The closest point method for the numerical solution of partial differential equations on moving surfaces

by

Argyrios Petras

B.Sc., National Technical University of Athens, 2012

Dissertation Submitted in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy in the Department of Mathematics Faculty of Science

© Argyrios Petras 2016
SIMON FRASER UNIVERSITY
Fall 2016

All rights reserved. However, in accordance with the Copyright Act of Canada, this work may be reproduced without authorization under the conditions for “Fair Dealing.” Therefore, limited reproduction of this work for the purposes of private study, research, criticism, review and news reporting is likely to be in accordance with the law, particularly if cited appropriately.
# Approval

<table>
<thead>
<tr>
<th>Name</th>
<th>Argyrios Petras</th>
</tr>
</thead>
<tbody>
<tr>
<td>Degree</td>
<td>Doctor of Philosophy (Mathematics)</td>
</tr>
<tr>
<td>Title</td>
<td><em>The closest point method for the numerical solution of partial differential equations on moving surfaces</em></td>
</tr>
<tr>
<td>Examining Committee</td>
<td>Chair: Paul Tupper</td>
</tr>
<tr>
<td></td>
<td>Professor</td>
</tr>
<tr>
<td></td>
<td>Steven J. Ruuth</td>
</tr>
<tr>
<td></td>
<td>Senior Supervisor</td>
</tr>
<tr>
<td></td>
<td>Professor</td>
</tr>
<tr>
<td></td>
<td>Nilima Nigam</td>
</tr>
<tr>
<td></td>
<td>Supervisor</td>
</tr>
<tr>
<td></td>
<td>Professor</td>
</tr>
<tr>
<td></td>
<td>Mary-Catherine Kropinski</td>
</tr>
<tr>
<td></td>
<td>Supervisor</td>
</tr>
<tr>
<td></td>
<td>Professor</td>
</tr>
<tr>
<td></td>
<td>John Stockie</td>
</tr>
<tr>
<td></td>
<td>Internal Examiner</td>
</tr>
<tr>
<td></td>
<td>Professor</td>
</tr>
<tr>
<td></td>
<td>Jean-Christophe Nave</td>
</tr>
<tr>
<td></td>
<td>External Examiner</td>
</tr>
<tr>
<td></td>
<td>Associate Professor</td>
</tr>
<tr>
<td></td>
<td>Department of Mathematics and Statistics</td>
</tr>
<tr>
<td></td>
<td>McGill University</td>
</tr>
<tr>
<td>Date Defended</td>
<td>8 December 2016</td>
</tr>
</tbody>
</table>
Abstract

Partial differential equations (PDEs) on surfaces arise in a wide range of applications. The closest point method is a recent embedding method that has been used to solve a variety of PDEs on smooth surfaces using a closest point representation of the surface and standard Cartesian grid methods in the embedding space. The original closest point method (CPM) was designed for problems posed on static surfaces, however the solution of PDEs on moving surfaces is of considerable interest as well. Here we propose two different approaches for solving PDEs on moving surfaces using a combination of the CPM and a grid based particle method. The grid based particle method (GBPM) represents and tracks surfaces using meshless particles and an Eulerian reference grid. In our first approach, a modification of the GBPM is introduced to ensure that all the grid points within a computational tube surrounding the surface are active. The modified GBPM is tested in geometric motions of surfaces to verify the correctness of the new algorithm. A coupled method is proposed combining the modified GBPM and the CPM and tested on a number of numerical examples. Our second approach uses generalized finite difference schemes derived from radial basis functions (RBF-FD) in the implementation of the closest point method. An explicit and an implicit formulation of the CPM using RBF-FD are presented along with numerical experiments for the convergence of the method, including the approximation of the solution of reaction-diffusion equations and the Cahn-Hilliard equation on a variety of surfaces. Finally, a coupled method of the CPM that uses RBF-FD and the original GBPM is proposed and used to solve PDEs on moving surfaces.

Keywords: closest point method; grid based particle method; partial differential equations on moving surfaces; radial basis functions finite differences; embedding method; Lagrangian particles
Acknowledgements

I thank my supervisor Steve Ruuth for introducing to me the closest point method. I acknowledge his support and his guidance throughout the PhD degree. I also thank Nilima Nigam for her useful advice and her guidance. In addition, I thank Mary-Catherine Kropinski for being in the supervisory committee and for her time.

I thank Leevan Ling for introducing to me the radial basis functions approximation and for his time to explain it to me in detail.

Finally, I would like to acknowledge my family for their support and my friends and colleagues that were always there when the going got tough.
# Table of Contents

Approval ii
Abstract iii
Acknowledgements iv
Table of Contents v
List of Tables viii
List of Figures x

1 Introduction 1
  1.1 Introduction .......................................................... 1
  1.2 Thesis outline ....................................................... 4

2 Theoretical background 5
  2.1 Differential geometry .............................................. 5
      2.1.1 Surface representation ........................................ 5
      2.1.2 Unit normal and unit tangent vectors ....................... 6
      2.1.3 Curvature ..................................................... 8
  2.2 Closest point representation of surfaces ....................... 10
  2.3 Surface differential operators ................................... 11
  2.4 Radial basis functions (RBF) approximation .................... 13
      2.4.1 The RBF-GA method ........................................... 15
  2.5 Time stepping methods ............................................. 16
      2.5.1 Explicit time stepping methods ............................. 17
      2.5.2 Implicit time stepping methods ............................. 18
      2.5.3 Implicit-explicit (IMEX) time stepping methods ............ 18

3 Problem statement 20
  3.1 Surface integration ............................................... 20
  3.2 PDE model derivation ............................................. 23
4 Numerical methods
   4.1 The closest point method .................................. 26
   4.2 The grid based particle method .............................. 28
       4.2.1 The Resampling Step .................................. 30

5 The closest point method coupled with a modified grid based particle method
   5.1 A modified grid based particle method ....................... 33
       5.1.1 Modification ............................................ 33
   5.2 Numerical tests of modified GBPM ............................. 35
       5.2.1 Mean curvature motion on a circle and a sphere ....... 36
       5.2.2 Vortex flow with rewind ................................ 37
       5.2.3 Merging circles ........................................ 39
       5.2.4 Vortex flow with rewind in 3D .......................... 39
       5.2.5 Topological change in 3D .............................. 40
   5.3 A coupled method ........................................... 43
   5.4 Numerical experiments ....................................... 45
       5.4.1 Diffusion on an expanding circle ....................... 45
       5.4.2 Advection-diffusion on an oscillating sphere .......... 48
       5.4.3 Strongly coupled flow on an evolving torus ........... 52
       5.4.4 A model for solid tumor growth ......................... 52

6 An explicit closest point method using RBF-FD
   6.1 An explicit formulation of the closest point method using RBF-FD .... 56
       6.1.1 Method’s description ................................... 56
       6.1.2 Parameters .............................................. 58
       6.1.3 Computational tube ...................................... 60
   6.2 Numerical experiments ....................................... 62
       6.2.1 Heat equation on a circle ............................... 63
       6.2.2 Advection equation on an ellipse ........................ 63
       6.2.3 Advection-diffusion equation on an ellipse ............ 64
       6.2.4 Heat equation on a sphere ............................... 65
       6.2.5 Advection on a torus .................................... 66
       6.2.6 Image denoising on a sphere ............................. 66
       6.2.7 Reaction-diffusion systems .............................. 67

7 An implicit RBF-FD discretization of the closest point method
   7.1 An implicit formulation of the closest point method using RBF-FD .... 70
   7.2 Numerical experiments on static surfaces ..................... 72
       7.2.1 Heat equation on a circle ............................... 72
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.2.2</td>
<td>Heat equation on a sphere</td>
<td>73</td>
</tr>
<tr>
<td>7.2.3</td>
<td>Reaction-diffusion systems</td>
<td>74</td>
</tr>
<tr>
<td>7.2.4</td>
<td>Cahn-Hilliard on a sphere</td>
<td>75</td>
</tr>
<tr>
<td>7.3</td>
<td>A coupled method</td>
<td>78</td>
</tr>
<tr>
<td>7.4</td>
<td>Numerical experiments on moving surfaces</td>
<td>79</td>
</tr>
<tr>
<td>7.4.1</td>
<td>Diffusion on an expanding circle</td>
<td>80</td>
</tr>
<tr>
<td>7.4.2</td>
<td>Advection-diffusion on an oscillating sphere</td>
<td>81</td>
</tr>
<tr>
<td>7.4.3</td>
<td>A cross-diffusion reaction-diffusion system</td>
<td>83</td>
</tr>
<tr>
<td>7.4.4</td>
<td>Cahn-Hilliard equation on an ellipsoid</td>
<td>85</td>
</tr>
</tbody>
</table>

8 Conclusions 86

Bibliography 89

Appendix A The Gauss circle problem 95

Appendix B Sample code 97
  B.1 Sample MATLAB code for the RBF-FD implementation of the CPM 97
  B.2 Sample MATLAB code for the implicit RBF-FD discretization of the CPM 99
List of Tables

Table 2.1 Definition of some commonly used RBFs. ........................................... 14

Table 5.1 Absolute errors as measured in the infinity norm and the estimated order of convergence (e.o.c.) at various times $t$. ................................. 48

Table 5.2 Absolute error as measured in the infinity norm and the estimated order of convergence (e.o.c.) at times $t = 0.01, 0.02, 0.03$ and $0.04$. . . 49

Table 6.1 The number of lattice points $m$ contained in a ball with radius $r = \sqrt{q}$ in two and three dimensions. .................................................. 61

Table 6.2 Computational tube radius $\gamma$ for an $m$-point RBF-FD stencil in two and three dimensions. .................................................. 62

Table 6.3 Relative error of the approximate solution at time $t = 1$ for the heat equation on the unit circle as measured in the infinity norm and the convergence rates. .................................................. 63

Table 6.4 Relative error of the approximate solution at time $t = 1$ for the advection equation on an ellipse as measured in the infinity norm and the convergence rates. .................................................. 64

Table 6.5 Relative error of the approximate solution at time $t = 1$ for the advection-diffusion equation on an ellipse as measured in the infinity norm and the convergence rates. .................................................. 65

Table 6.6 Relative error of the approximate solution at time $t = 1$ for the heat equation on the unit sphere as measured in the infinity norm and the convergence rates. .................................................. 65

Table 6.7 Relative error of the approximate solution at time $t = 1$ for the advection equation on a torus as measured in the infinity norm and the convergence rates. .................................................. 66

Table 7.1 Relative error of the approximate solution at time $t = 1$ for the heat equation on the unit circle as measured in the infinity norm and the convergence rates. .................................................. 73

Table 7.2 Relative error of the approximate solution at time $t = 1$ for the heat equation on the unit sphere as measured in the infinity norm and the convergence rates. .................................................. 74
Table 7.3  Relative errors as measured in the infinity norm and the estimated order of convergence (e.o.c.) at various times $t$.

