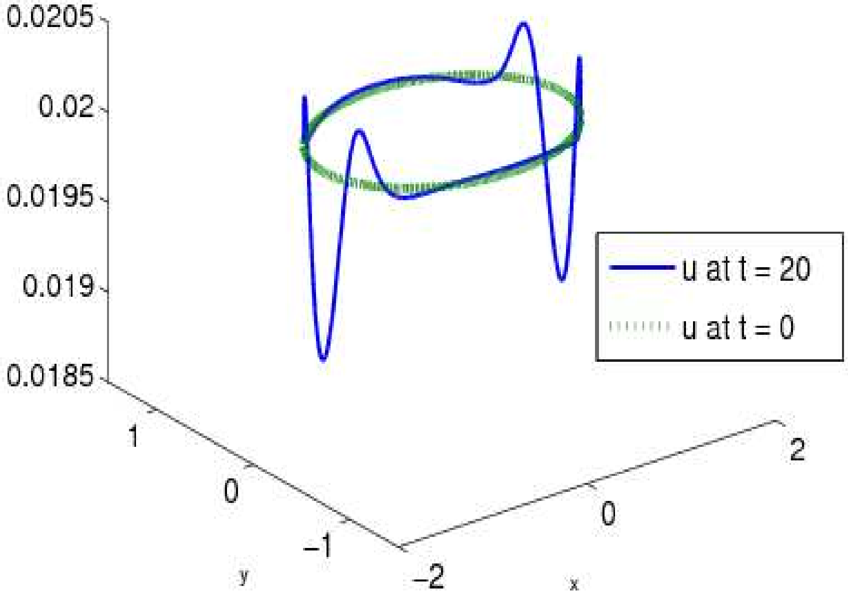
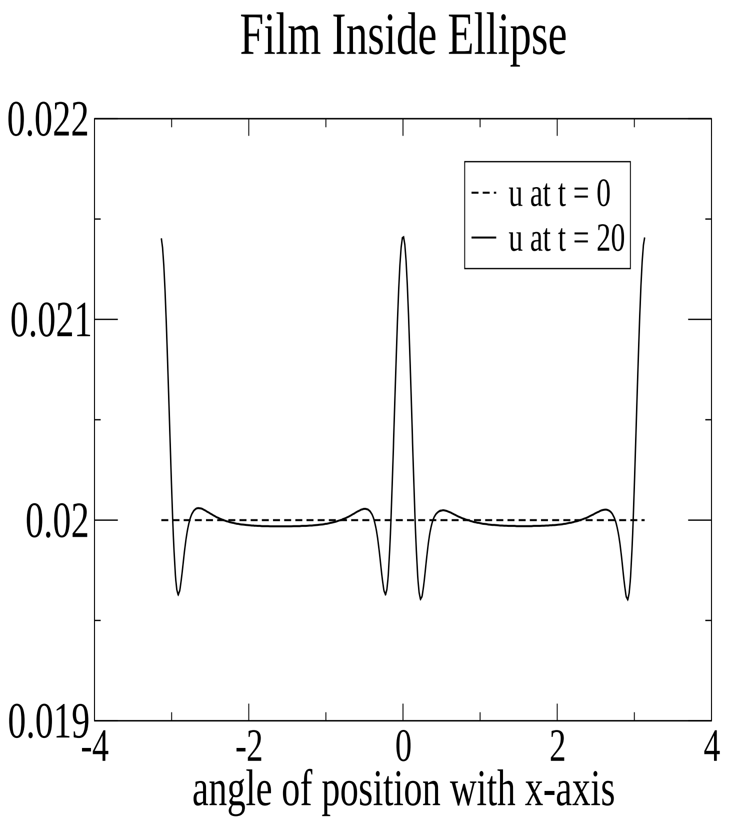


Figure 7: Thin film driven by curvature on the outside of an ellipse.



