THE MODIFIED CAHN-HILLIARD EQUATION ON GENERAL SURFACES

by

Bobak Shahriari

Bachelor of Science, McGill University, 2008

THESIS SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

IN THE DEPARTMENT

OF

MATHEMATICS

SIMON FRASER UNIVERSITY

Fall 2010

APPROVAL

Name: Bobak Shahriari

Degree: Master of Science

Title of Thesis: The Modified Cahn-Hilliard Equation on General Surfaces

Examining Committee: Prof. Razvan C. Fetecau (Chair)

_____

Prof. Steven J. Ruuth

_____

Prof. Ralf W. Wittenberg

_____

Prof. J. F. Williams

Date Approved: 3 December 2010

Declaration of
Partial Copyright Licence

Abstract

Diblock copolymer melts are of great interest in industry today. Their ability to naturally self-assemble at a microscopic scale is a great asset in manufacturing complex materials such as plastics, textiles, and integrated circuits. The modified Cahn-Hilliard (mCH) equation studied in this thesis is a partial differential equation (PDE) based mathematical model for diblock copolymer self-assembly. This is a stiff, non-linear, fourth-order parabolic PDE that presents some challenges when numerically solving it on general surfaces. This thesis presents several methods employed in overcoming these difficulties and produces results that support the accuracy of these methods. Our software uses the Closest Point Method (CPM), a central feature of which is geometric flexibility. This allows us to compute on simple analytically defined shapes, such as the sphere, as well as on complex shapes defined by triangulation with no modification to the code.

To Olga and Nilima...

...without whom there would be no thesis

Acknowledgments

I thank my supervisor, Professor Steve Ruuth, as well as Prof. Rustum Choksi for support and help in forming this project. Many thanks to Profs. Nilima Nigam and Paul Tupper for much needed moral support through difficult times. Thanks to Marc D. Ryser for his help with matters of white noise. Thanks to the PIMS office for making the last two years as enjoyable as possible, with special thanks to Sandie, Todd, and Brian for running the office so smoothly, and Kevin for countless interesting discussions.

Finally, this research was financially supported by a NSERC CGS and a SFU Department of Mathematics Graduate Fellowship, as well as by Profs. Ruuth and Choksi.

Contents

List of Figures

Chapter 1

A Modified Cahn-Hilliard Equation

The modified Cahn-Hilliard equation is a mathematical model for the micro-phase separation of diblock copolymers. These have many applications in industry due to their self-assembling properties. There are still many unanswered questions as to how curvature affects this model. The goal of this thesis is to solve the modified Cahn-Hilliard (mCH) equation on general surfaces. The methods implemented in this endeavour are general and are designed to be readily extended to any general smooth surface.

This introductory chapter presents diblock copolymers, provides a brief derivation of the mCH equation, and exposes its pattern forming property by performing a linear stability analysis on the finite plane with periodic boundary conditions, $\mathbb{T}^2$, and on the 2-sphere, S.

1.1 Diblock Copolymers

Monomers are molecules that are able to bond together to form long chains, or macromolecules, called polymers. These are ubiquitous in industry today. For example polystyrene is used to make Styrofoam, while polyester is widely used in the textile industry.

Polymers are chosen according to their inherent properties whether it be chemical, electrical, mechanical, or optical. In certain applications, one might wish to combine two different properties in one material. For instance, monomer A might have a desired mechanical property while monomer B has a desired optical property. One would like to ensure a structured and even distribution of the two properties throughout the material. To this end, chemists covalently bond two chains of polymers A and B, thus forming a macromolecule

called a diblock copolymer. A diblock copolymer melt is a high-temperature, disordered fluid composed of these macromolecules. When such a melt cools it can, under certain circumstances, undergo a phase separation and naturally form a complex material with intricate microstructure; in other words, domains of A and B nucleate and settle into a spatial configuration at the microscopic scale [BF99].

1.2 The Cahn-Hilliard Energy Functional

Before deriving the modified Cahn-Hilliard equation, let us introduce some of the mathematics by considering a slightly simpler problem. Suppose we are given a melted mixture of two *distinct* fluids, A and B on a finite domain $\mathscr{M}$ with periodic boundary conditions. The manifold $\mathscr{M}$ is also equiped with a metric g to measure distances. In this simpler case, chains of A and B are not bonded together and are free to segregate. The Cahn-Hilliard equation models this system as it is cooled.

At any point on $\mathscr{M}$, there are relative densities $\rho_A, \rho_B : \mathscr{M} \to [0,1]$. These represent the ratio of volume of A or B per volume of mixture, respectively. Suppose there are no other components in our system and no vacuum is allowed to form, so that the following constraint holds:

$$\rho_A + \rho_B = 1. \tag{1.1}$$

One can therefore reduce the number of variables by one and solve for a single function $\tilde{u} : \mathscr{M} \to [-1,1]$ defined as

$$\tilde{u} := \rho_A - \rho_B. \tag{1.2}$$

Notice $\tilde{u}(\mathbf{x}) = 1$ means that there is no B at $\mathbf{x} \in \mathscr{M}$, and $\tilde{u}(\mathbf{x}) = -1$ means there is no A at $\mathbf{x}$. The remainder of this section is an overview of manipulations carried out in a rigorous way in [CPW09] that lead to the modified Cahn-Hilliard equation.

The Cahn-Hilliard and modified Cahn-Hilliard equations are obtained by taking the H^{-1} gradient flow of an energy that reasonably describes the preferences of the system. The two fluids, A and B, will want to (i) be with like fluids, i.e. $\tilde{u} = \pm 1$ should be encouraged and (ii) minimize the interface of interaction with the other fluid, i.e. boundaries (where $|\nabla \tilde{u}| \gg 1$) should have minimal length. We work in the diffuse interface limit where the transition layer has thickness γ. One possible choice of energy functional is the following,

commonly known as the Cahn-Hilliard energy functional

$$E(\tilde{u};\gamma) \;:=\; \underbrace{\frac{1}{4}\int_{\mathscr{M}}(1-\tilde{u}^2)^2 d\Omega(g)}_{(i)} \;+\; \underbrace{\frac{1}{2\gamma^2}\int_{\mathscr{M}}|\nabla\tilde{u}|^2 d\Omega(g)}_{(ii)} \tag{1.3}$$

where $d\Omega(g)$ is an infinitesimal volume element that depends on the metric g. The parameter $\gamma > 0$ enables the control of the relative importance of the two terms in our energy.

Physically speaking, the energy should be bounded. Mathematically, (1.3) is bounded when $\nabla\tilde{u}$ is bounded in L^2. Therefore the energy minimizer $\tilde{u}$ is in the space $H^1(\mathscr{M})$. See the appendix for a more precise definition. The gradient of a function $\tilde{u}$ in this space must have a finite L^2 norm but is not necessarily continuous; therefore kinks in the minimizer of (1.3) are allowed.

1.2.1 Adding the Non-Local Term

Now consider the diblock copolymer melt where the A and B fluids cannot macroscopically segregate because each chain of A is attached to a chain of B. The energy functional needs the added condition that (iii) $\tilde{u}$ locally maintains its average value m. The H^{-1} norm of $\tilde{u}-m$, (iii) in (1.4), is the ideal tool to impose such a constraint. This added feature leads to the modified Cahn-Hilliard free energy functional

$$E(\tilde{u};\gamma,m) \;:=\; \underbrace{\frac{1}{4}\int_{\mathscr{M}}(1-\tilde{u}^2)^2 d\Omega(g)}_{(i)} \;+\; \underbrace{\frac{1}{2\gamma^2}\int_{\mathscr{M}}|\nabla\tilde{u}|^2 d\Omega(g)}_{(ii)} \;+\; \underbrace{\frac{1}{2}\int_{\mathscr{M}}|\nabla v|^2 d\Omega(g)}_{(iii)} \tag{1.4}$$

where v solves

$$-\Delta v = \tilde{u} - m \tag{1.5}$$

on $\mathscr{M}$ with periodic boundary conditions. With this energy functional one seeks a global minimizer of E, with fixed γ and m, $\tilde{u} \in \{w \in H^1(\mathscr{M}) | \int w = m\}$.

Link To Experiments

There are different ways to scale this PDE. The advantage of this scaling of the modified Cahn-Hilliard functional is that the parameters γ and m it features can be related in a very simple way to physically meaningful parameters [CPW09]. This helps one gain some intuition in the expected qualitative behaviour of global minimizers. More importantly, it helps relate results found numerically with the model to results found experimentally in the laboratory.

Indeed γ is proportional to a product of parameters known in the polymer science literature as χN. The dimensionless Flory-Huggins parameter χ quantifies the incompatibility of A and B per monomer, and N measures the length of a polymer chain in number of monomers (χN is therefore a unitless dimensionless quantity). In other words, the larger the χN the sharper the edges because if A and B feel a strong repulsion, there will be less mixing at the domain boundaries. This also agrees with the model since in the functional a large γ allows a greater larger variation at the boundaries, i.e. sharper edges. Physically, the Flory-Huggins parameter is also inversely proportional to temperature. In the model, if γ is decreased (temperature increased) then the fourth order term becomes more important and we expect smoother solutions.

1.3 A Modified Cahn-Hilliard Equation

In order to locate the minimizer function $\tilde{u}$, one derives an evolution equation from the modified Cahn-Hilliard energy functional by defining an artificial time variable t and redefining $\tilde{u} : \mathbb{R}^+ \times \mathscr{M} \to [-1,1]$. This way $\tilde{u}$ will travel across the energy landscape in directions that will decrease the Cahn-Hilliard free energy until it reaches a steady state which will be an, at least local, minimizer of the energy. By formally computing the H^{-1} directional gradient of (1.4) and making $\tilde{u}$ evolve along this direction one arrives [CPW09] at the following gradient flow equation

$$\tilde{u}_t = \Delta(\tilde{u}^3) - \Delta\tilde{u} - \frac{1}{\gamma^2}\Delta^2\tilde{u} - (\tilde{u} - m) \tag{1.6}$$

where $\tilde{u}(t,\cdot) \in \left\{ w \in H^1(\mathscr{M}) \middle| \int w = m \right\}$ for all $t > 0$. This fourth-order, non-linear parabolic partial differential equation can be rescaled by rescaling $\tilde{u}$

$$u = \tilde{u} - m \tag{1.7}$$

which yields

$$u_t = \Delta(u^3 + 3mu^2) - (1 - 3m^2)\Delta u - \frac{1}{\gamma^2}\Delta^2 u - u \tag{1.8}$$

where $u(t,\cdot) \in \left\{ w \in H^1(\mathscr{M}) \middle| \int w = 0 \right\}$. The new u now represents the deviation from the macroscopic average m. The goal of this thesis is to solve Equation 1.8 on general surfaces numerically. We shall first briefly analyse the equation.

1.4 Linear Stability Analysis

Equation (1.8) is the form of the modified Cahn-Hilliard equation that will be studied in this thesis.

This pattern-forming equation features bifurcation parameters, namely γ and m. Structure forms when the parameters are such that the homogeneous solution becomes linearly unstable and certain modes grow in amplitude. The non-linear terms are responsible for bounding the growth, while the most unstable modes dictate the length scale of the observed patterns. Because of the crucial role played by linear stability, in this section we provide a linear stability analysis of the equation on two different geometries: $\mathbb{T}^d$ and the 2-sphere S.

1.4.1 In $\mathbb{T}^d$

Let us begin the analysis in $\mathbb{T}^d$, the d-dimensional hypercube of side 2π with periodic boundary conditions, by considering spatially homogeneous solutions. In this case all the spatial derivatives vanish leaving

$$u_t = -u, \quad \text{in } \mathbb{R}^+ \times \mathbb{T}^d \tag{1.9}$$

which yields

$$u(t,\mathbf{x}) = u_0 e^{-t}, \quad (t,\mathbf{x}) \in \mathbb{R}^+ \times \mathbb{T}^d \tag{1.10}$$

which decays to $u = 0$. Therefore the constant solution $u(t, \mathbf{x}) \equiv 0$ is linearly stable to all spatially homogeneous perturbations. Recall from (1.7) that $u = 0$ means that there is no deviation in volume fraction, $\tilde{u}$, from the macroscopic average m. This solution corresponds to disorder: it does not exhibit any pattern or structure. At any point, any neighbourhood contains as many A as B monomers per volume if $m = 0$, and the corresponding proportion if $m \neq 0$.

Next consider the linear part of the modified Cahn-Hilliard equation

$$u_t = -(1 - 3m^2)\Delta u - \frac{1}{\gamma^2}\Delta^2 u - u \tag{1.11}$$

and consider small perturbations to the homogeneous solution. Since we are in $\mathbb{T}^d$ let us decompose the perturbation into Fourier modes since they are the eigenfunctions of the Laplace operator on $\mathbb{T}^d$

$$u(t, \mathbf{x}) = \sum_{\mathbf{k}} \hat{u}_{\mathbf{k}}(t) e^{\lambda_k t} e^{i\mathbf{k}\cdot\mathbf{x}} \tag{1.12}$$

where $|\hat{u}_{\mathbf{k}}(t)| \ll 1$, for all $\mathbf{k} \in \mathbb{N}^d$.