Table 7.4  Relative errors as measured in the infinity norm and the estimated order of convergence (e.o.c.) at various times $t$. 
## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 3.1</td>
<td>An illustration of a surface $M$ and the tubular neighborhood $\Omega_{\epsilon}$.</td>
<td>23</td>
</tr>
<tr>
<td>Figure 4.1</td>
<td>An illustration of the main steps of the GBPM (from top left to bottom right). Active grid points (blue dots) are connected to their footpoints (red dots) with blue lines. The green lines correspond to the grid.</td>
<td>29</td>
</tr>
<tr>
<td>Figure 5.1</td>
<td>The local grid based particle method reconstruction (blue line) using $m = 6$ footpoints (blue circles). The minimizer of distance (black star) lies outside the interpolating points. This leads to the deactivation of grid point $p$ (blue point).</td>
<td>34</td>
</tr>
<tr>
<td>Figure 5.2</td>
<td>Left: The osculating circle reconstruction (red line) from three points (red circles) and the minimizer of distance (black star). The minimizer is chosen as the new footpoint. Right: A zoom-in of the osculating circle reconstruction.</td>
<td>35</td>
</tr>
<tr>
<td>Figure 5.3</td>
<td>Blue lines give the computed radii as a function of time for the circle (left) and the sphere (right). Red dots mark the exact solution at selected times.</td>
<td>36</td>
</tr>
<tr>
<td>Figure 5.4</td>
<td>The error of the approximate radius in comparison to the theoretical radius in the $\infty$-norm over time for the circle (left) and the sphere (right).</td>
<td>37</td>
</tr>
<tr>
<td>Figure 5.5</td>
<td>The evolution of a circle under vortex flow with rewind at various times $t$ (from top left to bottom right).</td>
<td>38</td>
</tr>
<tr>
<td>Figure 5.6</td>
<td>Two circles moving outwards with constant unit normal speed. A merger occurs.</td>
<td>39</td>
</tr>
<tr>
<td>Figure 5.7</td>
<td>The evolution of a sphere under vortex flow with rewind at selected times $t$. The final time is $T = 1.5$.</td>
<td>41</td>
</tr>
<tr>
<td>Figure 5.8</td>
<td>The evolution of a dumbbell under mean curvature motion at selected times $t$.</td>
<td>42</td>
</tr>
<tr>
<td>Figure 5.9</td>
<td>The graph of the approximate solution of the diffusion model on the evolving circle at selected times $t$. The solution is depicted in relation to the surface parameter $\theta$.</td>
<td>46</td>
</tr>
</tbody>
</table>
Figure 5.10  The graph of the approximation of the solution of the diffusion model on the evolving circle at selected times $t$. The solution is shown in relation to the $x$ and $y$ coordinates. 47

Figure 5.11  The absolute error of the numerical solution of the diffusion model on the expanding circle over time. Errors are computed using the analytical solution. 48

Figure 5.12  The graph of the numerical solution of an advection-diffusion model on an oscillating ellipsoid at various times $t$. The solution is plotted as a graph of the surface parameters $\theta$ and $\phi$. Yellow corresponds to large solution values and dark blue corresponds to small values. 50

Figure 5.13  The graph of the numerical solution of an advection-diffusion model on an oscillating ellipsoid at various times $t$. The solution is visualized on the evolving surface. Yellow corresponds to large solution values and dark blue corresponds to small values. 51

Figure 5.14  The absolute error of the numerical solution of the advection-diffusion model on the oscillating ellipsoid over time for various mesh spacings. 51

Figure 5.15  A visualization of the numerical solution of the strongly coupled flow at selected times $t$. The initial surface is a torus. 53

Figure 5.16  A visualization of the numerical solution $u$ of the tumor growth model (5.3) at selected times $t$. The initial surface is a sphere. 54

Figure 5.17  A visualization of the numerical solution $w$ of the tumor growth model (5.3) at selected times $t$. The initial surface is a sphere. 55

Figure 6.1  The infinity norm relative error as a function of $\epsilon$ for the numerical approximation of the Laplace-Beltrami operator in two dimensions (left) and three dimensions (right). 59

Figure 6.2  Relative error as a function of grid spacing $\Delta x$ for various stencils for the approximation of the Laplace-Beltrami operator in two dimensions (left) and three dimensions (right). 60

Figure 6.3  An example of a 13-point stencil (black points with red circles) for the approximation of a differential operator for a point on the interface (green point). The red transparent circle centered at the surface node contains the closest grid points to the reference surface node. 61

Figure 6.4  Two cases of a surface node (black x) placement in a mesh grid (blue dots). 62

Figure 6.5  The initial image (top) is warped on a sphere and Gaussian noise is added. The noisy image (left) and the denoised image (right) are shown after 120 iterations on 1280598 points. 68
Figure 6.6  The solution of the Gray-Scott reaction-diffusion model for parameters \((k, F) = (0.06, 0.037)\) (left) and \((k, F) = (0.062, 0.03)\) (right).

Figure 7.1  The mean value of the relative error as measured in the \(\infty\)-norm for various spatial step sizes \(\Delta x\).

Figure 7.2  The mean value of the relative error as measured in the \(\infty\)-norm for various spatial step sizes \(\Delta x\).

Figure 7.3  The solution of the reaction-diffusion system on a bumpy torus. The parameters chosen are \(a = 3\), \(b = 10.2\) and \(\mu = 10/900\). Finally, \(\nu = 5/900\) on the left figure and \(\nu = 3.8/900\) on the right figure.

Figure 7.4  The solution of the Cahn-Hilliard equation on the unit sphere for various times \(t\). Yellow and blue colors correspond to high and low concentrations respectively.

Figure 7.5  The relative error of the numerical solution of the diffusion model on the expanding circle over time. Errors are computed using the analytical solution.

Figure 7.6  The relative error of the numerical solution of the diffusion model on the oscillating sphere over time. Errors are computed using the analytical solution.

Figure 7.7  A visualization of the numerical solution \(u\) of the cross-diffusion reaction-diffusion system (7.10) at selected times \(t\).

Figure 7.8  A visualization of the numerical solution \(u\) of the Cahn-Hilliard PDE (3.7) at selected times \(t\).

Figure A.1  Left: Five lattice points lie within a circle of radius 1. Right: Thirteen lattice points lie within a circle of radius 2.
Chapter 1

Introduction

Partial differential equations (PDEs) are ubiquitous in the sciences. In this thesis, we are interested in PDEs defined on moving surfaces and we develop numerical methods for approximating the solution of such PDEs by combining two well established methods: the closest point method [70] and the grid based particle method [51].

1.1 Introduction

Partial differential equations on surfaces arise in many applications in the natural and applied sciences. Applications can be found in image processing such as placing an image on a surface [77], restoring a damaged pattern on a surface [11], image segmentation on surfaces [76] and image denoising on surfaces [13]. In biology, applications include patterning on animal coats [63] and wound healing [65]. In computer graphics, applications can be found in the topic of real time fluid visualization on surfaces [8].

Partial differential equations on moving surfaces appear throughout the sciences. Biological applications include the modelling of biomembranes [23], cell motility and chemotaxis [25], and pattern formation [10, 79]. In material science, PDEs on moving surfaces arise in dealloying [22], while in fluid dynamics they appear in the modelling of the mass conservation/transportation of surfactants in two phase flows [4, 45]. A wide range of applications also appear in computer graphics. See [9] for some examples.

Many methods have been developed to solve PDEs on surfaces. There are methods applied on parametrized surfaces, on triangulated surfaces and on surfaces embedded in a higher dimensional space. Solution of PDEs on parametrized surfaces can be efficient for surfaces where a parametrization is possible [54, 28], however a parametrization of a surface often leads to distortions of the surface and singularities [28]. Triangulated surfaces avoid these issues. Finite difference methods can be applied to solve PDEs on triangulated surfaces [77], but there are difficulties in the calculation of geometric quantities, including the normal vector and the curvature of a surface [12]. On the other hand, finite element
methods are effective in solving parabolic or elliptic PDEs [19]. As for the embedded surfaces class, the surface and the corresponding surface PDE are extended in a d-dimensional space where standard Cartesian methods can be applied. A popular method employs a level set representation of surfaces and the use of a projection operator to solve surface PDEs [42, 30]. Typically, artificial boundary conditions need to be introduced to use a computational tube around the surface which can lead to a reduction in the accuracy [15]. Meshfree approximations using radial basis functions (RBFs) are also becoming popular in the embedded surfaces class [38, 74].

A variety of methods have been used to solve PDEs on moving surfaces. For example, Dziuk and Elliott [18] used moving triangle meshes as part of a finite element method for conservation laws. Their method is an example of a surface finite element method. (See [21] for further details and references related to the numerical analysis and applications of surface finite elements.) While finite element methods are commonly used with moving triangle meshes, other discretizations have also been proposed. For example, Nemadjieu [64] used a finite volume discretization to solve advection-diffusion equations on moving triangle meshes.

Particle and parametrization methods have also been used to represent and move surfaces. Of particular interest to us is the work of Leung and Zhao [51], where these concepts are combined into the grid based particle method. The grid based particle method was used in combination with a local parametrization method to solve advection-diffusion equations on moving surfaces in [49]. It was also used with the Gaussian beam summation method to approximate the high frequency asymptotic solution of the Helmholtz equation on moving surfaces in [52]. We provide a detailed description of the grid based particle method in Section 4.2.

Level set methods [73, 66] are another major class of methods for representing and moving surfaces. In the level set approach, the PDE-on-surface problem is solved on a narrow computational domain surrounding the surface. Methods of this type include the semi-implicit finite difference scheme of Xu and Zhao [82] and the finite element discretization of Dziuk and Elliott [20]. Related to the level set representation of a surface, is the closest point representation of a surface, whereby grid nodes store the closest point in Euclidean distance to the surface. Closest point representations have been used in the computation of diffusion-generated motions of curves on surfaces [61]. Closest point representations constructed from moving triangulated surfaces have been used in a closest point method for the Navier-Stokes and wave equations in [9]. Closest point representations constructed from level sets have also been used; see [46] for a closest point method for irrotational flow on moving surfaces. We provide a detailed description of the closest point method [70] in Section 4.1.

In this thesis, we explore two different approaches to solve PDEs on moving surfaces using the closest point method and the grid based particle method. The closest point
method is a simple embedding method for solving rather general PDEs on smooth stationary surfaces. It decouples surface geometry and PDE evolution via a closest point extension step, yielding a method that involves standard Cartesian grid methods in the embedding space. The original method only considered stationary surfaces, however. Complementing the closest point method is the grid based particle method. In the grid based particle method, a closest point representation is formed as part of an algorithm to move surfaces according to curvature-dependent (and higher-order [49]) motions. The grid based particle method is capable of tracking the motion of open surfaces [50] as well as surfaces where topological changes occur [51]. In addition, the method is capable of tracking motions of surfaces with corners.

Other methods are capable of tracking the motion of surfaces, i.e. the level set method [66] or even the use of moving triangle meshes. Both alternatives would require an extra step for the calculation of the closest point representation of the surface at each time step, which can be computationally expensive. On the other hand, the level set method can be implemented to be unconditionally stable and coupled with the implicit closest point method. In addition, level set method is capable of providing higher than second-order tracking of a moving smooth surface, which makes it desirable for the approximation of smooth surfaces.

In our first approach, we combine the closest point method and a (small) modification of the grid based particle method to solve PDEs on moving surfaces. Our proposed method is a highly modular combination of two well-known, tested methods. Indeed, in our approach, the component methods are applied in an alternating fashion, yielding a particularly simple method to solve PDEs on moving surfaces.