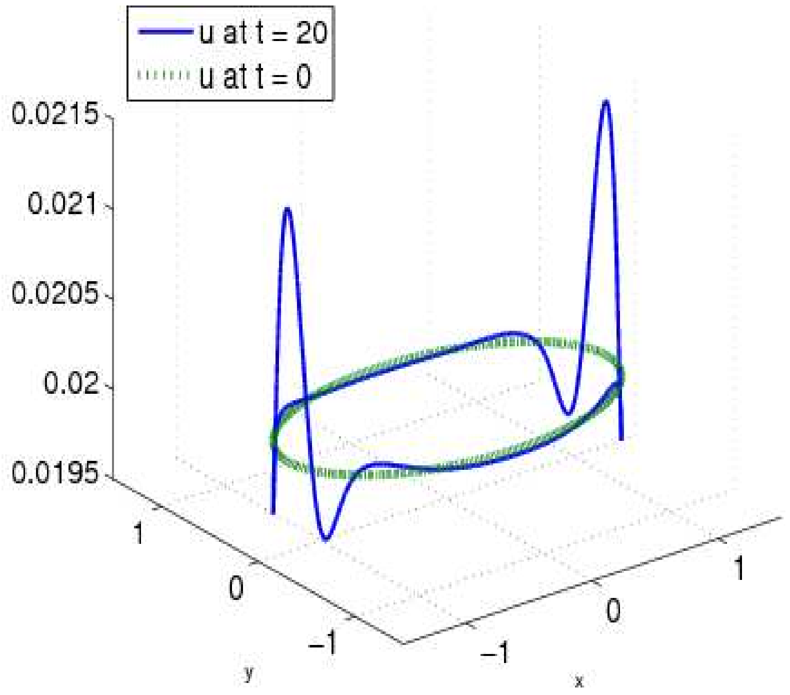
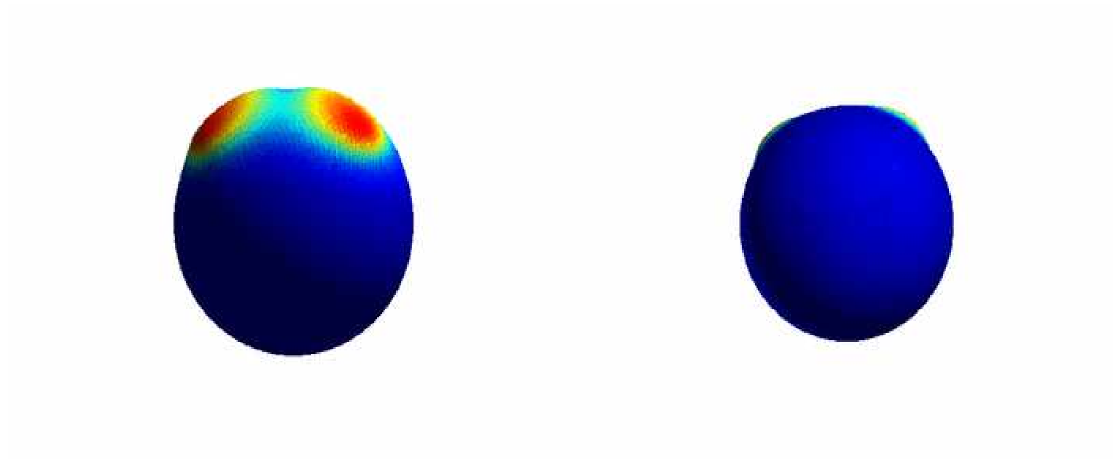
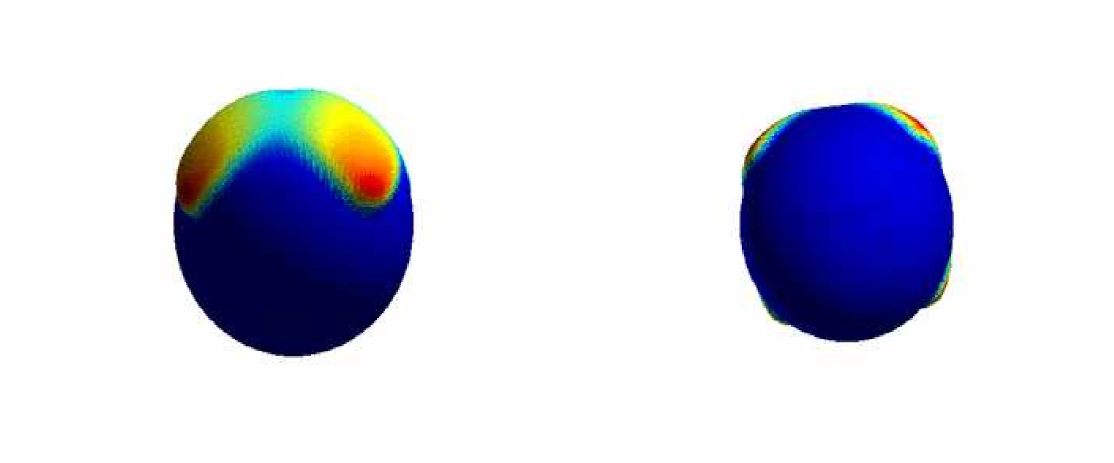


Figure 8: Thin film driven by curvature on the inside of an ellipse.