Plugging the perturbation ansatz (1.12) into (1.11), and using orthogonality of Fourier modes to decouple the problem, leads to the following growth rates λ_k for the k^{th} perturbation modes

$$\lambda_k \hat{u}_{\mathbf{k}} = \left[(1 - 3m^2)k^2 - \frac{1}{\gamma^2}k^4 - 1\right]\hat{u}_{\mathbf{k}} \tag{1.13}$$

where $k^2 = |\mathbf{k}|^2$. Henceforth, the label "k^{th} modes" refers to all modes $\mathbf{k} \in \mathbb{N}^d$ for which $|\mathbf{k}| = k$. The growth rate is plotted in Figure 1.1. One easily computes the quantities

$$k_{max}^2 = \frac{\gamma^2}{2}\left(1 - 3m^2\right) \tag{1.14}$$

and

$$\lambda_{max} = \frac{\gamma^2}{4}\left(1 - 3m^2\right)^2 - 1. \tag{1.15}$$

Given the form (1.12), the amplitude of the k^{th} perturbation modes grows when $\lambda_k > 0$ and decays to 0 when $\lambda_k < 0$. In other words, pattern formation occurs when $\lambda_k > 0$ or $\gamma(1 - 3m^2) > 2$; this is a curve in (γ, m) space known as the Order-Disorder Transition (ODT) curve. Perturbation modes k_{max} grow the fastest, quickly dominating other decaying or less rapidly growing modes.

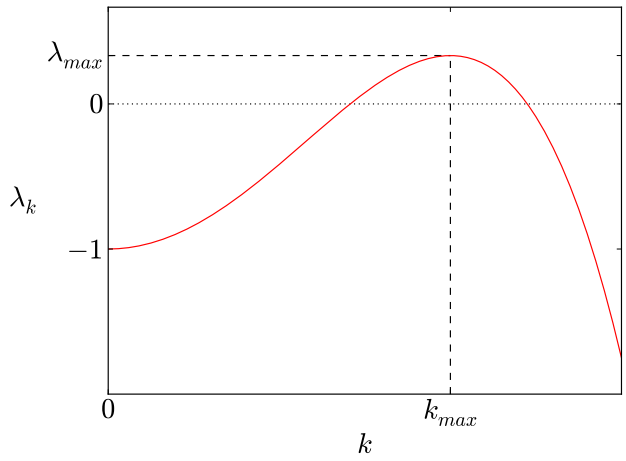


Figure 1.1: Growth rate of the k^{th} perturbation modes. The allowed values of k are integers, and the corresponding λ_k values are read off the curve.

Since $\mathbb{T}^d$ is bounded, the Fourier spectrum is discrete and k_{max} may not be an integer. In this case the allowed k_{max} modes are the $\mathbf{k} \in \mathbb{N}^d$ with highest associated λ_k. These perturbation modes dictate the length scale of the solution which is inversely proportional to k_{max}.

Depending on the geometry, different patterns may form. For instance, in $\mathbb{T}^1$ the solution resembles a sine curve while in $\mathbb{T}^2$ the different combinations of k_x and k_y allow for hexagonally packed spots as well as lamellar phases (characterized by stripes of alternating A and B domains) [Mar08]. In $\mathbb{T}^3$ the patterns can become quite intricate with interweaving domains of A and B and the (γ, m) phase diagram is a mosaic of seemingly stable structures [MB96]. Presumably, the chosen pattern is the one that minimizes the non-local Cahn-Hilliard energy functional given a set of bifurcation parameters γ and m.

1.4.2 On the 2-Sphere

The linear stability analysis of this equation on the 2-sphere S is remarkably similar to the analysis in $\mathbb{T}^d$. This is due to the similarities between the Laplacian in $\mathbb{T}^d$ and the

Laplace-Beltrami operator on S. Indeed, on a sphere of radius R equation (1.8) becomes

$$u_t = s^2\Delta_S\left(u^3+3mu^2\right) - s^2\left(1-3m^2\right)\Delta_S u - \frac{s^4}{\gamma^2}\Delta_S^2 u - u \tag{1.16}$$

where Δ_S is the Laplace-Beltrami operator and $s = \frac{1}{R}$. Spatially homogeneous solutions clearly still decay exponentially to $u \equiv 0$. The equation can be split into a linear and non-linear part

$$u_t = \mathscr{N}\{u\} + \mathscr{L}u \tag{1.17}$$

where

$$\mathscr{N}\{u\} = s^2\Delta_S\left(u^3+3mu^2\right) \tag{1.18}$$

and

$$\mathscr{L} = -s^2\left(1-3m^2\right)\Delta_S - \frac{s^4}{\gamma^2}\Delta_S^2 - 1. \tag{1.19}$$

For now, we are only concerned with the linear operator $\mathscr{L}$. Using the same argument as in Section 1.4.1 on the 2-sphere, perturbations should be decomposed into the eigen-basis of the Laplace-Beltrami operator, namely the spherical harmonics Y_l^n labeled by a non-negative positive integer l and a second integer index $-l \le n \le l$. The eigenvalue corresponding to Y_l^n is $-l(l+1)$, i.e.

$$\Delta_S Y_l^n = -l(l+1)Y_l^n. \tag{1.20}$$

Expanding perturbations in spherical harmonics yields

$$u(t,\cdot) = \sum_l \sum_n \hat{u}_n^l Y_l^n(\cdot)e^{\lambda t}. \tag{1.21}$$

Plugging this into (1.27) in turn yields the following growth rate for the $(l,n)^{th}$ mode

$$\lambda_{ln} = s^2\left(1-3m^2\right)l(l+1) - \frac{s^4}{\gamma^2}l^2(l+1)^2 - 1. \tag{1.22}$$

The 2-sphere is a finite domain and the spectrum is discrete. The expression for λ_{ln} looks a lot like λ_k from Section 1.4.1 and one readily finds the location of the peak to be

$$l_{peak}(l_{peak}+1) = \frac{\gamma^2}{2s^2}\left(1-3m^2\right), \tag{1.23}$$

$$l_{peak} = -\frac{1}{2} + \frac{1}{2}\sqrt{1+\frac{2\gamma^2}{s^2}\left(1-3m^2\right)}. \tag{1.24}$$

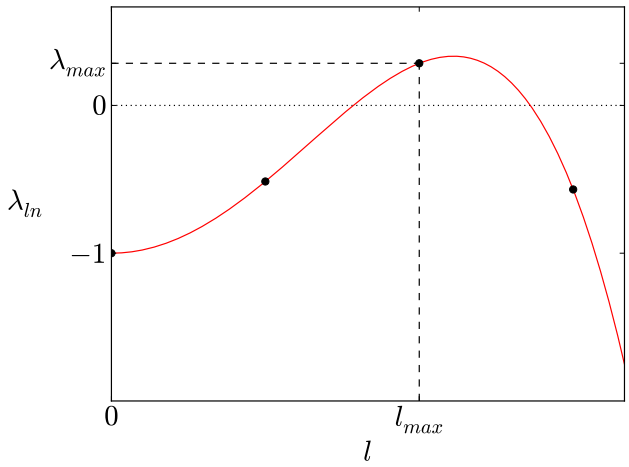


Figure 1.2: Growth rate of the l^{th} perturbation mode. The (red) curve is only a guide. Since l takes on positive integer values only, the growth rate is necessarily discrete (black dots).

However, since l must be an integer, the true fastest growing mode l_{max} will be one of the integer values sandwiching l_{peak}, the one that yields the highest λ, namely $\lambda_{max} := \lambda_{l_{max}n}$, refer to Figure 1.2.

Note that plugging (1.23) into (1.22) yields the following minimum condition for linear instabilitiy:

$$\frac{\gamma^2}{4}(1-3m^2)^2 - 1 > 0. \tag{1.25}$$

This is the same expression as in $\mathbb{T}^d$, the constant curvature does not affect the ODT curve.

We proceed with an asymptotic analysis of the behaviour of solutions to the equation as we study problems that are very close to the ODT curve. Consider a fixed point in parameter space $(s, \bar{\gamma}, \bar{m})$ on the ODT curve, and consider perturbing the problem in the m-direction, i.e.

$$m = (1-\varepsilon)\bar{m} \tag{1.26}$$

where $\varepsilon \ll 1$. So the perturbed problem becomes

$$u_t = s^2\Delta_S\left(u^3 + 3(1-\varepsilon)\bar{m}u^2\right) - s^2\left(1-3(1-2\varepsilon+\varepsilon^2)\bar{m}^2\right)\Delta_S u \\ - \frac{s^4}{4}\left[1-3\bar{m}^2(1-2\varepsilon+\varepsilon^2)\right]^2\Delta_S^2 u - u \tag{1.27}$$

Expanding the solution asymptotically around the homogeneous solution

$$u = \varepsilon u_0 + \varepsilon^2 u_1 + \varepsilon^3 u_2 + \dots \tag{1.28}$$

and plugging (1.28) into (1.27) leads to

$$\begin{aligned}(\varepsilon u_0 + \varepsilon^2 u_1 + \varepsilon^3 u_2 + \dots)_t = s^2\Delta_S\Big(\varepsilon^3 u_0^3 + 3\bar{m}\varepsilon^2 u_0^2 - 3\bar{m}\varepsilon^3 u_0^2 + \dots\Big) + \\ - s^2\left[\varepsilon(1-3\bar{m}^2)\Delta_S u_0 + 6\varepsilon^2\bar{m}^2\Delta_S u_0 + \varepsilon^2(1-3\bar{m}^2)\Delta_S u_1 + \dots\right] + \\ - \frac{s^4}{4}\left[\varepsilon(1-3\bar{m}^2)^2\Delta_S^2 u_0 + 12\varepsilon^2\bar{m}^2(1-3\bar{m}^2)\Delta_S^2 u_0 + \varepsilon^2(1-3\bar{m}^2)^2\Delta_S^2 u_1 + \dots\right] + \\ - \varepsilon u_0 + \varepsilon^2 u_1 + \dots\end{aligned} \tag{1.29}$$

where we have ignored most ε^3 terms. We now equate the orders individually:

First order:

$$(u_0)_t = -s^2(1-3\bar{m}^2)\Delta_S u_0 - \frac{s^4}{4}(1-3\bar{m}^2)^2\Delta_S^2 u_0 - u_0 \tag{1.30}$$

Second order:

$$\begin{aligned}(u_1)_t = -(1-3\bar{m}^2)\Delta_S u_1 - \frac{s^4}{4}(1-3\bar{m}^2)^2\Delta_S^2 u_1 - u_1 + \\ + s^2 3\bar{m}\Delta_S\big(u_0^2\big) - 6\varepsilon^2 s^2\bar{m}^2\Delta_S u_0 - 3s^4\bar{m}^2(1-3\bar{m}^2)\Delta_S^2 u_0\end{aligned} \tag{1.31}$$

or more succinctly

$$(u_1)_t = \mathscr{L}u_1 + \mathscr{R}\{u_0\} \tag{1.32}$$

Notice that the second-order equation looks like the first-order equation with a forcing term that arises from the lower order solution u_0. The first term of the second line of (1.31) is where part of the non-linear term is felt. We know how to solve (1.30) with spherical harmonics; u_1 can similarly be solved for with spherical harmonics Y_l^n as long as there are no resonant forcing caused by the non-linear operator $\mathscr{R}$. In other words, the solvability condition from Fredholm's alternative will dictate the allowed value(s) of n in the u_0 solution.

Unfortunately, this is not done in any more detail, and one may need to take the analysis further into the third order equation to derive restrictions on values of n.

1.5 Thesis Outline

Chapter 2 describes all the numerical methods implemented to solve the time-dependent PDE (1.27) accurately on general smooth surfaces. Chapter 3 presents some test cases and numerical results for the surface of the 2-sphere, providing convincing evidence that the methods used can provide accurate solutions to more complicated equations. The last section of Chapter 3 is also dedicated to numerical experiments on the Stanford bunny. Finally, Chapter 4 closes with exciting possibilities for future work.

Chapter 2

Numerical Methods

This chapter presents in detail all the numerical machinery needed to numerically study the mCH equation. We begin with an introduction to the Closest Point Method (CPM), followed by the various time-stepping methods tried, and conclude with a description of the algorithm used to compute surface integrals as well as the algorithm developed to find and label domains on surfaces.

2.1 The Closest Point Method

The CPM is one of a larger class of methods called embedding methods. It solves an equation on a complicated manifold by looking at a corresponding problem in a larger, usually higher dimensional, space: the embedding space. The correspondence is non-trivial, yet quite straightforward for most differential operators. Unlike other embedding methods that might use the level set of a function, the CPM does not require an "inside" and an "outside", so the computational manifold can be open, closed, oriented, or non-oriented (e.g., the Möbius strip). The method is not restricted to surfaces embedded in $\mathbb{R}^3$, it allows solving on filaments in $\mathbb{R}^3$; or on the Klein bottle in $\mathbb{R}^4$.

When solving the equation in the embedding space, the CPM can be used with well-known methods such as finite difference, finite volume, finite element, or even spectral methods. Once the equation is solved, the desired solution on the manifold is obtained by interpolating back onto the manifold.