In our second approach, we introduce a new extension in the closest point method framework by replacing the standard finite difference schemes and the barycentric Lagrangian interpolation with finite difference stencils derived from radial basis functions approximations (RBF-FD). By using these generalized finite difference stencils, second-order schemes can be achieved with smaller computational tubes around the surface compared to the standard finite difference discretization of the closest point method. This results in fewer points for the sampling of the surface. Another advantage is the simplicity of obtaining higher order schemes by increasing the number of points on the finite difference stencil. The interpolation order does not reduce the order of approximation of the differential operators since the interpolation matrix is built-in the differential operator’s approximation.

Next, we introduce an implicit formulation of the closest point method using RBF-FD. The issue of stabilizing the closest point method has been addressed before [58, 80], however we consider a different approach. By enforcing the constant normal extension of the solution in a separate equation, we obtain a least squares approach to stabilize the closest point method using RBF-FD. We test this approach in a variety of examples to show its convergence using implicit and implicit-explicit time stepping methods. Our tests
include examples where the computational tube surrounding the surface does not contain all the grid points that lie within.

Finally, we perform a simple coupling of the implicit formulation of the closest point method using RBF-FD with the grid based particle method to solve PDEs on moving surfaces.

1.2 Thesis outline

This thesis is organized as follows. Chapter 2 provides a theoretical background on differential geometry, radial basis functions approximations and numerical differentiation methods.

The description and the derivation of the conservation law considered throughout this thesis appear in Chapter 3 while Chapter 4 states a review of the numerical methods that are used to approximate the solution of PDEs on moving surfaces. Chapter 5 presents the first approach of coupling the closest point method and the grid based particle method by introducing a modification in the second method. Numerical results are presented to verify the validity of the modified grid based particle method and to show the convergence of the coupled method. Chapter 6 describes an explicit formulation of the closest point method using RBF-FDs and applies the method to PDEs on static surfaces. Chapter 7 introduces an implicit formulation of the closest point method using RBF-FDs and presents a simple coupling of the proposed method with the grid based particle method. Numerical experiments for PDEs on a variety of static and moving surfaces are presented. Finally, Chapter 8 expands on the conclusions and explores potential future work.
Chapter 2

Theoretical background

Partial differential equations on moving surfaces involve geometric quantities, such as the unit normal vector and mean curvature, as well as surface differential operators. To approximate the solution of PDEs on moving surfaces, we use numerical methods, including radial basis function interpolation and time stepping methods. In this chapter, we review some important definitions and concepts that we use throughout this thesis.

2.1 Differential geometry

In this section, the notion of a surface and its representations are presented. Also, a review of geometric quantities defined on surfaces is provided.

2.1.1 Surface representation

Following [3], the definitions of a curve and a surface are given.

Definition 1. A local curve is a piecewise differentiable mapping

\[ x : \mathcal{U} \rightarrow \mathbb{R}^n \]

where \( \mathcal{U} \) is an open set in \( \mathbb{R} \). If \( \mathcal{U} \) is an open set in \( \mathbb{R}^2 \) then the mapping is called a local surface.

Throughout this thesis, we work with curves in \( \mathbb{R}^2 \) and \( \mathbb{R}^3 \) and surfaces in \( \mathbb{R}^3 \).

As stated in Definition 1, a surface can be represented explicitly using two variables \( x(u, v) \), where \( (u, v) \in \mathcal{U} \). This representation is called parametric. In addition, there is an implicit representation of surfaces given by a level set of a function \( \phi : \mathbb{R}^3 \rightarrow \mathbb{R} \), i.e.

\[ \phi(x, y, z) = c, \]

where \( c \in \mathbb{R} \).
Similarly, by following Definition 1, the parametric representation of a curve has the form \( x(s) \), where \( s \in \mathbb{R} \). In the Euclidean plane \( \mathbb{R}^2 \), the implicit representation of a curve is given by a function \( \phi : \mathbb{R}^2 \rightarrow \mathbb{R} \), i.e.

\[
\phi(x, y) = c,
\]

where \( c \in \mathbb{R} \). In \( \mathbb{R}^3 \), a curve can be implicitly defined as an intersection of two surfaces, i.e.

\[
\phi_1(x, y, z) = c_1 \quad \text{and} \quad \phi_2(x, y, z) = c_2,
\]

with \( c_1, c_2 \in \mathbb{R} \).

A commonly used level set function is called the **signed distance function**. For a closed surface \( \Gamma \), this function is defined as follows:

\[
dist(x) = \begin{cases} 
\min_{z \in \Gamma} \| x - z \|, & x \text{ in the interior of } \Gamma, \\
0, & x \in \Gamma, \\
-\min_{z \in \Gamma} \| x - z \|, & x \text{ outside } \Gamma.
\end{cases} \tag{2.1}
\]

By definition, the zero level set of the function \( dist(x) \) provides a representation of the surface \( \Gamma \). At the points where the signed distance function is differentiable, it satisfies

\[
\| \nabla dist(x) \| = 1.
\]

### 2.1.2 Unit normal and unit tangent vectors

For any point on a surface, two types of vectors can be defined: the unit normal vector and the unit tangential vector(s). Based on [40], definitions of these vectors for surfaces, planar curves and space curves are provided.

**Definition 2.** The **unit tangential vectors** of a surface \( x(u, v) \) at the point \( (u_0, v_0) \) are defined by

\[
T_1(u_0, v_0) = \frac{x_u}{\|x_u\|}(u_0, v_0) \quad \text{and} \quad T_2(u_0, v_0) = \frac{x_v}{\|x_v\|}(u_0, v_0).
\]

The **unit normal vector** of a surface is defined by

\[
n(u_0, v_0) = \frac{x_u \times x_v}{\|x_u \times x_v\|}(u_0, v_0).
\]

Similarly, for any point on a curve, the unit normal vector and the unit tangent vector are given in Definition 3:
Definition 3. The unit tangent vector of a curve $x(s)$ defined in $\mathbb{R}^2$ at the point $s_0$ is defined by
\[ T(s_0) = \frac{x_s(s_0)}{\|x_s(s_0)\|}. \]
The unit normal vector of a curve is defined by
\[ n(s_0) = \frac{T(s_0)}{\|T(s_0)\|}. \]

For a curve defined in $\mathbb{R}^3$, along with the unit normal vector $n$ and the unit tangent vector $T$, the unit binormal vector is defined by
\[ B(s_0) = T(s_0) \times n(s_0). \]

Implicit representation formulas

Following [40], the unit normal vector and unit tangent vectors can be expressed using an implicit representation of a surface. Given a function $\phi : \mathbb{R}^3 \rightarrow \mathbb{R}$ whose level set $\phi(x, y, z) = c$ represents a surface ($c \in \mathbb{R}$), the unit normal vector for a point $(x_0, y_0, z_0)$ that lies on the surface is defined as
\[ n(x_0, y_0, z_0) = \frac{\nabla \phi}{\|\nabla \phi\|}(x_0, y_0, z_0), \]
provided that $\nabla \phi(x_0, y_0, z_0) \neq (0, 0, 0)$. The equation of the tangent plane at the point $(x_0, y_0, z_0)$ is given by
\[ \nabla \phi(x_0, y_0, z_0) \cdot (x - x_0, y - y_0, z - z_0) = 0. \]

Similarly, for an implicit planar curve, the unit normal vector for a point $(x_0, y_0)$ on the curve such that $\nabla \phi(x_0, y_0) \neq (0, 0)$ is defined by
\[ n(x_0, y_0) = \frac{\nabla \phi}{\|\nabla \phi\|}(x_0, y_0). \]
The tangent line at the point $(x_0, y_0)$ is given by
\[ \nabla \phi(x_0, y_0) \cdot (x - x_0, y - y_0) = 0. \]

Finally, for functions $\phi_1$ and $\phi_2$ whose intersection defines implicitly a space curve, the unit tangent vector at a point $(x_0, y_0, z_0)$ is defined by
\[ T(x_0, y_0, z_0) = \frac{\nabla \phi_1 \times \nabla \phi_2}{\|\nabla \phi_1\|\|\nabla \phi_2\|}(x_0, y_0, z_0), \]
provided that $\nabla \phi_1, \nabla \phi_2 \neq (0,0,0)$. The normal vector is defined as
\[
n(x_0, y_0, z_0) = (\nabla \phi_1 \times \nabla \phi_2)^T J(\nabla \phi_1 \times \nabla \phi_2)
\]
where $J$ is the Jacobian matrix. The unit binormal vector can be found using Definition 3.

2.1.3 Curvature

The definition of the different types of curvatures is provided, according to [40]. For planar and space curves, the curvature is defined as follows.

**Definition 4.** The **curvature of a space curve** $x(s)$ at a point $s_0$ is defined as
\[
\kappa(s_0) = \frac{(x_s(s_0) \times x_{ss}(s_0)) \cdot n(s_0)}{\|x_s(s_0)\|},
\]
where $n$ is the normal of the space curve. The **curvature of a planar curve** $x(s)$ at a point $s_0$ is defined as
\[
\kappa(s_0) = \frac{\det(x_s(s_0)x_{ss}(s_0)^T)}{\|x_s(s_0)\|^3}.
\]

There are two types of curvatures for surfaces: the mean curvature and the Gaussian curvature. Before continuing with the definition of the curvatures, it is required to define the first and second fundamental forms for parametrized surfaces.

**Definition 5.** The **first fundamental form** of a surface $x(u,v)$ with unit normal vector $n$ is defined as
\[
I = \begin{pmatrix}
x_u \cdot x_u & x_u \cdot x_v \\
x_v \cdot x_u & x_v \cdot x_v
\end{pmatrix}.
\]

The **second fundamental form** is defined as
\[
II = \begin{pmatrix}
x_{uu} \cdot n & x_{uv} \cdot n \\
x_{uv} \cdot n & x_{vv} \cdot n
\end{pmatrix}.
\]

Taking into consideration that $x_u \cdot n = 0$, we get that
\[
x_{uu} \cdot n = -x_u \cdot n_u \quad \text{and} \quad x_{uv} \cdot n = -x_u \cdot n_v.
\]

Similarly, the corresponding equalities for $x_v \cdot n = 0$ can be obtained. Thus, the second fundamental form can be written as
\[
II = \begin{pmatrix}
x_u \cdot n_u & x_u \cdot n_v \\
x_v \cdot n_u & x_v \cdot n_v
\end{pmatrix}.
\]

Using Definition 5, the mean and the Gaussian curvatures of a surface can be defined as follows.
Definition 6. The Gaussian curvature of a surface $\mathbf{x}(u,v)$ with unit normal vector $\mathbf{n}$ and first and second fundamental forms $I$ and $II$ respectively is defined as

$$G = \frac{\det(II)}{\det(I)}.$$

The mean curvature is defined as

$$\kappa = \frac{\text{Trace}(I * II^*)}{\det(I)},$$

where $II^*$ is the adjoint matrix of the second fundamental form.

An alternative formula for the mean curvature uses the unit normal vector $\mathbf{n}$ of a surface $\Gamma$. That is

$$\kappa = \nabla_{\Gamma} \cdot \mathbf{n},$$

where $\nabla_{\Gamma}$ is a surface differential operator (see Section 2.3).