$t = 0.0$



$t = 1.0$

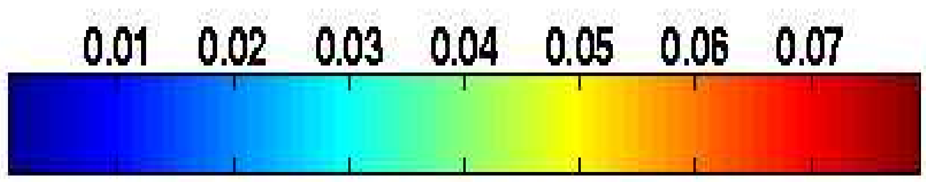
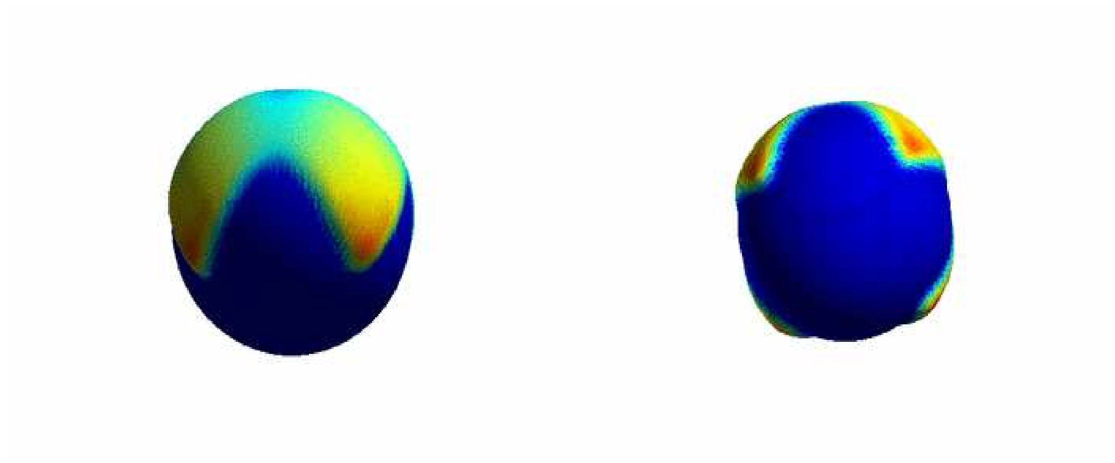
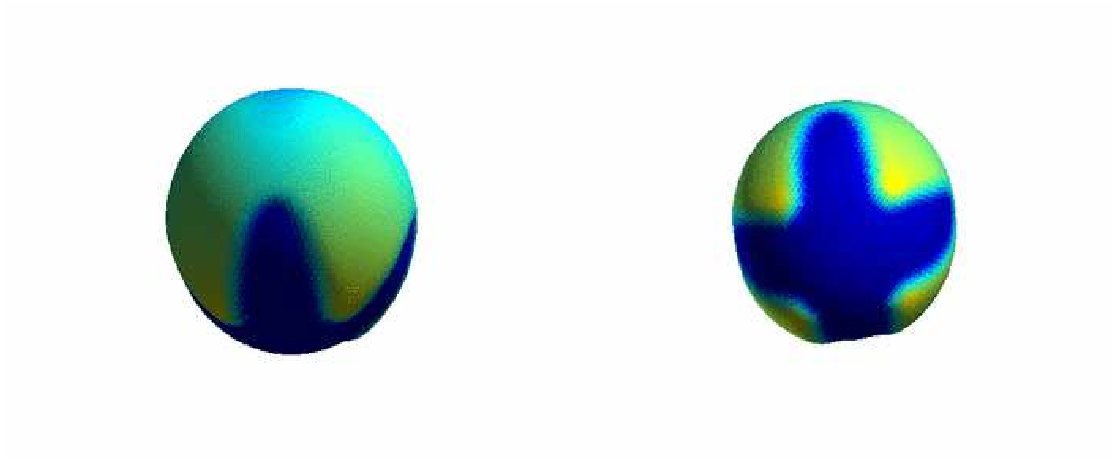


Figure 9: Equation (41) solved on the sphere. Each row gives a side and bottom view of the sphere at the same point in time. The sphere is covered with a precursor layer of thickness of $u = 5 \times 10^{-3}$. The color scheme is used to display fi lm thickness.