The next section explains what surface representation is used by the CPM and how to

compute it efficiently for general surfaces defined by triangulation. The subsequent section lists the steps of the algorithm, followed by a description of a pair of subroutines implemented to make the method more efficient, namely: banding and barycentric Lagrange interpolation.

2.1.1 The Closest Point Representation

In order to compute a numerical solution on a surface accurately, the surface must be unambiguously identified. Whether a numerical solver has a parameterization, a triangulation, or the level set of a given function, it must know where the surface lies. This section presents the closest point representation that is used by the CPM.

Given a manifold $\mathscr{M}$ embedded in a space $\mathbb{R}^d$ consider the operator

$$\text{cp} : \mathbb{R}^d \to \mathscr{M} \tag{2.1}$$

such that for each point $\mathbf{x} \in \mathbb{R}^d$, cp($\mathbf{x}$) is the point on $\mathscr{M}$ that is closest to $\mathbf{x}$, i.e.

$$\text{cp}(\mathbf{x}) = \underset{\mathbf{y}\in\mathscr{M}}{\text{argmin}} ||\mathbf{x}-\mathbf{y}||_{L^2(\mathbb{R}^d)}, \quad \forall \mathbf{x} \in \mathbb{R}^d \tag{2.2}$$

where since $\mathscr{M} \subset \mathbb{R}^d$, $(\mathbf{x}-\mathbf{y})$ is well defined.

Note that this representation is not necessarily unique. Indeed, for certain pathological manifolds, such as one with corners or kinks, it can be inherently non-unique, however, for smooth surfaces a fine enough discretization will lead to a unique representation in a neighbourhood of the surface.

Now consider the function

$$u_{\mathscr{M}} : \mathscr{M} \to \mathbb{R}. \tag{2.3}$$

This function can be extended to the embedding space by defining the function $u : \mathbb{R}^d \to \mathbb{R}$ and the extension operator $E : H^1(\mathscr{M}) \to H^1(\mathbb{R}^d)$ such that

$$u(\mathbf{x}) := Eu_{\mathscr{M}}(\mathbf{x}) = u_{\mathscr{M}}(\text{cp}(\mathbf{x})). \tag{2.4}$$

In other words, at every point in the embedding space, the operator E gives u the value of $u_{\mathscr{M}}$ at the corresponding closest point. Therefore, by construction, in a neighbourhood of $\mathscr{M}$, $Eu_{\mathscr{M}}$ is constant along directions normal to $\mathscr{M}$. It follows that for all $\mathbf{y} \in \mathscr{M}$

$$\nabla Eu_{\mathscr{M}}(\mathbf{y}) = \nabla_{\mathscr{M}} u_{\mathscr{M}}(\text{cp}(\mathbf{y})), \tag{2.5}$$

where ∇ is the standard gradient in $\mathbb{R}^d$ and $\nabla_{\mathscr{M}}$ is the intrinsic gradient on $\mathscr{M}$, which is everywhere tangent to the manifold. Similar equalities hold for the $\nabla\cdot$ and Δ operators [RM08]. Simply applying an operator E before a linear operator does not always provide the corresponding embedding space operator. As will be seen with the biharmonic operator, a simple correction provides the correct extended operator.

With these equalities, a large class of PDEs can readily be translated from an equation for $u_{\mathscr{M}}$ on $\mathscr{M}$ to an equation for u in $\mathbb{R}^d$ by simply switching the manifold's intrinsic differential operators to their counterparts in the embedding space. This way one can use a regular grid in $\mathbb{R}^d$, where there exist known, studied, and accurate discrete operators to numerically solve for an approximation to u. Once the approximation is found, it is a simple matter of interpolating the grid node values $u(\mathbf{x})$ back onto $\mathscr{M}$ to obtain an approximation to $u_{\mathscr{M}}$. It is important to note that for time-dependent PDEs this must be done at each time step, or at each stage of a Runge-Kutta method, in order to stay consistent with the original equation.

The following section goes over the algorithm step-by-step and illustrates the simplicity of the method on a simple example.

2.1.2 The Algorithm

Consider the heat equation on the unit circle, i.e.

$$\begin{cases} \frac{\partial}{\partial t} u_{\mathscr{M}} = \Delta_{\mathscr{M}} u_{\mathscr{M}}, & \text{on } \mathscr{M} \\ u_{\mathscr{M}}(0,\cdot) = u_0, \end{cases} \tag{2.6}$$

where $\mathscr{M}$ is the unit circle embedded in $\mathbb{R}^2$. Instead of solving this equation directly, consider in turn the corresponding equation in the embedding space $\mathbb{R}^2$, i.e.

$$\begin{cases} u_t = \Delta u, & \text{in } \mathbb{R}^2 \\ u(0,\mathbf{x}) = u_0(\mathrm{cp}(\mathbf{x})). \end{cases} \tag{2.7}$$

This is one of the simplest PDEs and solving it numerically in $\mathbb{R}^2$ can be much less challenging than (2.6) depending on $\mathscr{M}$. There is a plethora of methods and schemes to choose from depending on one's accuracy and speed requirements and all will work with the CPM.

A finite difference method with the second order centered stencil as a discretized Laplacian operator will be used throughout the thesis. Since the objective in this section is to introduce the method, no superfluous complications will be added. Therefore, the time-stepping method adopted in this illustration is the forward Euler method. More sophisticated, implicit and implicit-explicit time-stepping methods will be presented in Section 2.2. Finally, quadratic Lagrange interpolation is used to extract $u_{\mathscr{M}}$ from the solution u; a lower order interpolation, such as bilinear interpolation leads to inconsistent methods [MR09]. Schematically the method is:

Step 1: Compute the closest point representation:

Define a regular grid $\{\mathbf{x}_i\}$ in $\mathbb{R}^2$ and for each grid point $\mathbf{x}_i$ find the corresponding closest point $\mathrm{cp}(\mathbf{x}_i)$ on the surface.

Step 2: Extend the solution to $\mathbb{R}^2$:

Let component U_i correspond to $u(\mathbf{x}_i)$. Before the first time step, set $U_i = u_0(\mathrm{cp}(\mathbf{x}_i))$, in all subsequent steps set $U_i = u_{\mathscr{M}}(\mathrm{cp}(\mathbf{x}_i))$.

Step 3: Evolve the PDE in $\mathbb{R}^2$ by one time step:

Update U_i according to: $U_i = U_i + kD_2U_i$, where k is a time step respecting the Courant-Friedrichs-Lewy (CFL) condition, and D_2 is the standard second order finite differencing stencil for the Laplacian.

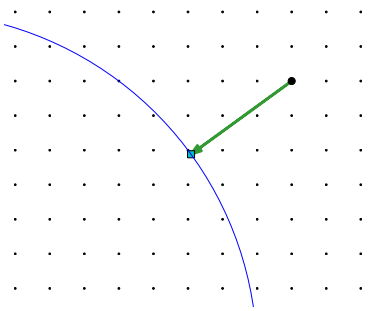
Step 4: Interpolate the solution back onto $\mathscr{M}$:

Interpolate the points near the manifold to obtain approximations to $u_{\mathscr{M}}(\mathrm{cp}(\mathbf{x}_i))$.

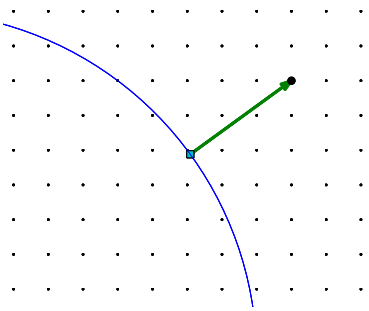
Step 5: Repeat steps 2-4 until the final time is reached.

Figure 2.1 provides a visualization of these steps. In practice, steps 4 and 2 are combined and the result of the interpolation in step 4 can be directly assigned to U_i. This is referred to as the extension step. In the previous section this was performed by the extension operator E.

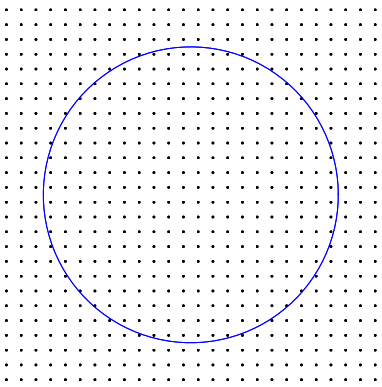
Recall from Section 2.1.1 that the intrinsic manifold $\mathscr{M}$ gradient, divergence, and Laplacian operators have simple counterparts in the embedding space namely $\Box E$ where $\Box$ is any of those operators. However as suggested by (1.27), the equation of interest in this



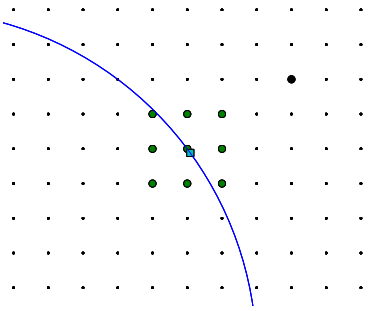
(a) Step 1: Get closest point representation of $\mathscr{M}$.



(b) Step 2: Extend initial condition to grid nodes.



(c) Step 3: Evolve PDE by a single time step on grid.



(d) Step 4: Use solution U_i at green nodes to interpolate the solution back onto $\mathscr{M}$.

Figure 2.1: Visualization of the steps involved in the CPM. The view is zoomed on the first quadrant. Black dots are grid nodes, the thicker one being a sample node. The blue curve is the unit circle and the cyan square represents the closest point to the sample node. Finally the green dots are grid nodes used in interpolating the solution at the cyan square dot. Recall that in this example quadratic interpolation is used, for higher order interpolations a wider band and consequently more green dots would be required.

thesis has a biharmonic operator $\Delta^2_{\mathscr{M}}$. In the CPM framework, this operator has to be split into two Laplacian operators $\Delta_{\mathscr{M}}\Delta_{\mathscr{M}}$ and each Laplacian must be preceded by an extension step as in (2.5). The resulting operator is

$$\Delta^2_{\mathscr{M}} \to \Delta E \Delta E. \tag{2.8}$$

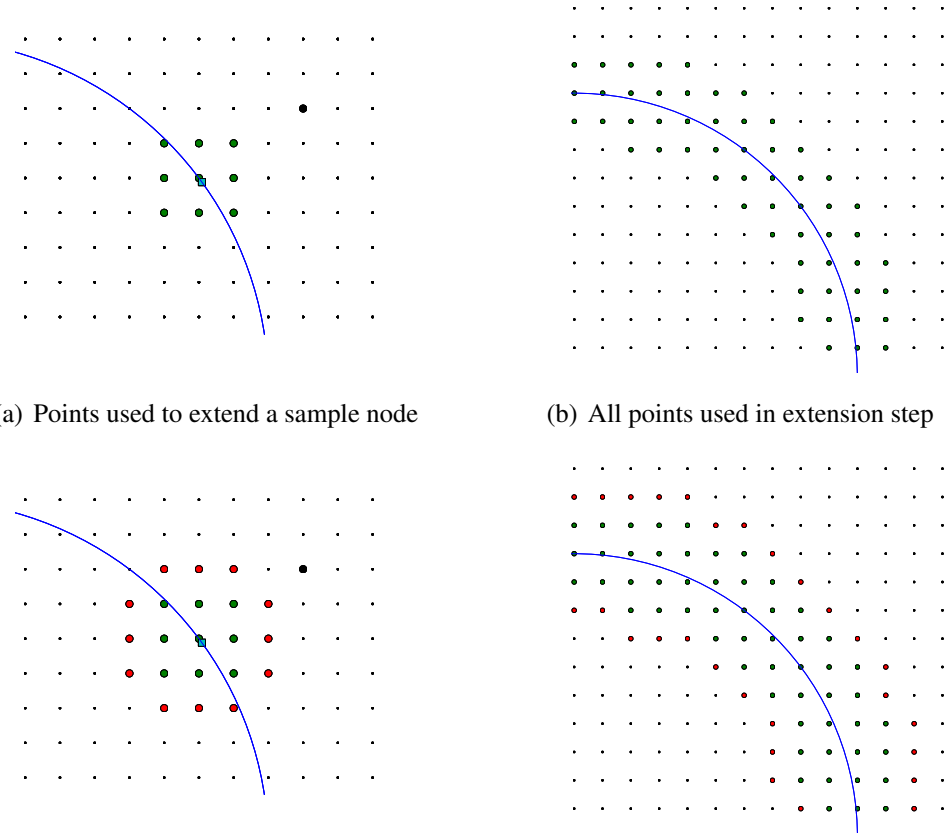
The consequences of this for the algorithm is simply that for a biharmonic operator, one computes $\Delta E u_{\mathscr{M}}$ as was shown in Step 3; the result is stored and ΔE is performed on it. Of course in an explicit scheme, the CFL condition for such an equation would be prohibitive, so in practice such a biharmonic term would be treated implicitly as is discussed in Section 3.1.2.