**Implicit representation formulas**

Using an implicit representation for curves and surfaces, the calculation of the curvature becomes simple. Following [40], for planar curves, given a function $\phi$ whose zero level set is the curve, the curvature is given by

$$\kappa = -\nabla \cdot \frac{\nabla \phi}{\|\nabla \phi\|} = -\nabla \cdot \mathbf{n}. \quad (2.2)$$

For a space curve, given two functions $\phi_1$ and $\phi_2$ whose intersection of their zero level set defines the curve, the curvature is given by

$$\kappa = \frac{\|((\nabla \phi_1 \times \nabla \phi_2)^T \ast \nabla(\nabla \phi_1 \times \nabla \phi_2)) \times (\nabla \phi_1 \times \nabla \phi_2)\|}{\|\nabla \phi_1 \times \nabla \phi_2\|^3}.$$

Finally, for a surface defined by the zero level set of a function $\phi$, the Gaussian curvature is given by

$$G = \frac{(\nabla \phi)^T \ast H^*(\phi) \ast \nabla \phi}{\|\nabla \phi\|^4},$$

and the mean curvature is given by

$$\kappa = \frac{(\nabla \phi)^T \ast H(\phi) \ast \nabla \phi - |\nabla \phi|^2 \text{Trace}(H(\phi))}{\|\nabla \phi\|^3},$$

where $H(\phi)$ is the Hessian matrix of the function $\phi$ and $H^*(\phi)$ is the adjoint of the Hessian matrix. The mean curvature of a surface can also be calculated using Equation (2.2).
The curvature properties can also be obtained using the curvature matrix [70] defined as

\[ K = \nabla n, \]

where \( n \) is the normal vector of a surface. The nontrivial eigenvalues of the matrix \( K \) are called principal curvatures \( \kappa_1 \) and \( \kappa_2 \). Using the principal curvatures, the mean and Gaussian curvature are given by

\[ \kappa = \kappa_1 + \kappa_2 \quad \text{and} \quad G = \kappa_1 \kappa_2. \]

In addition, the curvature matrix maps any vector in the normal direction to zero, i.e.

\[ K n = 0. \]

To illustrate, consider the example of an ellipse defined by the zero level set of the function

\[ \phi(x, y) = \frac{x^2}{a^2} + \frac{y^2}{b^2} - 1. \]

Its normal vector is given by

\[ n = \frac{\nabla \phi}{\|\nabla \phi\|} = \frac{(b^2 x, a^2 y)}{\sqrt{b^4 x^2 + a^4 y^2}}. \]

Then, the curvature matrix has the form

\[
K = \nabla n = \begin{pmatrix}
\frac{a^4 b^2 y^2}{(b^4 y^2 + a^4 y^2)^{3/2}} & -\frac{a^2 b^4 x y}{(b^4 y^2 + a^4 y^2)^{3/2}} \\
-a^4 b^2 x y & \frac{a^2 b^4 x^2}{(b^4 y^2 + a^4 y^2)^{3/2}}
\end{pmatrix}
\]

and satisfies \( K n = 0 \).

### 2.2 Closest point representation of surfaces

Following [61, 70, 60], we now define the closest point representation of a surface in the embedding space.

**Definition 7.** Let \( x \) be some point in the embedding space \( \mathbb{R}^d \). Then,

\[ \text{cp}_\Gamma(x) = \min_{z \in \Gamma} \|x - z\| \]

is the closest point of \( x \) to the surface \( \Gamma \).
The closest point representation is a different approach to represent a surface. Using \( cp_\Gamma \), given a point \( x \) on the embedding space \( \mathbb{R}^d \), the signed distance function can be written as

\[
\text{dist}(x) = \begin{cases} 
\|x - cp_\Gamma(x)\|, & x \text{ in the interior of } \Gamma, \\
0, & x \in \Gamma, \\
-\|x - cp_\Gamma(x)\|, & x \text{ outside } \Gamma.
\end{cases}
\] (2.3)

By differentiating this formula for \( x \notin \Gamma \), an expression for the unit normal vector is obtained as

\[
n(x) = \nabla \text{dist}(x) = \pm \frac{x - cp_\Gamma(x)}{\|x - cp_\Gamma(x)\|}.
\]

The formula of the unit normal vector provides an alternative expression for the closest point function using the signed distance function. That is

\[
\text{cp}_\Gamma(x) = x - \text{dist}(x)\nabla \text{dist}(x),
\]

which is well defined for all \( x \), since for \( x \in \Gamma \) we get \( \text{cp}_\Gamma(x) = x \). By taking the divergence of the closest point representation, we get

\[
\nabla \cdot \text{cp}_\Gamma(x) = d - \nabla \text{dist}(x) \cdot \nabla \text{dist}(x) - \text{dist}(x) \nabla^2 \text{dist}(x)
\]

\[
= d - 1 - \text{dist}(x) \nabla^2 \text{dist}(x),
\]

using \( \nabla \text{dist}(x) = n(x) \) and \( \|\nabla \text{dist}(x)\| = 1 \), where \( d \) is the dimension of the embedding space \( \mathbb{R}^d \). Taking the gradient of this expression, we get

\[
\nabla^2 \text{cp}_\Gamma(x) = -\nabla \text{dist}(x) \nabla^2 \text{dist}(x) - \text{dist}(x) \nabla (\nabla^2 \text{dist}(x))
\]

\[
= \kappa(x) n(x) - \text{dist}(x) \nabla (\nabla^2 \text{dist}(x)),
\]

using \( \kappa(x) = -\nabla^2 \text{dist}(x) \), where \( \kappa(x) \) is the mean curvature. Therefore, an expression for the mean curvature using the closest point representation takes the form

\[
\kappa(x) = (\nabla^2 \text{cp}_\Gamma(x)) \cdot n(x)
\]

on the surface \( \Gamma \) [71].

### 2.3 Surface differential operators

Surface differential operators are the differential operators that are defined on surfaces. For a surface \( \Gamma \) with unit tangent vectors \( T_1, T_2 \), the surface gradient of a function \( f \) is given
by
\[ \nabla_{\Gamma} f = \frac{\partial f}{\partial \mathbf{T}_1} \mathbf{T}_1 + \frac{\partial f}{\partial \mathbf{T}_2} \mathbf{T}_2. \]

For a surface \( \Gamma \) parametrized as \( \mathbf{x}(u,v) \), surface differential operators can be defined. Following [21, 47], if \( I = [g_{ij}]_{2\times2} \) is the first fundamental form (see Definition 5) and \( I^{-1} = [g^{ij}]_{2\times2} \) is its inverse, then the surface gradient of a function \( f \) is defined as
\[ \nabla_{\Gamma} f = g^{1,1} f_u \mathbf{x}_u + g^{1,2} f_u \mathbf{x}_v + g^{2,1} f_v \mathbf{x}_u + g^{2,2} f_v \mathbf{x}_v. \]

For a velocity field defined as \( V = v_1 \mathbf{x}_u + v_2 \mathbf{x}_v \) the surface divergence is defined as
\[ \nabla_{\Gamma} \cdot V = \frac{1}{\sqrt{\det(I)}} \left( \frac{\partial}{\partial u} \left( \sqrt{\det(I)} v_1 \right) + \frac{\partial}{\partial v} \left( \sqrt{\det(I)} v_2 \right) \right). \]

Combining the surface gradient and the surface divergence, the surface Laplacian or the Laplace-Beltrami operator for a function \( f \) can be defined as
\[ \Delta_{\Gamma} f = \frac{1}{\sqrt{\det(I)}} \left( \frac{\partial}{\partial u} \left( \sqrt{\det(I)} (g^{1,1} f_u + g^{2,2} f_v) \right) + \frac{\partial}{\partial v} \left( \sqrt{\det(I)} (g^{1,2} f_u + g^{2,2} f_v) \right) \right). \]

For example, consider the unit sphere parametrized as
\[ \mathbf{x}(u,v) = (\cos(u) \cos(v), \sin(u) \cos(v), \sin(v)). \]

The partial derivatives of \( \mathbf{x} \) in terms of \( u, v \) are
\[ \begin{pmatrix} \mathbf{x}_u \\ \mathbf{x}_v \end{pmatrix} = \begin{pmatrix} -\sin(u) \cos(v) & \cos(u) \cos(v) & 0 \\ -\cos(u) \sin(v) & -\sin(u) \sin(v) & \cos(v) \end{pmatrix}. \]

Using Definition 5, the first fundamental form can be calculated
\[ I = \begin{pmatrix} \cos^2(v) & 0 \\ 0 & 1 \end{pmatrix}, \]

with its determinant equal to
\[ \det(I) = \cos^2(v). \]

The inverse of the matrix \( I \) is
\[ I^{-1} = \begin{pmatrix} \frac{1}{\cos^2(v)} & 0 \\ 0 & 1 \end{pmatrix}. \]
The Laplace-Beltrami operator for a function $f$ on the unit sphere is expressed as

$$\Delta \Gamma f = \frac{1}{\cos^2(v)} f_{uu} + \frac{1}{\cos(v)} \frac{\partial}{\partial v} \left( \cos(v) f_v \right).$$

For a surface $\Gamma$, an alternative formulation of the surface tangential derivative can be found. Assume that $\tilde{f}$ is a smooth extension of the function $f$ in a neighborhood of the surface, such that $\tilde{f}|_\Gamma = f$. If $n$ is the unit normal vector of the surface and $T_1, T_2$ are the unit tangent vectors of the surface, then using the orthonormal basis $\{n, T_1, T_2\}$ for $\mathbb{R}^3$, the gradient of the function $\tilde{f}$ can be written as follows [68]

$$\nabla \tilde{f} = \frac{\partial \tilde{f}}{\partial n} n + \frac{\partial \tilde{f}}{\partial T_1} T_1 + \frac{\partial \tilde{f}}{\partial T_2} T_2.$$

By eliminating the normal component of the gradient of the function $\tilde{f}$, we get an alternative formulation of the surface gradient

$$\nabla \Gamma f = \nabla \tilde{f} - (\nabla \tilde{f} \cdot n) n,$$

on the surface $\Gamma$.

In the case of curves with unit tangent vector $T$, following the same process as above, the surface gradient of a function $f$ is defined as

$$\nabla \Gamma f = \frac{\partial f}{\partial T} T.$$

If the curve is parametrized as $x(s)$, then the surface gradient of a function $f$ is given by

$$\nabla \Gamma f = \frac{1}{\|x_s\|} \frac{\partial f}{\partial s} T.$$

### 2.4 Radial basis functions (RBF) approximation

RBF approximation is a powerful tool for approximating smooth functions on a variety of geometries. Following [27, 32], given an RBF $\phi(r)$ (see Table 2.1 for some RBF options) and a set of $m$ scattered points $\{z_j\}$ called RBF centers, the RBF interpolant has the form

$$u(x) \approx \sum_{j=1}^{m} \lambda_j \phi(\|x - z_j\|)$$

with coefficients $\lambda_j$. For given $n$ data sites $\{x_i\}$ and a function $u(x)$, this interpolant can be written in a matrix form as

$$u(x) \approx A\lambda$$

where $A = [\phi(\|x_i - z_j\|)]_{n \times m}$ and $\lambda = [\lambda_j]_{m \times 1}$ is a vector of coefficients.
Table 2.1: Definition of some commonly used RBFs.