$t = 2.0$



$t = 4.0$

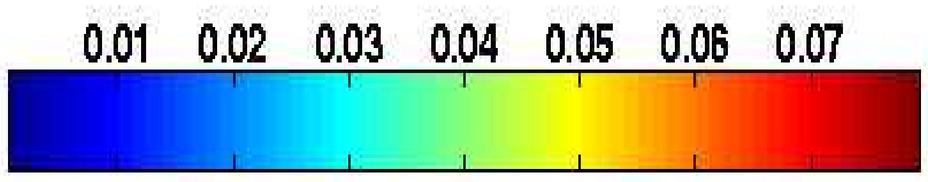
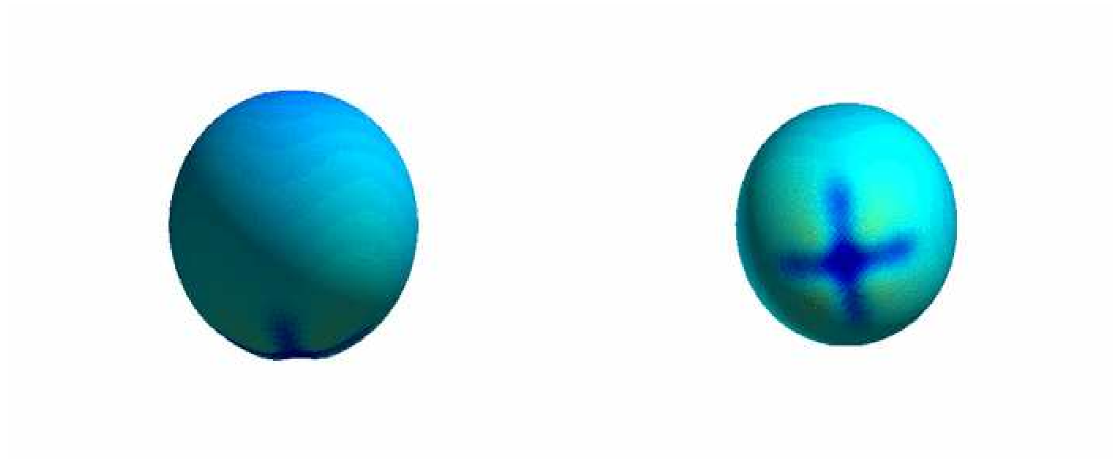
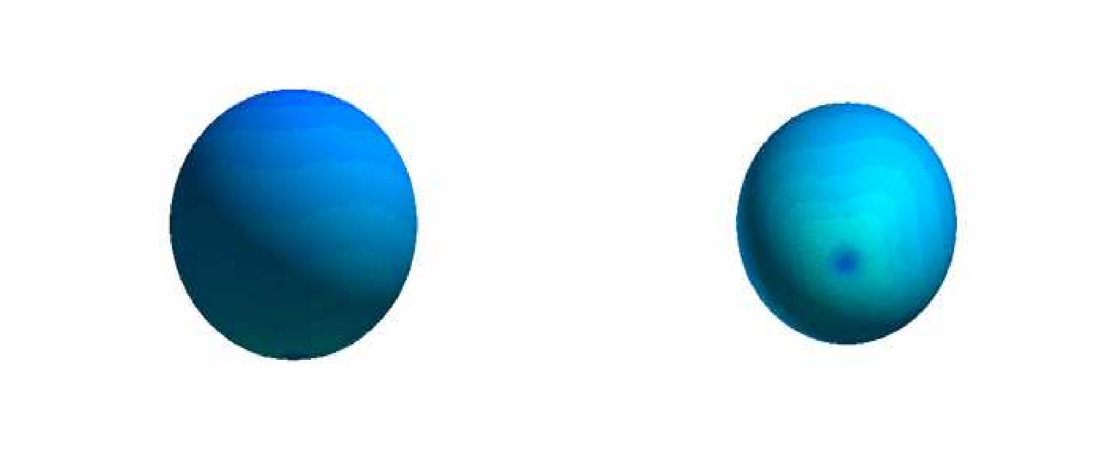


Figure 10: Equation (41) solved on the sphere. Each row gives a side and bottom view of the sphere at the same point in time. The sphere is covered with a precursor layer of thickness of $u = 5 \times 10^{-3}$. The color scheme is used to display fi lm thickness.



$t = 10.0$



$t = 16.0$

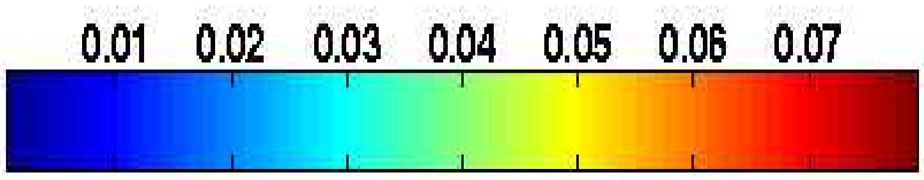


Figure 11: Later dynamics of equation (41) solved on the sphere. Each row gives a side and bottom view of the sphere at the same point in time. The sphere is covered with a precursor layer of thickness of $u = 5 \times 10^{-3}$. The color scheme is used to display fi lm thickness.

9 Conclusions

We have presented a method for solving fourth order PDEs on surfaces of arbitrary geometry with finite difference schemes on a Cartesian mesh. With our methods, the same code can be easily applied to many different surfaces by changing only the function implicitly defining the surface. In computing higher order PDEs on surfaces, we derive equations that are solved in an augmented band around the surface in Euclidean space. As PDEs on $\mathbb{R}^N$, these equations are unique in the severity of their degeneracy – we know of no other fourth order diffusions being studied where all diffusion locally is in one coordinate direction only.

The higher order equations introduce a number of challenges that are of little consequence for first or second order PDEs. Stability restrictions for time stepping fourth order equations require using implicit schemes, unlike most previous work on solving PDEs on implicit surfaces [2, 6, 32]. We derived semi-implicit schemes using convexity splitting ideas explored in [16, 18, 51] and presented a new means of combining convexity splitting schemes with ADI methods. Compared to lower order equations, fourth order PDEs require more careful extension and re-extension of data off of the surface, and they have more complicated boundary conditions. We discussed each of these issues in detail and applied our methods to linear fourth order diffusion, the Cahn-Hilliard equation, and a recently derived model for surface tension driven flows on curved substrates.

Our work is only a first step. Although our schemes are faster than explicit schemes, there is room for improvement. It remains a difficult problem to improve the time step restriction for flows on complex surfaces with high curvature, because the inherent geometry is embedded in the projection operator.

It is an interesting problem to develop schemes with higher numerical accuracy, in particular for the time step, as this might allow for computations with a larger time step. If the geometry introduces stiffness into the dynamics, these terms could also be dealt with using splitting methods. Local mesh refinement, while still taking advantage of the Cartesian geometry, might also help address this issue.