2.1.3 Banding

In the CPM, the two steps performed at every time step are the extension and the evolution of the solution. As can be visualized in Figure 2.2, for any given point, the only grid node values used in extending to it are located within a narrow band of the manifold $\mathscr{M}$ or, in this case, the unit circle. Let us refer to this set of points as S_{ex}. Notice that the width of the band and hence the number of points in S_{ex} depends on the order of interpolation. Therefore none of the other grid nodes play a role in the extension step.

Consider accurately evolving the values at points in the set S_{ex}, represented by green dots in Figure 2.2. Given the chosen finite difference stencil, the points needed in the evolution of S_{ex} are contained within a slightly wider band whose bandwidth depends on the order of interpolation as well as the width of the stencil. Therefore, this second set, S_{ev}, is a superset of S_{ex}. In this example the stencil only requires a point in each direction to evolve the equation. Therefore, S_{ev} includes the green and red dots in Figure 2.2.

After the evolution step, the extension step resets all the grid node values to their corresponding closest point's interpolated value. Hence, all grid nodes that are not included in the set S_{ev} (represented as black dots in Figure 2.2) take no active part in the computation. Indeed, the set S_{ev} contains all the points that come up in the entire algorithm. By ignoring all other points, one significantly reduces memory overhead by an order of magnitude while substantially increasing efficiency, allowing problems to be solved faster. In this example if N represents the number of points to a side, rather than keep track and compute



(a) Points used to extend a sample node

(b) All points used in extension step

(c) Points used to evolve a sample node

(d) All points used in evolution step

Figure 2.2: Spatial visualization of significant points used in the CPM. The view is zoomed on the first quadrant. Black dots are grid nodes, the thicker one being a sample node. The blue curve is the unit circle and the cyan square represents the closest point to the sample node. Finally the green dots are grid nodes used in interpolating while the red are used in evolving the green ones. Note a higher order interpolation would require more green points and a wider differencing stencil would require more red points; this example is for a second-order discretized Laplacian and quadratic interpolation.

with N^2 grid nodes, banding reduces the system size to cN points where c depends on the order of interpolation and the width of the stencil. Naturally, fewer points translates to less computation per time step. This effect is even more significant when using iterative solvers with implicit-explicit schemes since in this case work is not linearly proportional to system size.

2.1.4 Barycentric Lagrange Interpolation

As mentioned in the previous section, for time-dependent PDEs the interpolation (extension step) must be carried out for every point at each time step; hence it is important to have an efficient interpolation routine. Since the embedding computational grid is fixed, Barycentric Lagrange Interpolation is extremely efficient at the cost of a slight increase in memory overhead.

Consider the Lagrange interpolant in one dimension – the extension to d dimensions is straightforward: one simply applies the one-dimensional Lagrange interpolant to each dimension. Given a grid $\{x_i\}$ with corresponding solution values U_i, the extension step interpolates the values to obtain U_k at point x_k. The p^{th} Lagrange interpolation reads

$$U_k = \sum_{i \in N_p(x_k)} U_i v_i\big(\mathrm{cp}(x_k)\big) \tag{2.9}$$

with

$$v_i(x) = \frac{\prod_{j \neq i}(x - x_j)}{\prod_{j \neq i}(x_i - x_j)} \tag{2.10}$$

where $N_p(x)$ is a set of points around $\mathrm{cp}(x)$ that depends on the order of interpolation p. Because of the product nested in the sum, this process needs

$$\mathscr{O}\Big(\Big[(p+1)^{d-1} + (p+1)^{d-2} + \cdots + 1 \Big] (p+1)^2 \Big)$$

floating point operations per point in d dimensions; the factor in front of $(p+1)^2$ comes from performing one-dimensional interpolation in every direction. Since iterative implicit solvers need to extend multiple times every time step, the interpolation step takes the bulk of the processor time. Any improvement in the efficiency of this particular process can lead to an almost equivalent processor time reduction for the entire program.

The principle behind Barycentric Lagrange Interpolation (BLI) is to calculate and store the interpolation weights for each point so as to reuse them every time the interpolation routine is called. Naturally this only works because the grid and surface are fixed, which fixes the weights as well. Following a few simple manipulations, (2.9) can lead to the following equation

$$U_k = \frac{\sum_{i \in N_p(x)} W_{ik} U_i}{\sum_{i \in N_p(x)} W_{ik}} \tag{2.11}$$

where

$$W_{ik} = \frac{w_i}{\left(\mathrm{cp}(x_k) - x_i\right)}. \tag{2.12}$$

The intermediate steps involved are clearly presented in [BT04]. In the case of a regular grid such as the one used throughout the thesis, there is a simple definition of w_i

$$w_i = (-1)^i \binom{p}{i}. \tag{2.13}$$

Using BLI, the interpolation becomes a simple matrix multiplication where for each point a $p+1$ dimensional dot product is computed.

2.2 Time-Stepping

As mentioned in Section 1.3, (1.8) is a fourth-order parabolic equation; the PDE induces two very different time scales in the solution. Non-trivial solutions to this equation are characterized by a very fast formation of domains followed by an extremely slow shifting of these domains toward an optimal, energy minimizing spatial configuration. This is a quintessential stiff problem whence the requirement for implicit time-stepping methods. Several methods were implemented, methods with various stability and accuracy features, usually one coming at the expense of the other. An explicit method, such as forward Euler, can lead to prohibitively small time step restrictions. This is why all the methods tried have an implicit flavour. This section presents a few relevant methods.

2.2.1 Backward Euler

The simplest implicit scheme for solving time dependent PDEs is the backward Euler (BE) method. Let $U_n \in \mathbb{R}^M$ where M represents the number of points considered in the computation such that $\big(U_n\big)_i = u(t_n, \mathbf{x}_i)$ and let k be the time step. The method consists of approximating the time derivative as follows

$$u_t(t_{n+1}, \mathbf{x}) \sim \frac{1}{k}(U_{n+1} - U_n) \tag{2.14}$$

and evaluating the rest of the PDE at the next time step, i.e. at $u(t_{n+1}, \mathbf{x})$ or U_{n+1}. Indeed, in the problem at hand, implementing BE means considering the following

$$\frac{1}{k}(U_{n+1} - U_n) = \mathscr{N}\{U_{n+1}\} + \mathscr{L}U_{n+1} \tag{2.15}$$

where $\mathscr{N}$ and $\mathscr{L}$ were defined in (1.18) and (1.19), respectively. In other words, BE requires one to solve the non-linear system of coupled equations

$$\big(I - k\mathscr{N} - k\mathscr{L}\big)U_{n+1} = U_n. \tag{2.16}$$

This system is non-linear because of the operator $\mathscr{N}$. One can use non-linear solvers such as Newton's method but these can be costly. Instead, in this thesis $\mathscr{N}$ is linearized and the approximate iterative linear solver known as Generalized Minimal Residual (GMRES) is used. Recall that $\mathscr{N}\{u\} = u^3 + 3mu^2$ and consider the following linearization.

$$\mathscr{N}\{U_{n+1}\} \sim \bar{\mathscr{N}}U_{n+1} = \big(U_{\text{ap}}^2 + 3mU_{\text{ap}}\big)U_{n+1} \tag{2.17}$$

where U_{ap} is some explicit approximation of U. Notice that U_{n+1} is factored out and the remaining factors of U_{n+1} are replaced with an explicit approximation U_{ap}. One can implement a discretization of the desired order of accuracy for U_{ap} and simple Taylor expansions show that the following are first and second order, respectively.

$$U_{\text{ap}} = U_n \qquad = U_{n+1} + \mathscr{O}(k) \tag{2.18}$$

$$U_{\text{ap}} = 2U_n - U_{n-1} = U_{n+1} + \mathscr{O}(k^2) \tag{2.19}$$

Adopting a more accurate U_{ap} is unnecessary since BE is a first order method. Indeed, like its explicit counterpart the forward Euler method, BE exhibits first order accuracy in time.

However it features improved stability suitable for stiff problems. Our implementation of the linearized BE method consists of solving the following linear system of equation iteratively using GMRES:

$$\big(I - k\bar{\mathscr{N}} - k\mathscr{L}\big)U_{n+1} = U_n \tag{2.20}$$

where $\bar{\mathscr{N}}$ is defined in (2.17).

2.2.2 Crank-Nicholson

Like BE, Crank-Nicholson (CN) approximates the time derivative as (2.14). Where CN differs from BE is that it uses a trapezoidal rule approximation of the right hand side such that instead of (2.15) CN yields

$$\frac{1}{k}(U_{n+1} - U_n) = \frac{1}{2}\Big[\mathscr{N}\{U_{n+1}\} + \mathscr{L}U_{n+1}\Big] + \frac{1}{2}\Big[\mathscr{N}\{U_n\} + \mathscr{L}U_n\Big]. \tag{2.21}$$

With the linearization (2.17), the implementation of CN consists of solving the following linear system of equations iteratively using GMRES:

$$\big(I - \frac{k}{2}\bar{\mathscr{N}} - \frac{k}{2}\mathscr{L}\big)U_{n+1} = \big(I + \frac{k}{2}\bar{\mathscr{N}} + \frac{k}{2}\mathscr{L}\big)U_n. \tag{2.22}$$

CN is known to be second order in time. Unfortunately, it does not damp high frequency error as fast as BE. This can become a stability issue when large time-steps are taken.

2.2.3 Second Order Two-Step Methods

A family of second-order, two-step implicit-explicit (IMEX) methods were also implemented, more specifically Equation (14) from [ARW95]. It allows the use of many different schemes with a single implementation by simply taking two parameters. Article [ARW95] considers the problem

$$u_t = f(u) + g(u) \tag{2.23}$$

where f is to be solved explicitly and g is to be solved implicitly, then the family of second-order two-step IMEX methods is

$$\frac{1}{k}\left[\left(\gamma+\frac{1}{2}\right)U_{n+1}-2\gamma U_n+\left(\gamma-\frac{1}{2}\right)U_{n-1}\right]=(\gamma+1)\,f(U_n)-\gamma f(U_{n-1})+ \\ +\left[\left(\gamma+\frac{c}{2}\right)g(U_{n+1})+(1-\gamma-c)g(U_n)+\frac{c}{2}g(U_{n-1})\right] \quad (2.24)$$

where γ and c are free parameters that one can pick to choose a particular method. For example, if one selects $(\gamma,c)=(\frac{1}{2},0)$, equation (2.24) becomes the well-known Crank-Nicholson/Adams-Bashforth implicit-explicit method.

Within this class of methods, we mainly consider the second order Backward Differentiation Formula (BDF2), which is obtained from (2.24) by choosing $(\gamma,c)=(1,0)$. The semi-implicit BDF2 time-stepping scheme is

$$3U_{n+1}-4U_n+U_{n-1}=2k\Big(2f(U_n)-f(U_{n-1})+g(U_{n+1})\Big) \quad (2.25)$$

where U_n is defined in the previous section. In this thesis the following choices were made for f and g:

$$f(u)=0, \quad (2.26)$$

$$g(u)=\bar{\mathscr{N}}u+\mathscr{L}u. \quad (2.27)$$

The non-linear term is treated implicitly after proper linearization from (2.17). This leads to the solution of the following linear system using GMRES.

$$\Big(I-\frac{2}{3}k\bar{\mathscr{N}}-\frac{2}{3}k\mathscr{L}\Big)U_{n+1}=\frac{4}{3}U_n-\frac{1}{3}U_{n-1} \quad (2.28)$$

Notice that the implementation is like a backward Euler method where the first-order approximation to the time derivative is replaced with a second order one-sided differencing stencil. This is no accident, it is how (2.24) was designed.

2.2.4 Eyre's Scheme

A scheme that one encounters when researching time-stepping methods for the Cahn-Hilliard equation is the "unconditionally stable one-step scheme for gradient systems" suggested by David J. Eyre in [Eyr97]. This scheme is said to be unconditionally gradient

stable, which means that no matter the step size, the energy functional decreases after every time-step. For the modified Cahn-Hilliard equation the linearized scheme is given by [Mar08]

$$U_{n+1} - U_n = k\Big(-\frac{1}{\gamma^2} D_2^2 U_{n+1} + 2D_2 U_{n+1} - U_{n+1} \Big) + \\ + k\Big(D_2(U_n^3 + 3mU_n^2 + 3mU_n) - 3D_2 U_n \Big). \quad (2.29)$$

Though the scheme's stability is not contingent on the step size, the accuracy is. Therefore considerable care must be taken when updating the time step size, suggesting the use of an accurate error estimate (a component which has not yet been implemented). Eyre's scheme is not used in this thesis, however, it may be of value in situations where a reliable adaptive time-stepping scheme is available.

2.3 Surfaces

Thus far, details about how the closest points are computed have been omitted. This is an independent problem from the closest point method algorithm. It is performed only once at the beginning of the run and, in fact, it could be performed only once per discretization, saved to file, and loaded in for future runs. As it stands, this initialization takes such a negligible time when compared to the rest of the algorithm that the closest point computation is repeated at the beginning of every run.

Without going into too much detail, the next two sections will present how triangulated surfaces are read in, how the closest points are computed, and how surface integrals (necessary for energy computation) are approximated.