<table>
<thead>
<tr>
<th>Name of RBF</th>
<th>Abbreviation</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Smooth RBFs</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Gaussian</td>
<td>GA</td>
<td>$\phi(r) = e^{-(\epsilon r)^2}$</td>
</tr>
<tr>
<td>Multiquadratic</td>
<td>MQ</td>
<td>$\phi(r) = \sqrt{1 + (\epsilon r)^2}$</td>
</tr>
<tr>
<td>Inverse multiquadratic</td>
<td>IMQ</td>
<td>$\phi(r) = \frac{1}{\sqrt{1 + (\epsilon r)^2}}$</td>
</tr>
<tr>
<td>Inverse quadratic</td>
<td>IQ</td>
<td>$\phi(r) = \frac{1}{1 + (\epsilon r)^2}$</td>
</tr>
<tr>
<td>Piecewise smooth RBFs</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cubic</td>
<td>CU</td>
<td>$\phi(r) =</td>
</tr>
<tr>
<td>Thin plate spline</td>
<td>TPS</td>
<td>$\phi(r) = r^2 \ln</td>
</tr>
</tbody>
</table>

RBFs can be used to approximate the derivatives of a function. If $L$ is a differential operator, then the quantity $Lu$ can be approximated as

$$Lu(x) \approx \sum_{j=1}^{m} \lambda_j L\phi(\|x - z_j\|).$$  \hspace{1cm} (2.6)

For a set of data sites $\{x_i\}$, the matrix form of Equation (2.6) is

$$Lu(x) \approx B\lambda$$  \hspace{1cm} (2.7)

where $B = [L\phi(\|x_i - z_j\|)]_{n \times m}$ and the coefficients $\lambda$ are found from Equation (2.5).

By combining Equations (2.5) and (2.7), the quantity $Lu(x)$ can be approximated in terms of $u$ as

$$Lu(x) \approx BA^{-1}u,$$  \hspace{1cm} (2.8)

where $A^{-1}$ corresponds to the left inverse of the matrix $A$ and $u = [u(x_i)]_{n \times 1}$. This expression provides a finite difference approximation (RBF-FD) for the differential operator $L$. Specifically, the finite difference weights are given by

$$L \approx BA^{-1}.$$  \hspace{1cm} (2.9)

A direct calculation of the matrix $A$ using its definition is called RBF-Direct. For small shape parameters $\epsilon$, the radial basis functions become nearly flat resulting in an ill-conditioned basis to expand in. Consequently, matrix $A$ becomes nearly singular, [17, 37]. Stable approximations of the matrix $A$ are obtained using the Contour-Padé method [36].

14
the RBF-QR method\textsuperscript{1} [35, 33, 48] and the RBF-GA method\textsuperscript{2} [34]. In this thesis, we use Gaussian RBFs and the stable RBF-GA method.

Some compact RBF-FD stencils can be found in [31] for applications in geosciences. Details on RBF-FD stencils that use \( n - 1 \) nearest neighbors are found in [29].

\subsection{The RBF-GA method}

The RBF-GA algorithm is a method which provides a stable calculation of the system (2.5) using a well-conditioned basis that spans the space defined by the ill-conditioned nearly flat (for a small shape parameter \( \epsilon \)) Gaussian RBFs.

Following [34], consider the Gaussian RBF \( \phi(r) = e^{-(r)^2} \) centered at the RBF centers \( \{z_j\}, \ j = 1, \ldots, m \). For illustration purposes, we consider the two dimensional case \( x = (x_1, y_1) \). The extension in three dimensions follows the same steps.

For \( z_j = (z_1, z_2) \), the Gaussian RBF can be written as

\[ \psi^{(1)}_j(x) = e^{-\epsilon^2((x_1-z_1)^2+(y_1-z_2)^2)} \]

\[ = e^{-\epsilon^2(x_1^2+z_1^2)} \cdot e^{-\epsilon^2(x_1^2+y_1^2)} \cdot e^{2\epsilon^2(x_1z_1+y_1z_2)}. \]

Ignoring the constant factor \( e^{-\epsilon^2(x_1^2+z_2^2)} \) and by using a Taylor expansion of \( e^{2\epsilon^2(x_1z_1+y_1z_2)} \) of order \( n - 1 \), an equivalent basis function takes the form

\[ \psi^{(2)}_j(x) = e^{-\epsilon^2(x_1^2+y_1^2)} \cdot e^{2\epsilon^2(x_1z_1+y_1z_2)} \]

\[ = e^{-\epsilon^2(x_1^2+y_1^2)} \cdot \left( \sum_{k=0}^{n-1} \frac{1}{k!} (2\epsilon^2(x_1z_1+y_1z_2))^k + G_n(w) \right), \]

where

\[ G_n(w) = e^w - \sum_{k=0}^{n-1} \frac{w^k}{k!} \]

\[ = \frac{e^w}{(n-1)!} \int_0^w e^{-t}t^{k-1} dt, \]

and \( w = 2\epsilon^2(x_1z_1 + y_1z_2) \).

To eliminate the linear dependence of the first terms in the Taylor expansion for small \( \epsilon \), a new basis is constructed that eliminates these terms analytically. Using \textit{null} to denote

\footnote{For a MATLAB implementation, see http://www.it.uu.se/research/scientific_computing/software/rbf_qr}

\footnote{A MATLAB implementation is available online at http://www.mathworks.com/matlabcentral/fileexchange/48198-rbf-ga-differentiation-weights}
an orthogonal basis for the null-space of a matrix, the new basis takes the form

\[ \psi_1(x) = e^{-\epsilon^2(x_1^2+y_1^2)} \frac{1}{\epsilon^2} B_0 G_0 (2\epsilon^2 x \cdot z_1), \]

with \( B_0 = 1 \),

\[
\begin{pmatrix}
\psi_2(x) \\
\psi_3(x)
\end{pmatrix} = e^{-\epsilon^2(x_1^2+y_1^2)} \frac{1}{\epsilon^2} B_1 \begin{pmatrix}
G_1(2\epsilon^2 x \cdot z_1) \\
G_1(2\epsilon^2 x \cdot z_2) \\
G_1(2\epsilon^2 x \cdot z_3)
\end{pmatrix},
\]

with \( B_1 = [\text{null}([1 1 1])]^T_{2 \times 3} \),

\[
\begin{pmatrix}
\psi_4(x) \\
\psi_5(x) \\
\psi_6(x)
\end{pmatrix} = e^{-\epsilon^2(x_1^2+y_1^2)} \frac{1}{\epsilon^4} B_2 \begin{pmatrix}
G_2(2\epsilon^2 x \cdot z_1) \\
G_2(2\epsilon^2 x \cdot z_2) \\
G_2(2\epsilon^2 x \cdot z_3) \\
G_2(2\epsilon^2 x \cdot z_4) \\
G_2(2\epsilon^2 x \cdot z_5) \\
G_2(2\epsilon^2 x \cdot z_6)
\end{pmatrix},
\]

with \( B_2 = \left[ \text{null} \begin{pmatrix}
1 & 1 & 1 & 1 & 1 & 1 \\
x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \\
y_1 & y_2 & y_3 & y_4 & y_5 & y_6
\end{pmatrix} \right]_3^{T \times 6} \), etc.

Reaching the \( k \)-th stage of the algorithm in \( d \)-dimensions, the \((d+k-1)\) new basis functions are calculated using a matrix \( B_k \) of size \((d+k-1) \times (d+k)\) which contains an orthogonal basis of the null-space of a \((d+k-1) \times (d+k)\) polynomial matrix. Let \( P_k \) be the polynomial matrix of order \( k \) for all nodes, i.e.

\[
P_1 = \begin{pmatrix}
1 & 1 & \cdots & 1 \\
x_1 & x_2 & \cdots & x_n \\
y_1 & y_2 & \cdots & y_n
\end{pmatrix},
\]

and \( P_{k,j} \) the submatrix \( P_k \) with \( j \)-columns. Then, the number of the new basis functions at the \( k \)-th stage is given by \( r_k - r_{k-1} \), where \( r_k \) is the rank of the matrix \( P_k \), using the null-space of the \( P_{k-1,r_k} \) matrix. The process is repeated until \( n \) basis functions \( \psi_j \) are found.

### 2.5 Time stepping methods

Consider a system of ordinary differential equations (ODEs) of the form

\[
\dot{U}(t) = F(U(t)) \tag{2.10}
\]
with some initial condition $U(t_0) = U^0$, which arises from a semi-discretization in space of a partial differential equation. In order to solve this ODE system numerically, there are several finite difference discretization techniques available. The finite difference schemes can be categorized as explicit, implicit and implicit-explicit (IMEX).

2.5.1 Explicit time stepping methods

Explicit time stepping methods use information from previous time steps to evolve the numerical solution in time. The most popular explicit methods are the Runge-Kutta methods [14]. For an $s$-stage Runge-Kutta scheme, given the approximation $U^n$ at the $n$-th time step, the solution at $n + 1$ can be approximated via

$$U^{n+1} = U^n + \Delta t \sum_{i=1}^{s} b_i k_i$$  \hspace{1cm} (2.11)

where

$$k_1 = F(U^n),$$
$$k_2 = F(U^n + \Delta t (a_{2,1} k_1)),$$
$$\vdots$$
$$k_s = F(U^n + \Delta t (a_{s,1} k_1 + a_{s,2} k_2 + \ldots + a_{s,s-1} k_{s-1})).$$

and $a_{i,k}$, $1 \leq k < i \leq s$, $b_i$ $(i = 1, 2, \ldots, s)$ and $c_i = \sum_{k=1}^{i-1} a_{i,k}$ $(i = 2, 3, \ldots, s)$ are constants. This formulation can be written in the Butcher Tableau form

$$
\begin{array}{c|cccc}
0 & & & & \\
c_2 & a_{2,1} & & & \\
c_3 & a_{3,1} & a_{3,2} & & \\
\vdots & \vdots & \vdots & \ddots & \\
c_s & a_{s,1} & a_{s,2} & \cdots & a_{s,s-1} & \\
\hline & b_1 & b_2 & \cdots & b_{s-1} & b_s
\end{array}
$$

Some popular Runge-Kutta schemes include the one-stage Runge Kutta scheme or forward Euler

$$U^{n+1} = U^n + \Delta t F(U^n)$$

which is first-order accurate, and the three-stages third-order Runge-Kutta scheme

$$\bar{U}^{(1)} = U^n + \Delta t F(U^n),$$
$$\bar{U}^{(2)} = \frac{3}{4} U^n + \frac{1}{4} \bar{U}^{(1)} + \frac{1}{4} \Delta t F(\bar{U}^{(1)}),$$
$$U^{n+1} = \frac{1}{3} U^n + \frac{2}{3} \bar{U}^{(2)} + \frac{2}{3} \Delta t F(\bar{U}^{(2)}),$$

17
which is an optimal total variation diminishing Runge-Kutta method (TVD-RK) [41].

2.5.2 Implicit time stepping methods

Implicit time stepping methods use information from the current and previous time steps to evolve the solution in time.

A category of implicit time stepping methods is the backward differentiation formulas (BDF). For the initial value problem (2.10), the general formula for an s-step BDF [5] takes the form

\[ \sum_{i=0}^{s} \alpha_i^s U^{n+i} = \Delta t \beta^s F(U^{n+s}), \]  

where \( \alpha_i^s \) and \( \beta^s \) are constants.

In this thesis, the BDFs that are used are the BDF1 or the implicit Euler method

\[ U^{n+1} = U^n + \Delta t F(U^{n+1}), \]

which is first-order accurate, and the BDF2 method

\[ U^{n+1} = \frac{4}{3} U^n - \frac{1}{3} U^{n-1} + \frac{2}{3} \Delta t F(U^{n+1}), \quad n \geq 1, \]

which is second-order accurate. Note that in BDF2, an extra first step is required to calculate \( U^1 \). A step of the implicit Euler method can be applied before the use of BDF2 to obtain \( U^1 \).