It is our hope that the techniques developed in this paper will be useful for scientists interested in computing higher order PDEs on complicated geometries. Thin film flow is a particularly relevant problem where there is interest in modeling the liquid lining of the lungs, icing of airplane winds, and numerous other problems in which the underlying surface geometry is complex.

A Spatial Discretization

We now present the specific discretizations for the gradient, Laplacian, and Laplace-Beltrami operators. We compute everything on a Cartesian grid in $\mathbb{R}^N$. Assume the domain is given by $0 \le x_i \le L$ for $1 \le i \le N$. We use a cubic grid with $\Delta x_i = h$ for each i. Let $\omega = (\omega_i)$ be a vector of integers $0 \le \omega_i \le \frac{L}{h}$ and denote the coordinate directions by $\mathbf{e}_i$, so that each grid point is given by

$$x_\omega = \sum_{0 \le i \le N} h\omega_i \mathbf{e}_i.$$

Our discretizations follow the example of [6]. We calculate $\nabla\phi$ with centered differences,

$$\nabla\phi(x_\omega) = \frac{1}{2h}(\phi(x_\omega + h\mathbf{e}_i) - \phi(x_\omega - h\mathbf{e}_i))_i, \tag{42}$$

but use forward differences to compute gradients for everything else:

$$\nabla u(x_\omega) = \frac{1}{h}(u(x_\omega + h\mathbf{e}_i) - u(x_\omega))_i. \tag{43}$$

Let $\mathbf{v}(x_\omega)$ be a vector whose ith component is $v^i(x_\omega)$. We compute the divergence of $\mathbf{v}$ using backward differences:

$$\nabla \cdot \mathbf{v}(x_\omega) = \frac{1}{h} \sum_{0 \le i \le N} v^i(x_\omega) - v^i(x_\omega - h\mathbf{e}_i). \tag{44}$$

The Laplacian of a scalar quantity u is computed by applying (44) to (43), and the biharmonic is computed by repeated application of the Laplacian.

Projected derivatives are computed using appropriate averaging of the projection matrix. Given a projection matrix $\mathrm{P} = (p^{ij})$, we compute

$$\mathrm{P}\nabla u(x_\omega) = \Big(\sum_{1 \le j \le N} \overline{p}^{ij} u_{x_j}(x_\omega) \Big)_i, \tag{45}$$

where

$$\overline{p}^{ij} u_{x_j}(x_\omega) = \frac{1}{2h} (p^{ij}(x_\omega + h\mathbf{e}_j) + p^{ij}(x_\omega))(u(x_\omega + h\mathbf{e}_j) - u(x_\omega)).$$

Assuming that ϕ is a signed distance function, by applying (44) to (45) we compute

$$\Delta_S u(x_\omega) = \nabla \cdot (\mathrm{P}\nabla u(x_\omega)) = \sum_{1 \le i,j \le N} \frac{\overline{p}^{ij} u_{x_j}(x_\omega) - \overline{p}^{ij} u_{x_j}(x_\omega - h\mathbf{e}_i)}{h}. \tag{46}$$

We repeat (46) to calculate $\Delta_S \Delta_S u$.

B Degeneracy and ADI

To demonstrate the strong degeneracy of the fourth order equations considered in this paper, we apply standard ADI schemes to both (7) and (9) without the assistance of the operator splitting discussed in Section 4.1. Our calculations suggest the need for an indirect method (such as convexity splitting), while also showing that the second order problem is far less delicate than its higher order counterpart.

We examine the special case where S is the line in $\mathbb{R}^2$ making an angle of θ with the x-axis. We first discuss the ADI method applied to second order diffusion on the line, then continue with a discussion of ADI for intrinsic fourth order diffusion on the same line. We show that the stability of the fourth order diffusion scheme depends on θ, which hardly affects the second order diffusion scheme. For simplicity we discuss only the two dimensional problem, but note that similar results hold for three dimensions.

B.1 ADI without Convexity Splitting

To apply ADI to equations (7) and (9) without convexity splitting, we follow the example of [56] and compute terms with mixed derivatives explicitly and all remaining terms implicitly. We first examine equation (7), leaving equation (9) for Section B.2. Letting S be the line making angle θ with the x-axis, our distance function is

$$\phi = -\sin\theta\, x + \cos\theta\, y,$$

and the projection matrix is

$$\mathrm{P} = \begin{bmatrix} \cos^2\theta & \cos\theta\sin\theta \\ \cos\theta\sin\theta & \sin^2\theta \end{bmatrix}. \tag{47}$$

For this example, (5) simplifies to

$$u_t = \cos^2\theta\, u_{xx} + 2\cos\theta\sin\theta\, u_{xy} + \sin^2\theta\, u_{yy}. \tag{48}$$