2.3.1 Triangulated Surfaces and Closest Points

All our complex shapes are defined by triangulation, some found in the online repository [AIM] while some other ones were built with the help of 3D CAD tools. These are read in as a series of vertices followed by a series of triangles (three indices corresponding to vertices in the vertex list). After this step, one may iterate either through the vertices or the triangles.

The naïve approach of finding the closest point for each grid node by iterating over all triangles will take $\mathcal{O}(h^{-d}N_\Delta)$ work, where h is the spatial step, d is the dimension of the embedding space, and N_Δ is the number of triangles. Much of that work is unnecessary since points outside the computational band do not contribute to the solution. Using a code developed by Prof. Ruuth [MR08], one iterates through all the triangles, finding the closest points for nodes within the bandwidth. For any node, the minimum distance to all nearby triangles gives the closest point. This routine takes $\mathcal{O}(cN_\Delta)$ work where c only depends on the dimension d, the order of interpolation p, and the width of the differencing stencil.

2.3.2 Surface Integrals and Energy

When solving the modified Cahn-Hilliard equation it is important to have an estimate of the non-local Cahn-Hilliard energy (1.4). This provides a comparison metric between states as well as a quantitative measure of how much the solution is changing. We adopt a straightforward computational technique to compute surface integrals.

The barycentres and areas of all the triangles are computed and stored. Every time the energy is requested, necessary quantities such as the solution, solution gradient, and non-local term are interpolated at those barycentres. An approximation of the energy functional is then computed at the barycentres with the computed quantities. Finally, each solution value is multiplied by its corresponding triangle area and added to a running sum.

Figure 2.3 shows the linear convergence of this method with respect to the number of triangles N_Δ. In Figure 2.3(a), the method was used to compute the integral of the real part of the Y_4^3 spherical harmonic, I_1, i.e.

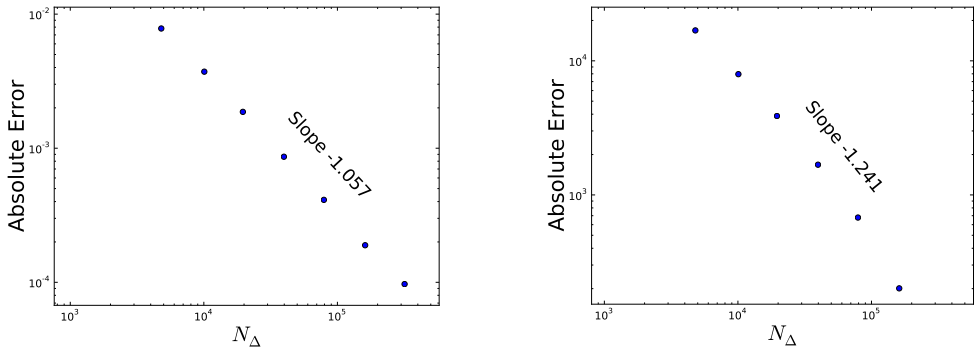
$$I_1 := \int_S Re\{Y_4^3\} d\Omega \tag{2.30}$$

the exact solution of which is known to be $\frac{1}{2}$. In Figure 2.3(b), the algorithm was also tested in computing the non-local Cahn-Hilliard energy of the same spherical harmonic, i.e.

$$I_2 := \frac{1}{4}\int_S \left(1-(Y_4^3)^2\right)^2 d\Omega + \frac{1}{2\gamma^2}\int_S |\nabla Y_4^3|^2 d\Omega + \frac{1}{2}\int_S |\nabla v|^2 d\Omega \tag{2.31}$$

where

$$-\Delta v = Y_4^3 \tag{2.32}$$



(a) Convergence of integral computation

(b) Convergence of the non-local Cahn-Hilliard energy computation

Figure 2.3: Convergence plots for the integral and non-local Cahn-Hilliard energy computations for the Y_4^3 spherical harmonic.

and the sample parameter space point $(\gamma, m) = (2.2, 0.125)$ was selected. in this case the finest discretization ($N_\Delta = 3.2 \cdot 10^5$) was used as an "exact" solution.

Figure 2.3 shows two log-log plots of the absolute error in the max-norm $||e||_\infty$, i.e.

$$||e_i||_\infty = ||I_{N_\Delta} - I_i||_\infty, \tag{2.33}$$

against N_Δ. On a log-log plot, the slope of a linear fit to the data gives an approximation to the polynomial order of convergence, e.g., if the slope is -1, then the method exhibits first-order convergence with respect to growing numbers N_Δ.

Since this method uses local, planar approximations to the surface (the triangles), we expect linear (first-order) convergence. This is very well depicted in the Figure 2.3(a), while in Figure 2.3(b), the convergence is much better than we expect (slope of 1.24). This is due to the way we chose our "exact" solution, it was computed with only twice the number of triangles as the second-finest discretization, and hence the last data point has an artificially small error. The order of convergence when discarding the final two data points is 1.08, which is closer to the expected order.

The surface is approximated by piecewise planes (triangles). The area of each triangle is computed by taking half of the moduli of the cross-product of two edges of the triangle. This corresponds to locally approximating the metric of surface $\mathscr{M}$ by the standard flat space metric.

2.4 Adaptive Time-Stepping

Solutions to the modified Cahn-Hilliard equation typically feature a rapid, initial, transient solution, followed by a slower evolution to a steady state. Initially, a very short time step is necessary to accurately solve the equation as the solution evolves quickly. Maintaining the small time step for the remainder of the computation would be grossly inefficient because the evolution soon slows down dramatically. This leads us to consider adaptive time-stepping, however, standard adaptive time-stepping methods require an accurate error estimate. Because we are interested in steady states, we chose a simpler approach: an estimate of the relative L^2-norm of the difference between the solution at two consecutive time steps is computed and we control this relative change by taking shorter or larger time step-sizes.

Let $||\Delta U_{rel}||_2$ be the L^2-norm of the relative difference of two consecutive solutions and let the target relative change be $||\Delta \bar{U}||_2$. Then our time steps are updated according to

$$k_{new} = \frac{||\Delta \bar{U}||_2}{||\Delta U_{rel}||_2} k_{old}. \tag{2.34}$$

The estimate of the L^2-norm is computed using the same method described Section 2.3.2.

With this update rule, the time step can grow too large for the iterative solver to converge. To avoid this problem we prescribe a maximum step size $\bar{k}$. Since t scales with the radius of the sphere, R, $\bar{k}$ also scales with R. When a tight upper bound for the step-size is found for a given problem and sphere size, the appropriate upper bound can be found for different sphere sizes via the same scaling, i.e. if one doubles the sphere size, one can safely double the upper bound.

2.5 Noise

Another difficulty in getting to the global minimizer of (1.4) is that the modified Cahn-Hilliard energy landscape has many local minima and it is very easy to fall into one of these stable wells of the energy. Once inside such a well, it is in principle impossible to get out since the equation attracts the solution to the corresponding local minimizer. This leads us to consider an external force. Additive space-time white noise is a particularly natural choise of forcing term and loosely corresponds to random thermal fluctuation. The

resulting equation looks like the Cahn-Hilliard-Cook equation with an added term due to the non-local term in the energy. For a careful stochastic analysis of the Cahn-Hilliard-Cook equation see [PD96]. The additive space-time white noise is uncorrelated in space and time and the amplitude of the noise is scaled by a parameter ε. In order to eventually settle into a steady state – hopefully the *global* minimizer – the parameter ε may be slowly decreased to 0.

Numerically, noise is modeled by using a normal random number generator based on the Box-Müller transform. Due to the discretized nature of the numerical problem, the noise must also be scaled by the space and time step sizes in the following way. Let $\xi \in \mathbb{R}^m$ represent the noise term where m is the number of computational points in the system, and let $Z \in \mathbb{R}^m$ be a vector of m independent identically distributed normal random variables with a unit standard deviation. Then the implemented additive noise is

$$\xi = \sqrt{\frac{k}{h^d}} Z \tag{2.35}$$

where h and k are the space and time step sizes, respectively, and d is the dimension of the embedding space. After further scaling (2.35) by ε as mentioned above, the noise is added to the explicit part of the IMEX solver. For instance, adding space-time white noise explicitly to (2.20) yields the system of equations

$$\big(I - k\bar{\mathscr{N}} - k\mathscr{L}\big)U_{n+1} = U_n + \varepsilon\xi. \tag{2.36}$$

2.6 Domain Finding

With a particular choice of parameters (γ, m), the modified Cahn-Hilliard equation on a two-dimensional surface can produce a pattern characterized by circular spots arranged on an imperfect hexagonal lattice. The particular pattern depends on the choice of parameters γ and m. In general and especially on curved surfaces, structured solutions often contain defects. Whether they are artifacts of the curvature or confinement, or naturally ocurring defects that minimize the free energy, they are of great interest because of their effect on the macroscopic properties of the material. A good way to systematically find these defects on a hexagonal lattice is to make a list of all the spots and count the number of neighbours for each.

Our approach to counting spots begins by outputting the solution at each vertex of the triangulated surface. Though the spots are easily detected by the human eye, an algorithm was needed to detect spots automatically. It is assumed that spots are domains where $u > \bar{u}$ where $\bar{u}$ is some threshold value. A list called ActiveList is kept where the i^{th} component of the list is deemed active if $u_i > \bar{u}$ and inactive otherwise. Therefore, after ActiveList is initialized, all vertices that are contained in a spot are active. The algorithm then walks through the vertices creating a list, SpotList, assigning an integer to each vertex. Vertices with the same integer in SpotList belong to the same spot.

Algorithm 1 Domain Finding

```
SpotCount = 1
for all vertex i do
  if vertex i is active then
    AssignSpotCount ( i , SpotCount )
    SpotCount++
  end if
end for
```

This is a recursive algorithm because AssignSpotCount() calls itself.

Algorithm 2 AssignSpotCount

```
SpotList( i ) = SpotCount
Deactivate vertex i
for all vertex j neighbouring vertex i do
  if vertex j is active then
    AssignSpotCount ( j , SpotCount )
  end if
end for
```

Algorithm 2 walks through all the active neighbours within a spot assigning the same integer until there are no active vertices left. At the end of Algorithm 1, SpotList associates each vertex with a single spot or none at all. A crude estimate of the barycentre of a spot can be computed by simply taking the mean position of vertices within it. Note that this

algorithm was used to number spots; it does not make any assumption on domain shape. In Section 3.2.3 an example of this algorithm's results is provided.

2.6.1 Neighbour Counting Algorithm

In a regime of phase space (γ, m) where the structure exhibits spots, one can be very interested in systematically finding defects. One way of doing this is to count the number of neighbours of each spot. We have developed an algorithm to give an estimate of the number of neighbours per spot. The Domain Finding Algorithm (DFA) is used to label the spots and locate the barycentres; the total number of spots is also output by the DFA. In the neighbour counting algorithm, the total area of the surface is divided by the number of spots. This gives an estimate for the area A of a hexagonal cell. The area is then converted to a radius corresponding to a disk of area A. Each spot is considered a neighbour of another if its barycentre is within this radius' length of another spot.

Chapter 3

Numerical Results

This third chapter is devoted to the various results obtained using the implementations discussed in the previous chapter. The first section covers results on test cases on the sphere where the exact solution is known; this helps validate the code. More ambitiously, the next section attempts to validate the code solving the modified Cahn-Hilliard equation where the exact solution is not known. Finally, in the last section, the flexibility of the Closest Point Method is revealed when the unmodified code is used to solve on the Stanford bunny, a smooth yet complex shape.

3.1 Test Cases on the 2-Sphere

Before attempting to solve the modified Cahn-Hilliard equation on general surfaces, we validate our code by predictably solving simpler problems on a familiar shape like the 2-sphere. These PDEs are not as challenging and, more importantly, their solutions are known so convergence studies are straightforward.

3.1.1 The Heat Equation

Consider the following heat equation on the unit sphere S

$$\begin{cases} u_t = \Delta_S u & \text{on } S \times (0,\infty) \\ u = Y_1^0 & \text{at } t = 0 \end{cases} \tag{3.1}$$

where in spherical coordinates $Y_1^0(\theta,\phi) = \cos(\theta)$. The solution of (3.1) is known exactly:

$$u(t) = e^{-2t} Y_1^0. \tag{3.2}$$

The subset of $\mathbb{R}^3$ used as an embedding space is the box centered at the origin with side $L = 4$. The orders of interpolation p that were run are $p = 2,3,4$, and 5, and the discretizations N sampled are $N = 41,61,81,101,121$, and 141 where $h = \frac{L}{N-1}$. This was solved with three time-stepping methods: Backward Euler (BE), Crank-Nicholson (CN), and the two-step Backward Differentiation Formula (BDF2). Since these are all implicit methods, an appropriate time step size is $k = \mathscr{O}(h)$ and we take $k = \frac{1}{8}h$. Since $k \sim N^{-1}$, when plotting the max norm of the relative error against N on log-log axes a negative slope of 1 or 2 is expected depending on whether the method is first or second-order, respectively. Note since both the space and the time step are refined with growing N, this is a convergence study of the algorithm, not of the individual time-stepping methods.