2.5.3 Implicit-explicit (IMEX) time stepping methods

In some nonlinear partial differential equations, the use of fully implicit time stepping methods is computationally inefficient. Implicit-explicit methods treat the stiff term \( g \) implicitly and the non-stiff term \( f \) explicitly. Consider an ODE system of the form

\[ \dot{U}(t) = f(U(t)) + g(U(t)) \]

arising from the semi-discretization of a PDE, where \( f \) consists of the non-stiff (usually nonlinear) terms and \( g \) contains the stiff terms.

Using IMEX methods:

1. large time step-sizes may be possible, as the stiff terms that introduce time step restrictions are treated implicitly.

2. the complications that arise when nonlinear terms are treated implicitly can be avoided, as the non-stiff terms are treated explicitly.
Following [7], two IMEX methods are used in this thesis. First, the IMEX Euler method applies backward Euler to the stiff terms and forward Euler to the non-stiff terms:

\[ U^{n+1} - U^n = \Delta t (f(U^n) + g(U^{n+1})). \]

The IMEX Euler method is a first-order method. A second-order IMEX method called SBDF2 is given by

\[ \frac{1}{2\Delta t}(3U^{n+1} - 4U^n + U^{n-1}) = 2f(U^n) - f(U^{n-1}) + g(U^{n+1}), \quad n \geq 1. \]

An initial step of the IMEX Euler can be used to calculate \( U^1 \). Other IMEX methods are also available, including the IMEX Runge-Kutta methods [6].
Chapter 3

Problem statement

In this chapter, a review on the definition of the surface integration is presented. Following this, using the surface integration formulae, the PDE model on moving surfaces is derived for the cases of a diffusive flux and a Cahn-Hilliard flux.

3.1 Surface integration

In this section, we review the definition of the surface integral for parametrized surfaces and we prove two surface integration formulas.

The surface integral of a scalar function $f$ over a surface $\Gamma$ parametrized as $x(u,v)$, $(u,v) \in \Omega \subset \mathbb{R}^2$, is given by

$$\int_\Gamma f \, dA = \int_\Omega f(x(u,v)) \|x_u \times x_v\| \, du \, dv$$

or equivalently

$$\int_\Gamma f \, dA = \int_\Omega f(x(u,v)) \sqrt{\det(I)} \, du \, dv$$

using the Lagrange’s identity $\|a \times b\|^2 = \|a\|^2\|b\|^2 - (a \cdot b)^2$ and the first fundamental form $I$ (see Definition 5). A transportation formula and a surface integration by parts are essential for the derivation of the PDE model considered throughout this thesis. The following two Lemmas and their proofs appear in [18, 21].

**Lemma 1.** Let $M(t)$ be an evolving surface with velocity $v = Vn + T$, where $V$ in the velocity component in the normal direction, $n$ is the unit normal vector and $T$ is the tangential velocity. Suppose that $u$ is a function defined on $M(t)$ such that $\partial u/\partial t$ exists. Assume that the surface integral of $u$ and $\partial u/\partial t$ over $M(t)$ exists for all times $t \geq 0$. Then,

$$\frac{d}{dt} \int_{M(t)} u \, dA = \int_{M(t)} \frac{D}{Dt} u + u \nabla M(t) \cdot v \, dA,$$
where \( Du/Dt \) is the total derivative of the function \( u \) and \( \nabla_{M(t)} \cdot v \) is the surface divergence of the velocity \( v \) defined in Section 2.3.

Proof. Let \( \Omega \subset \mathbb{R}^n \) be an open set and let \( x(\theta, t), \theta \in \Omega \), be a local regular parametrization that maps \( \Omega \) to an open portion of the surface \( U \subset M(t) \) for a time \( t \geq 0 \) and which evolves according to \( x_t = v(x(\theta, t), t) \). The first fundamental form is given by \( I = [g_{i,j}] = [x_{\theta_i} \cdot x_{\theta_j}] \) and \( I^{-1} = [g^{ij}] \) is its inverse.

Define
\[
F(\theta, t) = u(x(\theta, t), t) \quad \text{and} \quad V = v(x(\theta, t), t).
\]

Then, using the determinant differentiation formula [21],
\[
\frac{\partial}{\partial t} \sqrt{\det(I)} = \sqrt{\det(I)} \sum_{i,j=1}^{n} g^{ij} x_{\theta_i} \cdot V_{\theta_j},
\]
we have
\[
\frac{d}{dt} \int_U u \, dA = \frac{d}{dt} \int_\Omega F \sqrt{\det(I)} \, d\theta
\]
\[
= \int_\Omega \frac{\partial F}{\partial t} \sqrt{\det(I)} + F \frac{\partial}{\partial t} \left( \sqrt{\det(I)} \right) \, d\theta
\]
\[
= \int_\Omega \left( \frac{\partial u}{\partial t}(x, t) + \nabla u(x, t) \cdot x_t \right) \sqrt{\det(I)}
\]
\[
+ F(x, t) \sqrt{\det(I)} \sum_{i,j=1}^{n} g^{ij} x_{\theta_i} \cdot V_{\theta_j} \, d\theta
\]
\[
= \int_U D \frac{D}{D t} u + u \nabla_{M(t)} \cdot v \, dA,
\]
where in the last equality we used
\[
(\nabla_{M(t)} \cdot v)(x, t) = \sum_{i,j=1}^{n} g^{ij} x_{\theta_i} \cdot V_{\theta_j}.
\]

Lemma 2. Let \( M = M(t) \) be a hypersurface in \( \mathbb{R}^{n+1} \) at some time \( t \) with smooth boundary \( \partial M \) and let \( u \in C^1(M) \). Then,
\[
\int_M \nabla_{M(t)} u \, dA = \int_M u \kappa n \, dA + \int_{\partial M} u \mu \, dS
\]
where \( n \) is the normal of the surface \( M \), \( \mu \) is the co-normal vector to the boundary \( \partial M \) tangent to \( M \), and \( \kappa \) is the mean curvature of the surface (see Figure 3.1). In the case that \( \partial M = \emptyset \), the last term on the right hand side vanishes.

21
**Proof.** We extend the function $u$ to a tubular neighborhood $\Omega_\epsilon$ with radius $\epsilon$ from the surface $M$ using the closest point representation $c_{PM}$

$$\tilde{u}(x) = u(c_{PM}(x)), \quad x \in \Omega_\epsilon.$$ 

Then, using the chain rule, we get

$$\frac{\partial \tilde{u}}{\partial x_j} = \sum_{k=1}^{n+1} \frac{\partial u(c_{PM}(x))}{\partial cp_k^M(x)} \frac{\partial cp_k^M(x)}{\partial x_j}.$$ 

Let $dist(x)$ be a signed distance function for the surface $M$ and $n(x)$ the unit normal vector. Then, we compute

$$a_{kj} = \frac{\partial cp_k^M(x)}{\partial x_j} = \frac{\partial}{\partial x_j} (x_k - dist(x)n_k) = \delta_{jk} - n_j(x)n_k(x) - dist(x)K_{jk}(x),$$

where $K(x)$ is the curvature matrix. Note that the matrix $A = [a_{kj}]$ maps any vector to a tangent vector, i.e.

$$An(x) = n(x) - (n(x)n^T(x))n(x) - dist(x)K(x)n(x) = 0.$$ 

Thus, we obtain

$$\nabla \tilde{u}(x) = (I - dist(x)K(x))\nabla Mu(c_{PM}(x)),$$

where $I$ is the identity matrix, as the quantity $n(x)n(x)^T\nabla Mu(c_{PM}(x)) = 0$ from orthogonality. By applying the Gauss theorem to $\tilde{u}$ on $\Omega_\epsilon$, we get

$$\int_{\Omega_\epsilon} \nabla \tilde{u} \, dx = \int_{\partial \Omega_\epsilon} \tilde{u} n_{\partial \Omega_\epsilon} \, dS.$$ 

Note that the boundary $\partial \Omega_\epsilon = M(\epsilon) \cup M(-\epsilon) \cup N(\epsilon)$ where $M(\epsilon) = \{x + \epsilon n(x) | x \in M\}$ and $N(\epsilon) = \{x + r n(x) | x \in \partial M, r \in [-\epsilon, \epsilon]\}$, as shown in Figure 3.1. Using the formula for $\nabla \tilde{u}$, we get

$$\frac{1}{2\epsilon} \int_{\Omega_\epsilon} (I - d(x)K(x))\nabla Mu(c_{PM}(x)) \, dx = \frac{1}{2\epsilon} \left( \int_{M(\epsilon)} \tilde{u}(x)n(x) \, dS - \int_{M(-\epsilon)} \tilde{u}(x)n(x) \, dS \right. \right.$$

$$+ \int_{N(\epsilon)} \tilde{u}(x)\mu(x) \, dS \left. \right),$$

where $n$ is the unit normal vector of the surface $M$ extended to be normal on the boundaries $M(\epsilon)$ and $M(-\epsilon)$ and $\mu$ is the unit co-normal vector of the boundary $N(\epsilon)$. Taking the limit $\epsilon \to 0$, we get

$$\lim_{\epsilon \to 0} \frac{1}{2\epsilon} \int_{\Omega_\epsilon} (I - dist(x)K(x))\nabla Mu(c_{PM}(x)) \, dx = \int_{M} \nabla Mu(x) \, dA(x).$$
 Figure 3.1: An illustration of a surface $M$ and the tubular neighborhood $\Omega_\epsilon$.

For the first two terms of the right hand side, we get

$$\frac{d}{d\epsilon} \int_{M(\epsilon)} \bar{u}(x)n(x) \, dA(x) \bigg|_{\epsilon=0} = \int_{M} u(x)\kappa(x)n(x) \, dA(x),$$

where $\kappa(x)$ is the mean curvature of the surface at $x$. In this calculation, we used Lemma 1 with $v = n$ and $D\bar{u}/D\epsilon = 0$. Finally, the last term becomes

$$\lim_{\epsilon \to 0} \frac{1}{2\epsilon} \int_{N(\epsilon)} \bar{u}(x)\mu(x) \, dA(x) = \int_{\partial M} u(x)\mu(x) \, dS(x).$$

In case of a closed surface, $\partial M = \emptyset$ and the boundary of the tubular neighborhood takes the form $\partial \Omega_\epsilon = M(\epsilon) \cup M(-\epsilon)$. The proof follows the same steps as above.

3.2 PDE model derivation

In this section, we state and derive the PDE on moving surfaces that we are interested in. Following [75, 21], let $u$ be the density of a scalar quantity that is conserved on a surface $\Gamma(t)$. Then, for an arbitrary part of the surface $M(t)$, the conservation law takes the form

$$\frac{d}{dt} \int_{M(t)} u \, dA = \int_{\partial M(t)} q \cdot \mu \, dS,$$

where $\mu$ is the co-normal vector of $M(t)$ pointing outwards, tangent to $\Gamma(t)$, and $q$ is the surface flux. This conservation law considers the flux only in the co-normal direction assuming that there is no leak in the normal direction to the surface.
Using Lemma 1, the left hand side of Equation (3.1) takes the form
\[
\frac{d}{dt} \int_{M(t)} u \, dA = \int_{M(t)} \frac{D}{Dt} u + u \nabla \Gamma \cdot \mathbf{v} \, dA,
\] (3.2)
where \( Du/Dt \) is the total derivative of \( u \). Note that \( \nabla \Gamma \equiv \nabla M \) in the interior of \( M(t) \).