Define

$$D_x = \cos^2\theta\,\partial_{xx},$$

$$D_y = \sin^2\theta\,\partial_{yy},$$

$$L_x = I - kD_x,$$

and

$$L_y = I - kD_y.$$

Note that $L_xL_yu^n = (I - kD_x - kD_y + k^2D_xD_y)$ so ignoring the term of $O(k^2)$, we see that L_xL_y approximates (48) without its mixed partials. We now introduce $O(k)$ errors by computing the mixed partials explicitly. This can be done more accurately so that these errors are reduced – see [56]. So we have

$$L_xL_y\,u^{n+1} = -2k\cos\theta\sin\theta\,u^n_{xy}.$$

We know subtract $L_xL_yu^n$ from both sides of the above equation and let $v^{n+1} = u^{n+1} - u^n$ to get

$$L_xL_yv^{n+1} = k\nabla\cdot(\mathrm{P}\nabla u^n) = k\left(\cos^2\theta\,u^n_{xx} + 2\cos\theta\sin\theta\,u^n_{xy} + \sin^2\theta\,u^n_{yy}\right). \tag{49}$$

We use Von Neumann stability analysis to study scheme (49) for this example. Solutions of the continuous in space and discrete in time solutions of (49) can be written as

$$u^n(x,y) = \sigma(\alpha)^n\exp i\,(\alpha_1x + \alpha_2y), \tag{50}$$

where the superscript n on the left corresponds to the n-th time step, while the superscript on the right corresponds to a power of σ. Following the example of [56], we rescale the wave number as $\hat{\alpha} = \frac{1}{k^2}\alpha$ and substitute (50) into (49) to get

$$\sigma(\hat{\alpha}) - 1 = -\frac{(\cos\theta\alpha_1 + \sin\theta\alpha_2)^2}{1 + \cos^2\theta\alpha_1^2 + \sin^2\theta\alpha_2^2 + \sin^2\theta\cos^2\theta\alpha_1^2\alpha_2^2}.$$

We see (49) is unconditional stable, since

$$\frac{(\cos\theta\alpha_1 + \sin\theta\alpha_2)^2}{1 + \cos^2\theta\alpha_1^2 + \sin^2\theta\alpha_2^2 + \sin^2\theta\cos^2\theta\alpha_1^2\alpha_2^2} \;\leq\; 2 \tag{51}$$

is satisfied for all α and θ. Though simpler than (7) on general surfaces, this example shows that (49) is stable even when the diffusion is not in the direction of the x or y axes. We next show that the fourth order problem is very different, as it is too degenerate for this type of ADI scheme.

B.2 Fourth Order Problem

We now discretize (9) as in [56]. Define

$$D_x = \cos^4\theta\,\partial_{xxxx},$$

$$D_y = \sin^4\theta\,\partial_{yyyy},$$

$$L_x = I + kDx,$$

and

$$L_y = I + kDy.$$

Once again letting $v^{n+1} = u^{n+1} - u^n$, the scheme

$$L_x L_y v^{n+1} = -k\nabla \cdot (\mathrm{P}\nabla \cdot (\mathrm{P}\nabla u^n)) \tag{52}$$

is an $O(k)$ approximation of (9).

We repeat the Von Neumann analysis used for (49), this time rescaling $\hat{\alpha} = \frac{1}{k^4}\alpha$ to get

$$\sigma(\hat{\alpha}) - 1 = -\frac{(\cos\theta\alpha_1 + \sin\theta\alpha_2)^4}{1 + \cos^4\theta\alpha_1^4 + \sin^4\theta\alpha_2^4 + \sin^4\theta\cos^4\theta\alpha_1^4\alpha_2^4}.$$

Stability of (52) thus requires

$$\frac{(\cos\theta\alpha_1 + \sin\theta\alpha_2)^4}{1 + \cos^4\theta\alpha_1^4 + \sin^4\theta\alpha_2^4 + \sin^4\theta\cos^4\theta\alpha_1^4\alpha_2^4} \le 2 \tag{53}$$

Note that (53) is clearly satisfied when $\sin\theta = 0$ or $\cos\theta = 0$, which corresponds to diffusion in the direction of one of the coordinate axes. To show that (52) is only conditionally stable for certain choices of θ we consider the special case $\theta = \frac{\pi}{2}$, so we require

$$\frac{(\alpha_1 + \alpha_2)^4}{4 + \alpha_1^4 + \alpha_2^4 + \frac{1}{4}\alpha_1^4\alpha_2^4} \le 2.$$

This is not satisfied, for example, when $\alpha_1 = \alpha_2 = 1$. The instability is easily demonstrated in numerical simulations and avoiding it requires using time steps on the same order of magnitude as an explicit scheme, $k = O(h^4)$.

References

[1] D. Adalsteinsson and J. A. Sethian. A fast level set method for propagating interfaces. *J. Comput. Phys.*, 118(2):269–277, 1995.

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

[3] C. Bajaj and G. Xu. Anisotropic diffusion of subdivision surfaces and functions on surfaces. *ACM Transactions on Graphics*, 22(1):4–32, 2003.

[4] J. W. Barrett and J. F. Blowey. Finite element approximation of the Cahn-Hilliard equation with concentration dependent mobility. *Math. Comp.*, 68:487–517, 1999.

[5] J. W. Barrett, J. F. Blowey, and H. Garcke. Finite element approximation of a fourth order degenerate parabolic equation. *Numer. Math.*, 80(4):525–556, October 1998.

[6] M. Bertalmío, L. T. Cheng, S. Osher, and G. Sapiro. Variational problems and partial differential equations on implicit surfaces. *J. Comput. Phys.*, 174(2):759–780, 2001.