Figure 3.1 shows that the code produced the expected orders of convergence with all three methods.

3.1.2 The Biharmonic Heat Equation

The modified Cahn-Hilliard equation has a biharmonic operator hence the importance of being able to solve the biharmonic heat equation. The conditioning of this problem is poor and with the same $k = \frac{1}{8}h$ condition GMRES took too many iterations to be practical (see Appendix B). Improved speed was obtained by taking the time step size restriction $k = \frac{1}{2}h^2$.

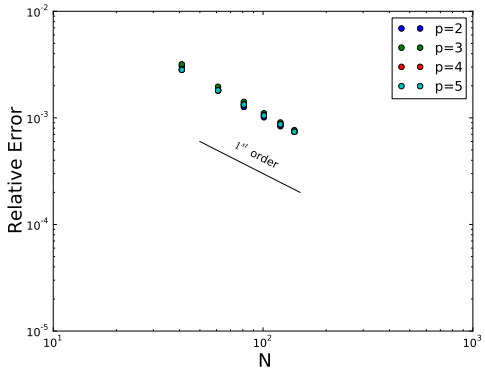
Consider the biharmonic heat equation on the unit sphere S

$$\begin{cases} u_t = -0.1\Delta_S^2 u & \text{on } S \times (0,\infty) \\ u = Y_1^0 & \text{at } t = 0. \end{cases} \tag{3.3}$$

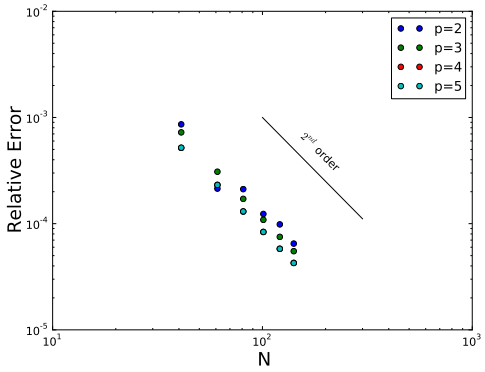
The solution of (3.3) is given exactly by

$$u(t) = e^{-4t} Y_1^0. \tag{3.4}$$

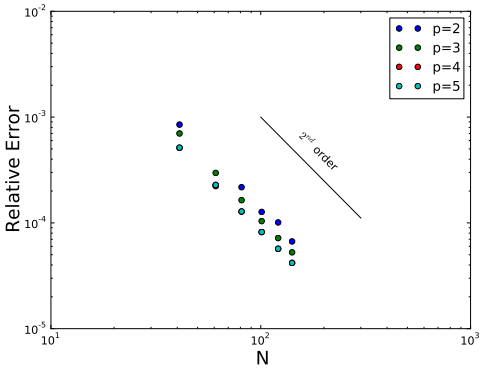
Once again, the subset of $\mathbb{R}^3$ used as an embedding space is the box centered at the origin with side $L = 4$. The orders of interpolation p that were run are $p = 2,3,4$, and the discretizations N sampled are $N = 41,61,81,121$ where $h = \frac{L}{N-1}$. The same three methods



(a) Backward Euler

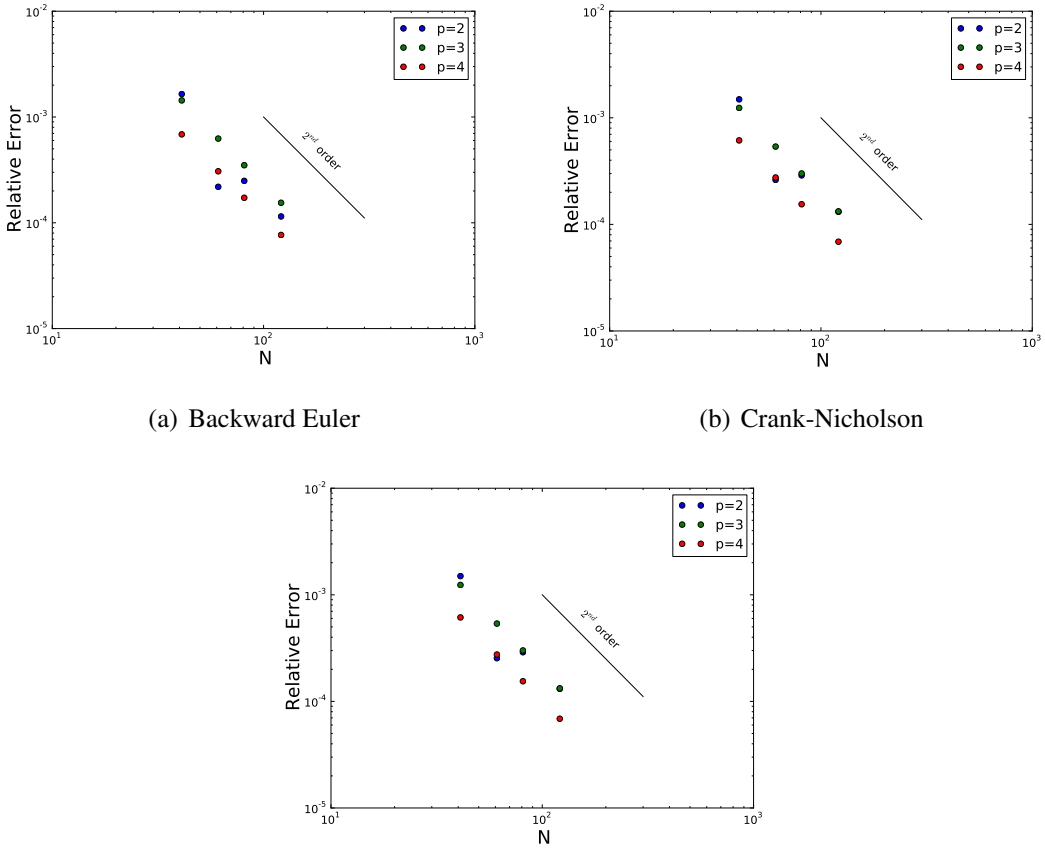


(b) Crank-Nicholson



(c) BDF2

Figure 3.1: Heat equation convergence plot: log-log plot of the max norm of the relative error against the discretization N. As expected, BE shows first-order convergence while CN and BDF2 both converge to second-order. Note that the $p = 4$ and $p = 5$ runs overlap almost overlap.



(a) Backward Euler

(b) Crank-Nicholson

(c) BDF2

Figure 3.2: Biharmonic heat equation convergence plot: log-log plot of the max norm of the relative error against the discretization N. BE is still first order in time but since $k \sim N^{-2}$ this translates to second order with respect to N. Were it not for the second order discretized spatial scheme, CN and BDF2 would be expected to both exhibit fourth order convergence with respect to N.

as in the previous section were tried, namely BE, CN, and BDF2. When neglecting spatial error and with $k \sim N^{-2}$, plotting the max norm of the relative error against N on log-log axes is expected to yield a negative slope of 2 or 4 depending on whether the method is first or second order, respectively. However, as can be seen in Figure 3.2, CN and BDF2 both only exhibit second order convergence. This is due to the fact that the discretized Laplacian is only second order accurate in space. Therefore, while the methods are second order in time, the error made in the spatial discretization is the dominant error. A reduction of this spatial error is possible, however, by considering a wider, higher order Laplacian stencil.

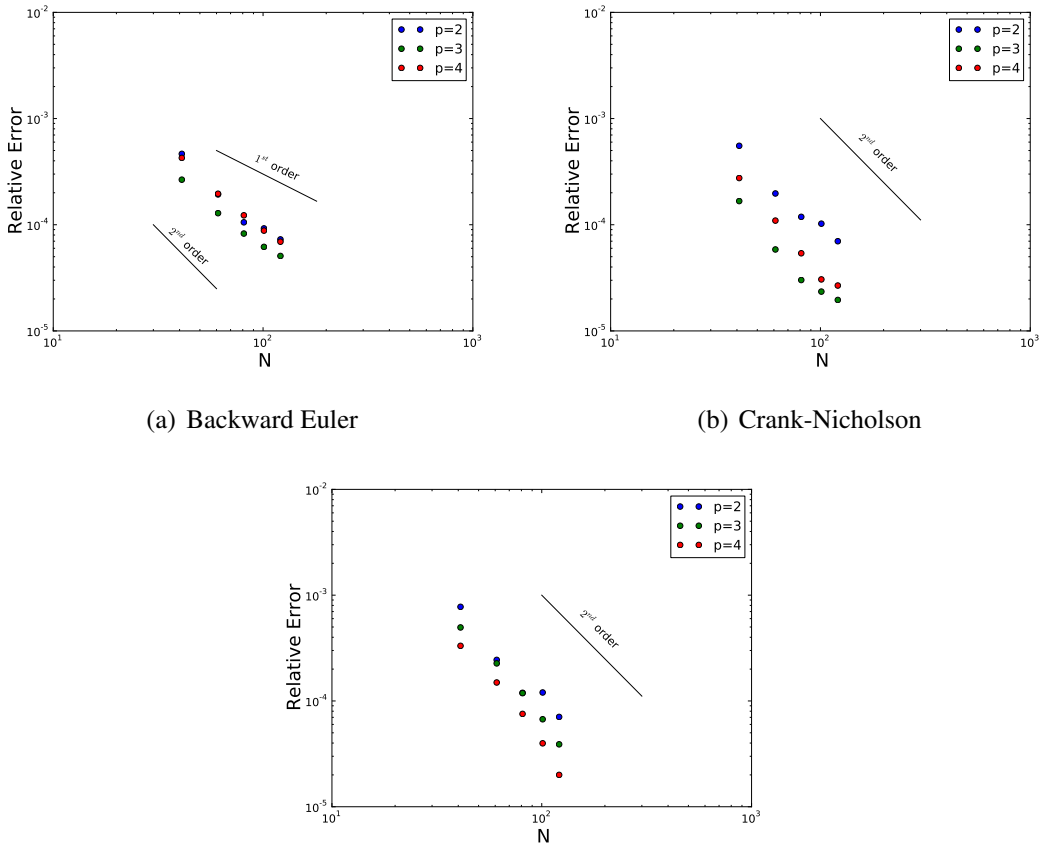
3.2 The Non-Local Cahn-Hilliard Equation

Having verified the correctness of the code for known test problems, the modified Cahn-Hilliard equation can be attempted. Before the code is trusted for problems on complex geometries, we consider the convergence of the method on the sphere.

3.2.1 Convergence Study

Though an analytical solution is not known for the mCH equation, a convergence study may still be obtained by taking a very fine discretization as the “exact" solution. In this case, the mCH solver was run on the unit sphere, the embedding grid was contained in a box of side $L = 4$ centered at the origin, and the three methods BE, CN, and BDF2 were utilized. The code was run for interpolations of order $p = 2,3,4$ and discretizations $N = 41,61,81,101,121$. The “exact” solution was a BDF2 run with $p = 4$ and $N = 161$. Finally, the time step restriction used was $k = \frac{1}{8}h^2$ where once again $h = \frac{L}{N-1}$.

Figure 3.3 presents the results. All three methods converge (to the “exact” solution). CN and BDF2 still converged to second order. BE was slightly less effective on this problem than on the heat and biharmonic heat equations but still managed an average negative slope of 1.64.



(a) Backward Euler

(b) Crank-Nicholson

(c) BDF2

Figure 3.3: Modified Cahn-Hilliard equation convergence plot: log-log plot of the max norm of the relative error against the discretization N. The order of convergence for BE has dipped slightly while CN and BDF2 are still performing as well as in previous test cases.

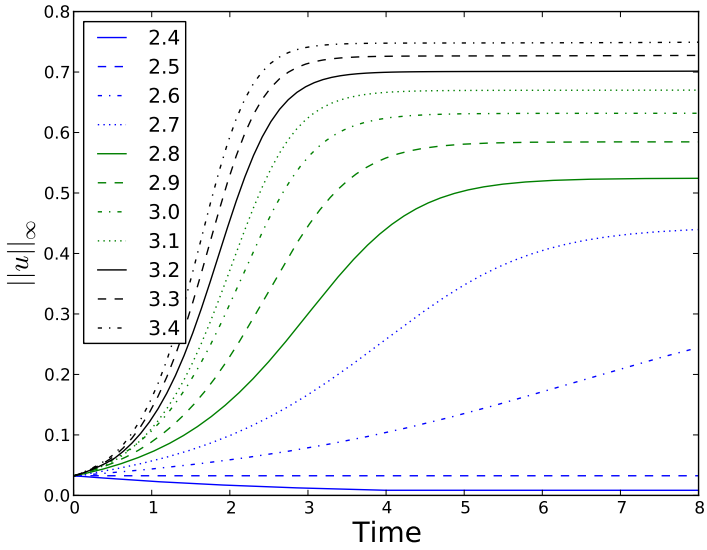


Figure 3.4: Evolution of the max-norm of the amplitude in time. The legend shows the different values of γ. Note the $\gamma = 2.4$ amplitude decays, and the $\gamma = 2.5$ amplitude persists while the $\gamma > 2.5$ amplitudes all grow exponentially to some plateau value.