Using Lemma 2, the right hand side of Equation (3.1) takes the form
\[
\int_{\partial M(t)} \mathbf{q} \cdot \mathbf{n} \, dS = \int_{M(t)} \nabla M(t) \cdot \mathbf{q} \, dA + \int_{M(t)} \mathbf{q} \cdot \mathbf{n} \kappa \, dA = \int_{M(t)} \nabla M(t) \cdot \mathbf{q} \, dA,
\] (3.3)
using the fact that there is no flux in the normal direction of the surface, i.e. \( \mathbf{q} \cdot \mathbf{n} = 0 \).

Combining Equations (3.2) and (3.3), we get the conservation law
\[
\int_{M(t)} \frac{D}{Dt} u + u \nabla M(t) \cdot \mathbf{v} + \nabla M(t) \cdot \mathbf{q} \, dA = 0,
\]
which holds for any arbitrary part \( M(t) \) of the surface \( \Gamma(t) \), thus leading to a pointwise conservation law
\[
\frac{D}{Dt} u + u \nabla M(t) \cdot \mathbf{v} + \nabla M(t) \cdot \mathbf{q} = 0.
\] (3.4)

An alternative formulation of Equation (3.4) is available. Assume that \( \bar{u} \) is a smooth extension of the function \( u \) in a neighborhood \( \Omega_\epsilon \) of the surface, such that \( \bar{u} \big|_{M(t)} = u \). By splitting the velocity into normal and tangential components \( \mathbf{v} = V \mathbf{n} + \mathbf{T} \) and extending it in the domain \( \Omega_\epsilon \), the first two terms of the conservation law take the form
\[
\int_{M(t)} \frac{D}{Dt} u + u \nabla M(t) \cdot \mathbf{v} \, dA = \lim_{\epsilon \to 0} \int_{\Omega_\epsilon} \bar{u}_t + \mathbf{v} \cdot \nabla \bar{u} + \bar{u} \nabla M(t) \cdot \mathbf{v} \, d\mathbf{x}
\]
\[
= \lim_{\epsilon \to 0} \int_{\Omega_\epsilon} \bar{u}_t + V \mathbf{n} \cdot \nabla \bar{u} + \mathbf{T} \cdot \nabla \bar{u}
\]
\[
+ \bar{u} V \nabla M(t) \cdot \mathbf{n} + \bar{u} \nabla M(t) \cdot \mathbf{T} \, d\mathbf{x}
\]
\[
= \lim_{\epsilon \to 0} \int_{\Omega_\epsilon} \bar{u}_t + V \frac{\partial \bar{u}}{\partial \mathbf{n}} - \bar{u} V \kappa + \mathbf{T} \cdot \nabla M(t) \bar{u} + \bar{u} \nabla M(t) \cdot \mathbf{T} \, d\mathbf{x}
\]
\[
= \lim_{\epsilon \to 0} \int_{\Omega_\epsilon} \bar{u}_t + V \frac{\partial \bar{u}}{\partial \mathbf{n}} - \bar{u} V \kappa + \nabla M(t) \cdot (\bar{u} \mathbf{T}) \, d\mathbf{x}
\]
\[
= \int_{M(t)} u_t + V \frac{\partial u}{\partial \mathbf{n}} - V u \kappa + \nabla M(t) \cdot (u \mathbf{T}) \, dA.
\]

In the third equality, we used the definition of the mean curvature \( \kappa \) and by orthogonality the property
\[
\mathbf{T} \cdot \nabla \bar{u} = \mathbf{T} \cdot (\nabla M(t) \bar{u} + (\nabla \bar{u} \cdot \mathbf{n}) \mathbf{n}) = \mathbf{T} \cdot \nabla M(t) \bar{u}.
\]
In the fourth equality, we used the surface integration by parts

\[ \nabla_{M(t)} \cdot (\bar{u} T) = \bar{u} \nabla_{M(t)} \cdot T + T \cdot \nabla_{M(t)} \bar{u}. \]

Therefore, the pointwise alternative formulation of the conservation law takes the form

\[ u_t + V \frac{\partial u}{\partial n} - V \kappa u + \nabla \Gamma \cdot (u T) - \nabla \Gamma \cdot q = 0 \quad (3.5) \]

where \( \kappa \) is the mean curvature of the surface. There are two choices of flux that we consider in this thesis. In the first case, a diffusive flux is chosen, i.e.

\[ q = -D \nabla_{M(t)} w, \]

where \( w \) is a variable dependent on \( u \) and \( D > 0 \) is a constant diffusivity parameter. Setting \( w = u \) leads to the advection-diffusion equation \[21\]

\[ \frac{D}{Dt} u + u \nabla_{M(t)} \cdot v - \nabla_{M(t)} \cdot D \nabla_{M(t)} u = 0. \quad (3.6) \]

By choosing

\[ w = -\mu \Delta \Gamma u + \psi'(u) \]

where \( \psi(u) \) is a double well potential function and \( \mu > 0 \) is the chemical potential, we get an advection-diffusion equation with a fourth-order elliptic operator called Cahn-Hilliard equation on moving surfaces \[21\]. That is:

\[ \frac{D}{Dt} u + u \nabla_{M(t)} \cdot v - \nabla_{M(t)} \cdot D \nabla_{M(t)} (-\mu \Delta \Gamma u + \psi'(u)) = 0. \quad (3.7) \]

There are PDE models that we are interested in applying our method, i.e. the reaction-advection-diffusion PDE on moving surfaces.
Chapter 4

Numerical methods

In this chapter, we review the numerical methods that we use to solve PDEs on moving surfaces. The closest point method [70] is a method for solving PDEs on stationary surfaces. The grid based particle method [51] is used to track the motion of a surface. The combination of these two methods leads to our simple coupled methods for solving PDEs on moving surfaces.

4.1 The closest point method

The closest point method (CPM) [70] is a numerical method for approximating PDEs on stationary surfaces. The method is applied to the solution of rather general PDEs on surfaces, including the calculation of the solution of eigenvalue problems on surfaces [55] and the approximation of the solution of level set equations on surfaces [57].

In the original formulation [70], a uniform finite difference grid is formed around the surface. Grid nodes store the coordinates of the closest point in Euclidean distance to the surface given by a closest point representation \( \text{cp}_\Gamma \) of the surface, as defined in Section 2.2. If not given, a closest point representation of a simple surface can be calculated analytically. For surfaces where a parametrization is possible, the closest point representation can be calculated by standard optimization techniques. Techniques are also available for triangulated surfaces [70].

The evolution of the surface PDE is carried out using an embedding PDE. To form the embedding PDE, derivatives intrinsic to the surface in the original PDE are replaced with a combination of closest point operators and derivatives expressed in standard Cartesian coordinates in the embedding space. Two principles are used to define the embedding PDE. The first of these relies on the fact that if surface values are extended off the surface in the normal direction, then, at the surface, the standard gradient in the embedding space agrees with the gradient intrinsic to the surface. This fundamental idea is expressed in Principle 1, equivalence of gradients:
Principle 1. Let $v$ be any function on $\mathbb{R}^d$ that is constant along normal directions of $\Gamma$. Then, at the surface, intrinsic gradients are equivalent to standard gradients, $\nabla_\Gamma v = \nabla v$.

Similarly, the extension of surface vectors into the embedding space leads to Principle 2, equivalence of divergence:

Principle 2. Let $v$ be any vector field on $\mathbb{R}^d$ that is tangent to $\Gamma$ and tangent to all surfaces displaced by a fixed distance from $\Gamma$. Then, at the surface, $\nabla_\Gamma \cdot v = \nabla \cdot v$.

Combinations of these two principles yield embeddings for more general differential operators [70, 58]. By combining Principles 1 and 2, we may replace the Laplace-Beltrami operator with the standard Laplacian after a constant normal extension of surface values $u$. This yields to the equivalence of diffusion.

To initialize the CPM, we set up the closest point representation of the surface on a tubular computational domain $\Omega_c$ surrounding the surface, and extend the initial values of $u$ onto the computational domain. Time-stepping for the explicit CPM is carried out by alternating two steps:

1. **Closest point extension.** The solution on $\Gamma$ is extended to the computational domain by replacing $u$ with $u(\text{cp}_\Gamma(x))$ for all $x \in \Omega_c$.

2. **Evolution.** The embedding PDE is solved on the tubular computational domain $\Omega_c$ for one time step (or one stage of a Runge-Kutta method).

Note that the closest point extension defined in the first step will require interpolation since $\text{cp}_\Gamma(x)$ is not necessarily a grid point in $\Omega_c$. Following [70], interpolation is carried out using barycentric Lagrange interpolation with polynomial degree $p = q + r - 1$, where $q$ is the order of the differencing scheme and $r$ is the derivative order. The radius of the computational tube depends on the width of the differencing and interpolation stencils. For second-order finite differences, a computational tube radius of

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

is needed in the $d$-dimensional embedding space, using polynomial interpolation of degree $p$ [70].

An implicit formulation of the closest point method [58] is also available. Introducing implicit finite difference schemes in time to the original closest point formulation leads to an unstable system [58]. For illustration purposes, consider the Laplace-Beltrami operator $\Delta_\Gamma$ on a surface $\Gamma$ and $\text{cp}_\Gamma$ is a closest point representation. Then, following the principles 1 and 2, $\Delta_\Gamma$ is replaced with the Laplacian operator $\Delta$ in the embedding PDE, i.e.

$$
\Delta_\Gamma u = \Delta u(\text{cp}_\Gamma)
$$

(4.2)
on a $C^3$-smooth surface, where $u$ is a $C^2$-smooth scalar function and $c \mathbf{p}_r$ is $C^2$-smooth [60]. In matrix form, Equation (4.2) takes the form

$$\Delta u(c \mathbf{p}_r) \approx \Delta_h E u =: \tilde{M}u$$

where $\Delta_h$ is the discretized Laplacian using finite differences and $E$ corresponds to the closest point extension. The matrix $\tilde{M}$ leads to an unstable system when treated implicitly in time [58]. To stabilize the discretization of $\Delta$, a new matrix is introduced of the form

$$M = \text{diag}(\Delta_h) + (\Delta_h - \text{diag}(\Delta_h))E.$$ 

By using $M$ in the discretization of the operator $\Delta$, implicit time-stepping schemes can be formulated.

4.2 The grid based particle method

We now review Leung and Zhao’s grid based particle method (GBPM) for capturing the motion of a surface [51]. The method has been applied to open surfaces [50] and higher-order geometric motions [49].

To initialize the GBPM, a grid is constructed that contains the surface. Grid points that are within a Euclidean distance $\gamma_{GBPM}$ (the tube radius) of the surface are identified: These are the active grid points. As part of the construction of the computational tube, we compute the closest point on the surface for each active grid point. The closest points on the surface are called footpoints in the GBPM and give the surface representation.

After initialization, the system is evolved in time. Each time step of size $\Delta t$ consists of three steps:

1. **Motion**: The footpoints are moved according to the desired motion law.

2. **Resampling**: For each active grid point, the closest point to the surface (as defined by the footpoints) is computed. This gives the updated footpoints.

3. **Update of the Computational Tube**: This step consists of two stages. The first stage activates all the grid points that have neighboring active grid points and applies the resampling step to find their corresponding footpoints. The second stage deactivates all the grid points that are far from the surface (i.e., the distance between the grid point and its footpoint is larger than the tube radius $\gamma_{GBPM}$).