[7] A. L. Bertozzi. The mathematics of moving contact lines in thin liquid films. *Notices of the American Math. Soc.*, 45(6):689–697, 1998.

[8] D. L. Chopp and J. A. Sethian. Motion by intrinsic Laplacian of curvature. *Interfaces Free Bound.*, 1(1):107–123, 1999.

[9] U. Clarenz, U. Diewald, and M. Rumpf. Processing textured surfaces via anisotropic geometric diffusion. *IEEE Transactions on Image Processing*, 13(2):248–261, 2004.

[10] U. Clarenz, M. Rumpf, and A. Telea. Finite elements on point based surfaces. *Computers and Graphics*, 2004. to appear.

[11] B. D. Coleman, R. S. Falk, and M. Moakher. Space-time fi nite element methods for surface diffusion with applications to the theory of the stability of cylinders. *SIAM J. Sci. Comput.*, 17(6):1434–1448, 1996.

[12] S. D. Conte. Numerical solution of vibration problems in two space variables. *Pacific J. Math*, 7:1535–1544, 1957.

[13] S. D. Conte and R. T. Dames. An alternating direction method for solving the biharmonic equation. *Math. Tables Aids Comput.*, 12:198–205, 1958.

[14] S. D. Conte and R. T. Dames. On an alternating direction method for solving the plate problem with mixed boundary conditions. *J Assoc. Comput. Mach.*, 7:264–273, 1960.

[15] M. Droske and M. Rumpf. A level set formulation for Willmore fbw. *Interfaces and Free Boundaries*, 6(3):361–378, 2004.

[16] D. J. Eyre. Unconditionally gradient stable time marching the Cahn-Hilliard equation. In *Computational and mathematical models of microstructural evolution (San Francisco, CA, 1998)*, volume 529 of *Mater. Res. Soc. Sympos. Proc.*, pages 39–46. MRS, Warrendale, PA, 1998.

[17] S. F. Frisken, R. N. Perry, A. Rockwood, and T. Jones. Adaptively sampled fi elds: A general representation of shape for computer graphics. *ACM SIGGRAPH*, 2000.

[18] K. Glasner. A diffuse interface approach to Hele-Shaw fbw. *Nonlinearity*, 16(1):49–66, 2003.

[19] G. Grün and M. Rumpf. Nonnegativity preserving convergent schemes for the thin fi lm equation. *Numer. Math.*, 87:113–152, 2000. [delete].

[20] G. Grün and M. Rumpf. Simulation of singularities and instabilities arising in thin fi lm fbw. *Europ. Appl. Math.*, 12:293–320, 2001.

[21] D. Halpern, O. E. Jensen, and J. B. Grotberg. A theoretical study of surfactant and liquid delivery into the lung. *J. Appl. Physiology*, 85:333–352, 1998.

[22] M. Hofer and H. Pottmann. Energy-minimizing splines in manifolds. *ACM Trans. Graphics*, 2004.

[23] H. Hoppe and M. Eck. Automatic reconstruction of b-spline surfaces of arbitrary topological type. *ACM SIGGRAPH*, 1996.

[24] A. E. Hosoi and L. Mahadevan. Axial instability of a free-surface front in a partially-fi lled horizontal rotating cylinder. *Phys. Fluids*, 11(1), 1999.

[25] G. Jiang and D. Peng. Weighted ENO schemes for Hamilton-Jacobi equations. *SIAM J. Sci. Comput.*, 21(6):2126–2143 (electronic), 2000.

[26] G. Jiang and C. W. Shu. Effi cient implementation of weighted ENO schemes. *J. Comput. Phys.*, 126(1):202–228, 1996.

[27] R. Kimmel and J. A. Sethian. Computing geodesic paths on manifolds. *Proc. Natl. Acad. Sci. USA*, 95(15):8431–8435 (electronic), 1998.

[28] D. Kincaid, J. Respess, and D. Young. Itpack 2c: A fortran package for solving large sparse linear systems by adaptive accelerated iterative methods. http://rene.ma.utexas.edu/CNA/ITPACK/.

[29] V. Krishnamurthy and M. Levoy. Fitting smooth surfaces to dense polygon meshes. *ACM SIGGRAPH*, pages 313–324, 1996.

[30] Stanford University Computer Graphics Laboratory. Stanford bunny. http://graphics.stanford.edu/data/3Dscanrep/.

[31] F. Mémoli and G. Sapiro. Fast computation of weighted distance functions and geodesics on implicit hyper-surfaces. *J. Comput. Phys.*, 173(2):730–764, 2001.

[32] F. Mémoli, G. Sapiro, and S. Osher. Solving variational problems and partial differential equations mapping into general target manifolds. *J. Comput. Phys.*, 195(1):263–292, 2004.

[33] F. Memoli, G. Sapiro, and P. Thompson. Implicit brain imaging. *Human Brain Mapping*, 23:179–188, 2004.

[34] T. G. Myers. Thin fi lms with high surface tension. *SIAM Rev.*, 40(3):441–462 (electronic), 1998.