3.2.2 Linear Stability Analysis and Bounded Amplitude

This section presents experiments to further validate the correctness of our implementation. On the one hand, we check that our code agrees with the linear stability analysis of Section 1.4.2 for small amplitudes, while simultaneously, we show that the non-linear term keeps the amplitude bounded.

On a sphere of radius $R = 2$, in an embedding box of $L = 6$, with a discretization of $N = 61$, performing a quadratic (i.e. $p = 2$) interpolation, the modified Cahn-Hilliard equation was run with initial condition $u(0) = \frac{1}{25} Y_4^3$. Multiple values of γ were tried but $m = 0$ was fixed. For these runs BDF2 was used with adaptive time-stepping.

Note that with these particular choices of R, l and m, the initial perturbation profile (i.e. Y_4^3) becomes unstable at precisely $\gamma = 2.5$; this value is obtained using (1.22). Indeed, at the critical value $\gamma = 2.5$, the profile no longer decays, but instead persists. Above this critical value, it grows exponentially. This is in agreement with Figure 3.4.

We further check that the growth rate is in agreement with (1.22). This is done for a

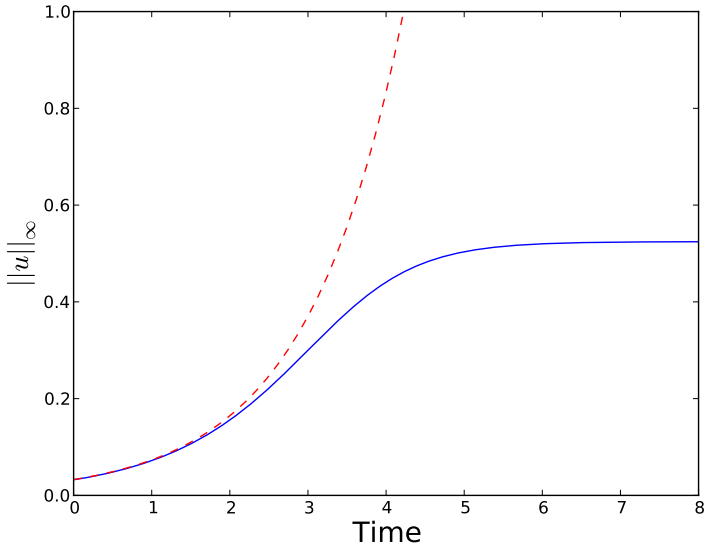


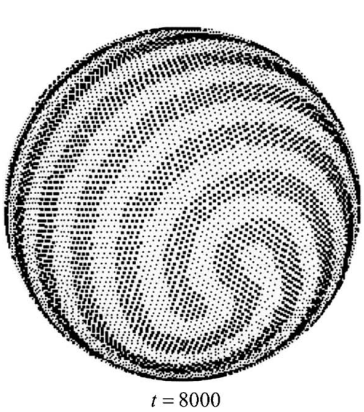
Figure 3.5: Comparison of the numerical solution to the full non-linear mCH equation (blue curve) to the analytic solution to the linear part of the mCH (red dashed curve). This suggests that the non-linear term prevents the solution from growing indefinitely.

sample run where $\gamma = 2.8$. Figure 3.5 plots both the analytic solution to the linear part of the mCH equation (red dashed curve) and the numerical solution to the full non-linear equation (blue curve). The two curves agree until the amplitude grows beyond approximately 0.2 where the non-linear terms are no longer negligible.

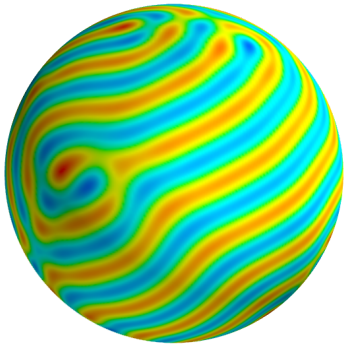
3.2.3 Qualitative Agreement with Other Numerical Experiments

In this section we compare our implementation to other numerical results to the mCH equation [TQZY05]. The approach taken by Tang *et al.* consists of partitioning the sphere into cells and running a finite volume method on that discretization. The method is very different yet we obtain very similar results as can be seen in Figure 3.6.

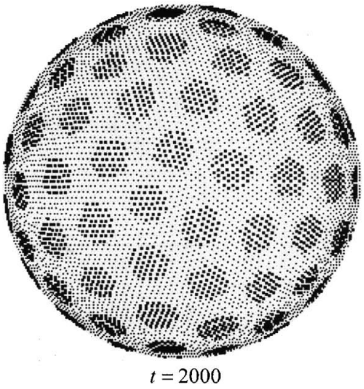
Our solutions are not steady states or energy minimizers but presumably computing for larger times might lead us to convincing energy minimizers like the ones obtained by Tang *et al.* We have obtained solutions in the presence of a white noise forcing term but we do not yet have conclusive evidence that the addition of space-time white noise helps reach a



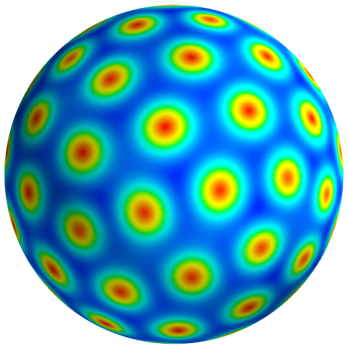
(a) Tang *et al.*



(b) Our result at $T = 100$



(c) Tang *et al.*



(d) Our result at $T = 100$

Figure 3.6: Comparison with Tang *et al.* experiments. Given a γ, for sufficiently large m (i.e. for a sufficiently asymmetric quantity of the different monomer types), the pattern obtained will feature close packed spots like the ones observed in Figures 3.6(c) and 3.6(d). Our runs took under 13 hours on a 32-bit 3 GHz processor.

global minimizer or a steady state. More experimenting with different amplitudes of noise is needed.

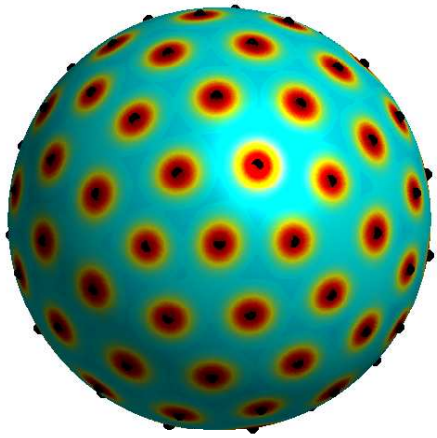


Figure 3.7: Demonstration of the domain finding algorithm. The computed centers of the spots are shown as big black dots.

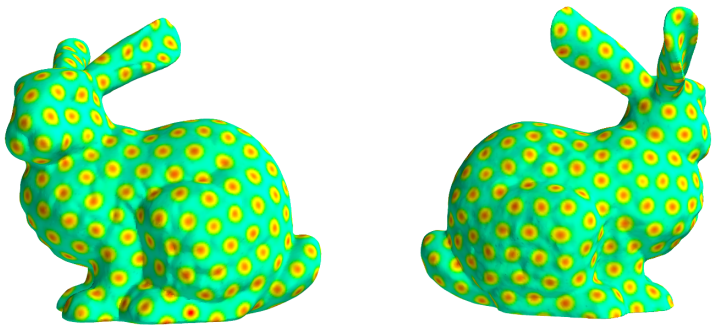
On the run featuring spots, our domain finding algorithm was run. In 25 seconds, it found and labeled 89 spots shown in Figure 3.7. Moreover, it tabulated that 21 spots had 5 neighbours, 63 had 6 neighbours, and 5 spots had 7 neighbours. There is a topological reason for a high number of 5-neighbour spots, however, according to [BVW10] for such a number of total spots, there should be no 7-neighbour spots. This might be due to the fact that we have not reached an energy minimizer.

3.3 General Smooth Surfaces

In this final section, we present results on a more complex surfaces. We have subsequently solved the mCH equation on multiple surfaces, including the torus, and other shapes from the AIM@SHAPE online repository [AIM]. For the purpose of demonstrating the geometric flexibility of our implementation, we show the solution on the Stanford bunny contained in a box centered at the origin, of side $L = 60$. The chosen mCH parameters were $(\gamma, m) = (2.8, -0.3)$, the discretization used was $N = 181$, and the interpolation was quadratic, i.e. $p = 2$. We prescribe normally distributed random noise $N(0, 0.1)$ at the

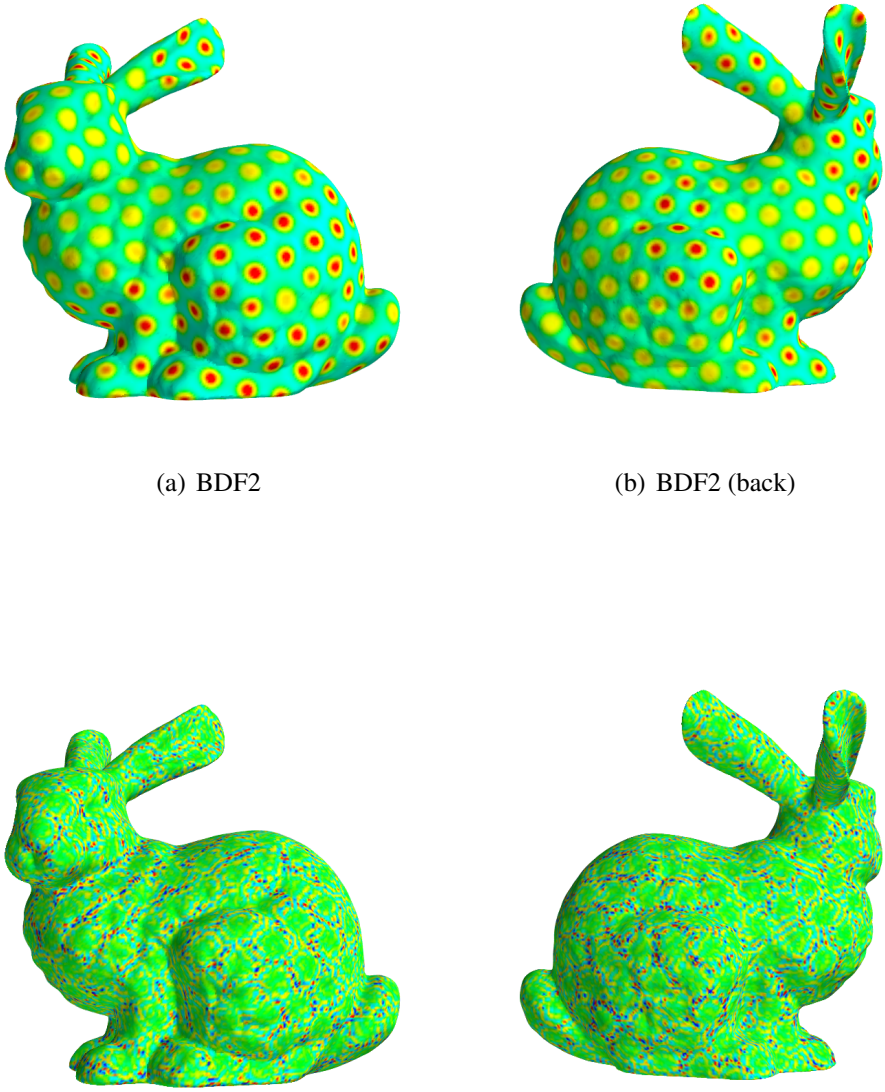
initial time step and evolve the equation using adaptive time-stepping with BDF2. The solution after 300 units of time is shown in Figure 3.8.

Figure 3.9 shows pathologies that can arise from allowing the time step to grow too large. Notice that for CN time-stepping high frequency error is introduced. This is due to the fact that Crank-Nicholson does not damp high frequency error nearly as rapidly as BDF2 and BE do.



(a) Front (b) Back

Figure 3.8: Implementation solving the mCH equation on the Stanford bunny. This run took just over 20 hours.



(a) BDF2 (b) BDF2 (back)

(c) CN (d) CN (back)

Figure 3.9: Pathologies that can arise from large time steps.

Chapter 4

Conclusion

This thesis presents strong evidence suggesting that our implementation is capable of producing an accurate approximation to the solution of the modified Cahn-Hilliard equation on general surfaces. This section summarizes the major results and proposes avenues for future work.

We have built an implementation in C of the CPM method along with time-stepping methods, as well as time adaptivity and domain finding tools described in Chapter 2. This implementation solved a fourth-order non-linear parabolic PDE called the modified Cahn-Hilliard equation.

We have shown that our implementation produces convergent solutions to the mCH equation on the sphere. Furthermore, our solutions were in qualitative agreement with results published by Tang et al. who implemented a completely different approach. Due to the non-linearity of the equation and the symmetry of the sphere a direct quantitative comparison of solutions (e.g. subtracting one from the other) is difficult. However, there exist indirect strategies based on global properties of the pattern, for instance, for solutions featuring spots, one can compare the number of spots on the domain, or even the number of spots with 5, 6, or 7 neighbours. Presumably, these quantities would be invariant under the sphere's rotational symmetries. In this thesis we propose a simple algorithm for spot finding and neighbour counting, though we do not make extensive use of it in this body of work.