[35] T. G. Myers. Extension to the messinger model for aircraft icing. *AIAA Journal*, 39(2):211–218, 2001.

[36] T. G. Myers and J. P. F. Charpin. A mathematical model for atmospheric ice accretion and water fbw on a cold surface. *International Journal of Heat and Mass Transfer*, 47(25):5483–5500, 2004.

[37] T. G. Myers, J. P. F. Charpin, and S. J. Chapman. The fbw and solidifi cation of a thin fluid fi lm on an arbitrary three-dimensional surface. *Phys. Fluids*, 14(8):2788–2803, 2002.

[38] T. G. Myers, J. P. F. Charpin, and C. P. Thompson. Slowly accreting ice due to supercooled water impacting on a cold surface. *Physics of Fluids*, 14(1):240–256, 2002.

[39] T. G. Myers and D. W. Hammond. Ice and water fi lm growth from incoming supercooled droplets. *International Journal of Heat and Mass Transfer*, 42(12):2233–2242, 1999.

[40] A. Novick-Cohen and R. L. Pego. Stable patterns in a viscous diffusion equation. *Trans. Amer. Math. Soc.*, 324(1):331–351, 1991.

[41] A. Novick-Cohen and L. A. Segal. Nonlinear aspects of the Cahn-Hilliard equation. *Physica D*, 10:277–298, 1984.

[42] A. Oron, S. H. Davis, and S. G. Bankoff. Long-scale evolution of thin liquid fi lms. *Rev. Mod. Phys.*, 69(3):931–980, July 1997.

[43] S. Osher and R. Fedkiw. *Level set methods and dynamic implicit surfaces*, volume 153 of *Applied Mathematical Sciences*. Springer-Verlag, New York, 2003.

[44] R. L. Pego. Front migration in the nonlinear Cahn-Hilliard equation. *Proc. Roy. Soc. London Ser. A*, 422(1863):261–278, 1989.

[45] D. Peng, B. Merriman, S. Osher, H. Zhao, and M. Kang. A PDE-based fast local level set method. *J. Comput. Phys.*, 155(2):410–438, 1999.

[46] T. Preußer and M. Rumpf. A level set method for anisotropic geometric diffusion in 3D image processing. *SIAM J. Appl. Math.*, 62(5):1772–1793, 2002.

[47] R. Valéry Roy, A. J. Roberts, and M. E. Simpson. A lubrication model of coating flows over a curved substrate in space. *J. Fluid Mech.*, 454:235–261, 2002.

[48] L.W. Schwartz and D.E. Weidner. Modeling of coating flows on curved surfaces. *Journal of Engineering Mechanics*, 29:91–103, 1995.

[49] J. A. Sethian. *Level set methods and fast marching methods*, volume 3 of *Cambridge Monographs on Applied and Computational Mathematics*. Cambridge University Press, Cambridge, second edition, 1999. Evolving interfaces in computational geometry, fluid mechanics, computer vision, and materials science.

[50] L. Simon. *Lectures on geometric measure theory*, volume 3 of *Proceedings of the Centre for Mathematical Analysis, Australian National University*. Australian National University Centre for Mathematical Analysis, Canberra, 1983.

[51] P. Smereka. Semi-implicit level set methods for curvature and surface diffusion motion. *J. Sci. Comput.*, 19(1-3):439–456, 2003. Special issue in honor of the sixtieth birthday of Stanley Osher.

[52] M. Sussman, P. Smereka, and S. Osher. A level set approach for computing solutions to incompressible two-phase flows. *J. Comput. Phys.*, 114(1):146–159, 1994.

[53] A. Toga. *Brain Warping*. Academic Press, New York, 1998.

[54] B. P. Vollmayr-Lee and A. D. Rutenberg. Fast and accurate coarsening simulation with an unconditionally stable time step. *Phys Rev E*, 68, 2003.

[55] D.E. Weidner, L.W. Schwartz, and M.H. Eres. Simulation of coating layer evolution and drop formation on horizontal cylinders. *Journal of Colloid and interface science*, 187:243–258, 1997.

[56] T. P. Witelski and M. Bowen. ADI schemes for higher-order nonlinear diffusion equations. *Appl. Numer. Math.*, 45(2-3):331–351, 2003.

[57] J. Xu and H. K. Zhao. An Eulerian formulation for solving partial differential equations along a moving interface. *J. Sci. Comput.*, 19(1-3):573–594, 2003. Special issue in honor of the sixtieth birthday of Stanley Osher.

[58] N. N. Yanenko. *The method of fractional steps. The solution of problems of mathematical physics in several variables*. Springer-Verlag, New York, 1971. Translated from the Russian by T. Cheron. English translation edited by M. Holt.

[59] G. Yngve and G. Turk. Creating smooth implicit surfaces from polygonal meshes. *Technical Report GIT-GVU-99-42, Graphics, Visualization, and Usability Center. Georgia Institute of Technology*, 1999.

[60] L. Zhornitskaya and A. L. Bertozzi. Positivity preserving numerical schemes for lubrication-type equations. *S.I.A.M. J. Num Anal.*, 37(2):523–555, 2000.