The mCH equation was solved numerically using three different time stepping methods: BE, CN, and BDF2. CN and BDF2 are not only more accurate in time, but they solved the

problems in just under half the time. Though this has not yet been demonstrated, we believe it may be due to the higher accuracy, resulting in fewer GMRES iterations.

Our adapative time-stepping method allows our method to go from time steps on the order of 10^{-3} at the start of the computation to 10^{-1} when the solution evolves much more slowly. This helps reach steady states faster. Presently, it must be prevented from growing too large to avoid instability.

The space-time white noise was successfully added to our implementation and numerical solutions are reasonable. However, the addition of noise to the model substantially slows down the computation. This is due to the iterative solver taking more iterations to reach our tolerance. Therefore, we need to study more precisely the effect of the space-time white noise forcing on the solution and decide whether the slowdown is compensated by a faster approach to the global minimizer. Alternatively, we could consider adding white noise forcing only when the solution stops evolving to kick it out of a possibly local minimum of the energy.

Finally, we presented a numerical solution of the mCH on a general smooth surface, namely the Stanford bunny. Solutions are qualitatively reasonable when considering the solution on a sphere of similar size with the same mCH parameters (γ, m).

4.1 Future Work

There are many areas where our implementation can be improved in terms of performance and added functionality. The following lists a few examples.

4.1.1 Parallelization

Increasing efficiency should be very straightforward; this is because most of the computation time is spent on a single subroutine: the BLI function for the extension step. This is a very simple operation on each grid node of our computational domain and more importantly, it is tremendously parallelizable. By that we mean each grid node value can be computed separate from all other nodes. Rather than use a cluster, one can implement the BLI routine to run on a graphical processing unit (GPU) which is a single chip that holds up to 500 computing cores. It is a simple matter of learning one (or both) of the two most

popular APIs: CUDA or OpenCL.

Moreover, with a GPU or multiple CPU cores available, our code can use a CUDA or OpenCL implementation of GMRES, e.g. [fM].

4.1.2 Preconditioning

Presently, the GMRES solver adopted is a C implementation that takes in a left and a right preconditioner [Ber]. We currently have no preconditioners for our method but these could drastically decrease the number of iterations required to reach the desired tolerance and hence substantially shorten processing times.

4.1.3 Brushes

The Closest Point Method is not limited to computing on surfaces embedded in three dimensions. In the CPM framework it is possible to solve equations defined on volumes embedded in three dimensions, in other words, solving on a subset of $\mathbb{R}^3$ while extending to a larger subset of $\mathbb{R}^3$. Adding this functionality to our implementation would be interesting when considering diblock copolymer layers of non-negligible thickness on curved surfaces. These are known in the literature as brushes.

4.1.4 Boundary Conditions

Another aspect that has been overlooked thus far, is the consideration of boundary conditions (BC) and how they fit in the CPM. Indeed, for open surfaces, our current implementation simply adopts homogeneous Neumann boundary conditions which are imposed naturally by the method. Imposing other types of BCs is a feature yet to be added to the current code.

4.1.5 Effect of Curvature

Most of the results obtained in this thesis were limited to spherical domains, a shape with constant curvature. One can ask how non-constant curvature can affect the micro-structure. For instance, can patterns be made to align in a certain direction, or can one control the

position and alignment of defects by introducing bumps and valleys on the computational manifold?

4.1.6 Curvature Motion

For many applications, the dynamics of the modified Cahn-Hilliard equation are not needed, only the steady state (which is hopefully the global minimizer) is wanted. This means that we may apply external forces to our solution with hopes of decreasing the mCH energy. In the case of lamellae on the sphere for example, undulated lamellae as seen in Figure 3.6 seem to be energetically less favorable than spirals that follow geodesics more closely. Given this assumption, one may treat the solution with a few steps of curvature motion to smooth out excess curvature in the lamella.

4.1.7 Varying γ

Thus far we have not considered problems with a time-dependent $\gamma = \gamma(t)$. Since $\gamma \propto \chi N$ and $\chi \propto T^{-1}$ where T is temperature, decreasing γ corresponds to increasing the temperature of the diblock copolymer melt. Physically, this facilitates the motion of the molecules. Numerically, as can be seen in Figure 3.4, a smaller γ leads to smaller amplitudes which facilitates the motion of interfaces. As discussed in [BVW10], as γ is decreased, there is a pre-melting of defects; in other words the solution changes more easily around defects. Our hope is that varying γ and possibly even m will help the solution out of local minimizer states. Naturally the value of γ should be slowly varied back to its initial desired value.

Appendix A: The $H^1(S)$ Space

Some care must be taken when discussing functions in $H^1(S)$ since the notion of weak derivatives and integration by parts is more complex on S than on $\mathbb{T}^2$. We adopt Hebey's notion of the $H^1(S)$ space which he names $H^{p=2}_{k=1}(S)$ in [Heb96]. In the following we briefly define this important space.

Let $\mathscr{M}$ be a smooth manifold with metric g. Let $C^\infty(\mathscr{M})$ be the space of infinitely smooth functions on $\mathscr{M}$ and let $\mathscr{C}^1(\mathscr{M}) \subset C^\infty(\mathscr{M})$ be the space of such functions that have a continuous covariant derivative ∇u that is bounded in $L^2(\mathscr{M})$. $H^1(\mathscr{M})$ is defined to be the completion of $\mathscr{C}^1(\mathscr{M})$ with respect to the norm

$$||u||_{H^1(\mathscr{M})} := \left(\int_{\mathscr{M}} u^2 \sqrt{det(g_{ij})}dx\right)^{\frac{1}{2}} + \left(\int_{\mathscr{M}} |\nabla u|^2 \sqrt{det(g_{ij})}dx\right)^{\frac{1}{2}}. \tag{4.1}$$

Equation (4.1) is called the $H^1(\mathscr{M})$ norm.

On the sphere S and in spherical coordinates

$$\sqrt{det(g_{ij})}dx = R^2 \sin(\theta) d\phi d\theta. \tag{4.2}$$

Functions in $H^1(S)$ are not necessarily in $\mathscr{C}^1(S)$ which means $H^1(S)$ includes functions that have cusps, for example at boundary edges. Given the energy functional (1.4), we must look for a solution in $H^1(S)$, because $\mathscr{C}^1(S)$ does not include all solutions that have a bounded non-local Cahn-Hilliard energy.

Consider $u \in H^1(S)$, and a sequence $\{u_m\} \subset C^1(S)$ such that $u_m \to u$ with respect to the $H^1(S)$ norm. Then the quantity ∇u is defined in a weak sense as

$$\nabla u_m \rightharpoonup \nabla u. \tag{4.3}$$

Let $\phi \in C^\infty$, the previous weak convergence is to be understood as

$$\int_S \nabla u_m \phi R^2 d\Omega \to \int_S \nabla u \phi R^2 d\Omega. \tag{4.4}$$

Appendix B: GMRES convergence

When solving the biharmonic heat equation

$$\begin{cases} u_t = -0.1\Delta_S^2 u & \text{on } S \times (0,\infty) \\ u = Y_1^0 & \text{at } t = 0, \end{cases} \tag{4.5}$$

with a time-step $k = \frac{h}{8}$, the GMRES iterative solver was not converging to tolerance (10^{-6}) from the third time-step onward. Since the time-step in our actual runs was going to be updated adaptively, we did not put too much importance in the initial value of the time step.

The following results were obtained for $h = 0.05$. Figure 4.1 shows that as the time-step is decreased, the norm of the residual approaches the desired tolerance.

Figure 4.2 shows the number of iterations required to reach the tolerance with respect to the step-size condition. Though the step-size is halved and therefore the number of time-steps to take is doubled, the number of GMRES iterations necessary is not decreased proportionally. This is why in our biharmonic heat convergence studies, a condition $k = \frac{h^2}{2}$ is taken with a tolerance of 10^{-5}.

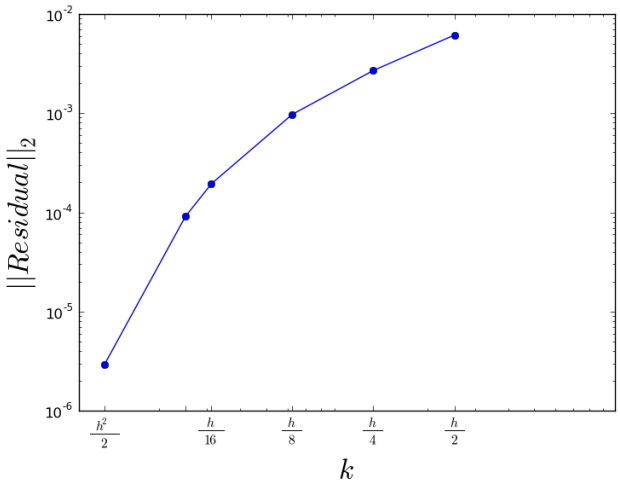


Figure 4.1: Adjusting the step size to reach tolerance of 10^{-6}. For different conditions on k, the blue dots represent values of the norm of the residual at which further (GMRES) iteration no longer lowers the residual.

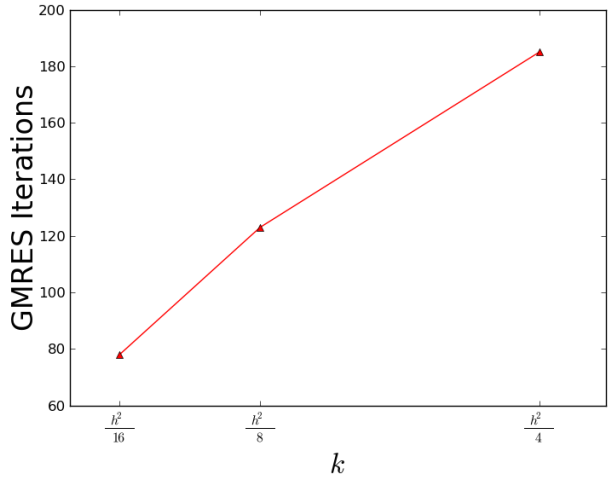


Figure 4.2: GMRES Iterations required to reach tolerance depending on the step-size. Notice that as the step-size is decreased after $\frac{h^2}{4}$, a further decrease is not beneficial.

Bibliography

[AIM] AIM@SHAPE. http://shapes.aim-at-shape.net.

[ARW95] U. M. Ascher, S. J. Ruuth, and B. T. R Wetton. Implicit-explicit methods for time-dependent PDE's. *SIAM J. Numer. Anal.*, 32:797–823, 1995.

[Ber] Zuse Institute Berlin. http://www.zib.de/en/nc/home.html.

[BF99] F. S. Bates and G. H. Fredrickson. Block copolymers – designer soft materials. *Physics Today*, 52(2):32–38, 1999.

[BT04] J.-P. Berrut and L. N. Trefethen. Barycentric Lagrange Interpolation. *SIAM Review*, 46(3):501–517, 2004.

[BVW10] R. Backofen, A. Voigt, and T. Witkowski. Particles on curved surfaces: A dynamic approach by a phase-field-crystal model. *Physical Review E*, 81:025701, 2010.

[CPW09] R. Choksi, M. A. Peletier, and J. F. Williams. On the phase diagram for microphase separation of diblock copolymers: An approach via a nonlocal Cahn-Hilliard functional. *SIAM J. Appl. Math.*, 69(6):1712–1738, April 2009.

[Eyr97] D. J. Eyre. An unconditionally stable one-step scheme for gradient systems. *Unpublished*, 1997.

[fM] Institute for Microelectronics. http://viennacl.sourceforge.net.

[Heb96] Emmanuel Hebey. *Sobolev Spaces On Riemannian Manifolds*. Springer, 1996.

[Mar08] M. Maraš. The 2D phase diagram assiciated with a PDE model for the self-assembly of diblock copolymers. Master's thesis, Simon Fraser University, Burnaby, BC, 2008.

[MB96] M. W. Matsen and F. S. Bates. Unifying weak- and strong-segregation block copolymer theories. *Macromolecules*, 29(4):1091–1098, 1996.

[MR08] C. B. Macdonald and S. J. Ruuth. Level set equations on surfaces via the Closest Point Method. *J. Sci. Comput.*, 35(2-3):219–240, 2008.

[MR09] C. B. Macdonald and S. J. Ruuth. The implicit Closest Point Method for the numerical solution of partial differential equation on surfaces. *SIAM J. Sci. Comput.*, 31(6):4330–4350, 2009.

[PD96] G. Da Prato and A. Debussche. Stochastic Cahn-Hilliard equation. *Nonlinear Analysis, Theory, Methods & Applications*, 26(2):241–263, 1996.

[RM08] S. J. Ruuth and B. Merriman. A simple embedding method for solving partial differential equations on surfaces. *J. Comput. Phys.*, 227(3):1943–1961, 2008.

[TQZY05] P. Tang, F. Qiu, H. Zhang, and Y. Yang. Phase separation patterns for diblock copolymers on spherical surfaces: A finite volume method. *Phys. Rev. E*, 72:016710, 2005.