An illustration of the main steps of the GBPM is provided in Figure 4.1. The initialization sets all the grid points that are within a distance $\gamma_{GBPM}$ of the surface to active, and sets the footpoints to be the corresponding closest points. This yields a closest point representation over a computational tube of radius $\gamma_{GBPM}$. The motion step evolves the surface.
Figure 4.1: An illustration of the main steps of the GBPM (from top left to bottom right). Active grid points (blue dots) are connected to their footpoints (red dots) with blue lines. The green lines correspond to the grid.
by moving the footpoints; the footpoints no longer give a closest point representation of
the surface. In the resampling step, the footpoint of each active grid point is re-assigned to
be the closest point on the surface. Finally, we update the computational tube by adding
points that are neighboring the active grid points and by deleting points outside the tube
radius. This restores the tubular closest point representation.

The resampling step involves a local reconstruction of the surface and a local minimiza-
tion problem for each active grid point. The details are analyzed in the next section.

4.2.1 The Resampling Step

In this section, we provide a detailed analysis of the resampling step of the grid based
particle method.

Collection of points

During the resampling step, $m$ points are required for the local reconstruction of the surface.
For large $m$, the surface reconstruction becomes more accurate and results in an increased
computational cost of the resampling step. For each active grid point $p$, a small neighbor-
hood of at least $m$ active grid points ($p_1 = p$, ..., $p_N, N \geq m$) around $p$ are collected. Each
active grid point stores information about its footpoint $y$ on the surface and its unit normal
vector $n$, which might no longer be normal to the surface. The corresponding footpoints
$y_i$ of the grid points $p_i$ are sorted in an ascending order according to their distance from
the grid point $p$. The first $m$ points closest to the active grid point $p$ are chosen. However,
some conditions need to be met.

At first, the corresponding footpoints of the chosen grid points need to be distinct and
have a minimum distance $\delta = O(\Delta x)$ between each other. A small minimum distance $\delta$
might results in an inaccurate local reconstruction of the surface or a strict CFL condition
for time step. A distance of $\delta = 0.35\Delta x$ is chosen in this thesis, unless stated otherwise.

Another important condition on the choice of $m$ points arises when two parts of the
surface are close to each other. To choose $m$ grid points from the same segment of the
surface, a condition on the unit normal vectors is applied. The normal vectors $n$ need to
provide consistent information along the same segment of the surface. This condition is
enforced using

$$n \cdot n_i > \cos \theta_{max}$$

where $n$ is the normal vector stored in the base grid point $p$ and $n_i$ is the normal vector
of a candidate grid point $p_i$. In this thesis, the minimum angle is chosen as $\theta_{max} = \pi/2$,
unless stated otherwise.

Finally, in cases where $m$ points cannot be found that have consistent Lagrangian in-
formation, the reference grid point $p$ is deactivated along with its footpoint.
Change to local coordinates

Following the selection process, a local coordinate system is constructed based on the normal vector \( \mathbf{n} \) of the active grid point \( \mathbf{p} \). The basis of the local coordinate system consists of the unit normal vector and the unit tangent vectors \( \{\mathbf{n}, \mathbf{T} = \mathbf{n}^\perp\} \). To find the tangent vector for curves is straightforward. For surfaces, the two tangent vectors are found using the unique Householder matrix that maps \( \mathbf{n} \) to the vector \((0, 0, 1)\). The local coordinates \( Y_i \) of the footpoints are given by

\[
Y_i = \begin{pmatrix} T^T \\ \mathbf{n}^T \end{pmatrix} (y_i - \mathbf{y}),
\]

where \( \mathbf{T} \) corresponds to the matrix that contains the unit tangent vectors as columns.

Local reconstruction polynomial

Having a collection of \( m \) points in local coordinates \( \{Y_i\}_{1 \ldots m} \), a local polynomial is constructed to approximate the surface in the least squares sense. A polynomial \( f(\mathbf{x}) \) of degree at most \( m - 1 \) is calculated that minimizes the Euclidian distance between \( f(Y_i^n) \) and \( Y_i^n \), where \( Y_i = (Y_i^1, \ldots, Y_i^m) \). For example, for planar curves, a polynomial of the form \( y = f(x) \) is constructed in the least squares sense for the footpoints \( Y_i \), \( i = 1, \ldots, m \). Throughout this thesis, quadratic polynomials are chosen to locally reconstruct the surface.

Finding the new footpoint

Using the local reconstruction polynomial, the new footpoint can be found on the surface by solving a minimization problem

\[
\min_{\mathbf{x}} \|\mathbf{P} - f(\mathbf{x})\|
\]

where \( \mathbf{P} \) is the grid point \( \mathbf{p} \) in local coordinates, \( f(\mathbf{x}) = (x_1, f(x_1)) \) for planar curves and \( f(\mathbf{x}) = (x_1, x_2, f(x_1, x_2)) \) for surfaces. Some necessary conditions need to be met for the new footpoint. Specifically, the local reconstruction process should be an interpolation process and not an extrapolation. Therefore, the coordinates of the new footpoint \( \mathbf{X} \) must lie in between the coordinates of the local reconstruction points, i.e.

\[
\min_i Y_i^1 \leq X_1 \leq \max_i Y_i^1.
\]

If this condition is not met, the grid point and its new footpoint are deactivated.

Calculation of geometric quantities

Using the local reconstruction polynomial, geometric quantities such as the unit normal vector and the curvatures of the surface can be calculated. The unit normal vector can be
calculated either by using the normal vector of the local polynomial $f$ evaluated at the new footpoint $X$ or by using the closest point relation of the grid point $P$ and $X$, i.e.

$$n = \pm \frac{P - X}{\|P - X\|}.$$  

The direction of the newly calculated normal unit vector is given by the previous normal unit vector stored in the grid point $P$. The mean and Gaussian curvatures $\kappa$ and $G$ respectively can be calculated using the local reconstruction polynomial evaluated at the minimizer $X$. Note that the normal unit vector needs to be changed back to Cartesian coordinates but the curvatures are translation and rotation invariant.

The calculation of the mean curvature provides another condition for the newly found footpoint $X$. For high curvature areas, the condition

$$\kappa \leq \frac{c}{\Delta x}$$

is applied, where $c$ is a constant. If the grid is not fine enough to capture the high curvature areas of the surface, then either the grid point and its footpoint are deactivated or the grid is locally refined and the resampling step is repeated. Throughout this thesis, we do not use local adaptivity and we deactivate the grid points that have footpoints with mean curvature larger than $1/\Delta x$.

**Change to global coordinates**

Finally, the new footpoint is transformed back to Cartesian coordinates along with its new unit normal vector. The outcome of the resampling step is stored in the grid point $p$ and the process is repeated for all the active grid points.
Chapter 5

The closest point method coupled with a modified grid based particle method

In this chapter, we propose our first approach to couple the closest point method and the grid based particle method. A modification of the GBPM is necessary in order to perform a simple coupling with the closest point method. Numerical experiments show the convergence of the modified GBPM and the coupled method.

5.1 A modified grid based particle method

In this section, a modified grid based particle method algorithm is proposed which is amenable to coupling with the CPM. Numerical experiments on a variety of geometric motions are provided to verify the correctness of the new algorithm.

5.1.1 Modification

The grid based particle method constructs a closest point representation of the surface at every time step of a surface evolution, making it a natural candidate to consider for coupling with the closest point method. However, a challenge must be overcome before combining the methods: In the closest point method, all nodes within the computational tube must have valid closest point values, otherwise a standard implementation of the method will fail. This condition is not met with the grid based particle method, since the method deactivates grid points within the computational tube in certain situations (see Section 4.2.1).

Note that grid point deactivation is expected to occur with greater frequency in a coupled method than in the standard grid based particle method. The reason for this is that a wider computational tube is required in the closest point method than in the standard grid based particle method, leading to the use of a wider tube in a coupled method as well. To
illustrate the difference, consider a closest point method discretization of diffusion using a second-order finite difference scheme and cubic polynomial interpolation. In the closest point method, this leads to an approximate tube radius of $3.6\Delta x$ in two dimensions and $4.1\Delta x$ in three dimensions using (4.1). In contrast, a standard grid based particle method implementation in 2D or 3D would use a relatively narrow tube radius of up to $1.5\Delta x$.

To obtain a method for geometric surface motion that is compatible with the closest point method, we introduce a (slightly) modified grid based particle method. The modified grid based particle method applies the original grid based particle method for the initialization, motion (step 1), and update of the computational tube (step 3). For resampling (step 2), a change is introduced: Whenever a footpoint deactivation occurs in the original grid based particle method resampling step, the modified method constructs an osculating circle/sphere to locally approximate the curve/surface. The closest point on the reconstruction is accepted as the updated footpoint.

We now detail this additional reconstruction step. Consider first the reconstruction for the case of curves in 2D. When a footpoint deactivation is flagged by the grid based particle method, we select three non-co-linear points that are closest to the grid point $p$ and fit the unique osculating circle through these points. The update to the footpoint is taken to be the local minimizer of distance to $p$ on the osculating circle.

A 2D illustration of the grid based particle method and the proposed modification appears in Figures 5.1 and 5.2. Recall that for a grid point $p$, the grid based particle method collects the $m$ closest footpoints and locally reconstructs the surface. The new footpoint is determined by minimizing the $L^2$-distance between the grid point $p$ and the local reconstruction. However, in some cases, the grid based particle method rejects the new footpoint, and flags the point $p$ for deactivation. Figure 5.1 gives an example where the grid point is flagged for deactivation because the minimizer (black star) lies outside the
m = 6 interpolating points. In our modified algorithm, an osculating circle based on the three points closest to \( p \) is constructed. The closest point (black star) on the circle to \( p \) gives the updated footpoint; see Figure 5.2.

For the case of surfaces, a similar strategy is employed. Specifically, whenever a grid point is flagged for deactivation by the grid based particle method, we reconstruct the surface by fitting a sphere to four non-coplanar footpoints. Spherical shapes have been shown to be effective for reconstructing surfaces moving by curvature-dependent speeds in Hon et al. [44]. See [44] for a method that uses circles and spheres to locally reconstruct curves and surfaces in the least squares sense.

For both cases (circles and spheres), it is necessary to impose the condition that the new footpoint lies at the same side as the interpolation points. Two candidate points are considered: the minimizer and its antipodal point on the circle/sphere. The point closest to the closest point of the reference grid point is the new footpoint.

With this proposed modification, all grid points within the computational tube remain active. We emphasize that the osculating circle and sphere reconstruction should not be used as a replacement for the grid based particle method. The grid based particle method is an effective, tested method for reconstructing a closest point representation. Only rarely, when grid point deactivation occurs, do we apply the osculating circle/spherical reconstruction.

5.2 Numerical tests of modified GBPM

In this section, we apply the modified GBPM to a number of examples to illustrate the performance and versatility of the method. In view of our objective of coupling the method with the CPM, we select a tube radius of \( \gamma_{CPM} \) (see Equation (4.1)). Quadratic polynomials are used to locally reconstruct the surfaces in the least squares sense. In two dimensions,
Figure 5.3: Blue lines give the computed radii as a function of time for the circle (left) and the sphere (right). Red dots mark the exact solution at selected times.

$m = 6$ footpoints with a minimum distance of $\delta = 0.35\Delta x$ between one another are used, while $m = 20$ footpoints with $\delta = 0.5\Delta x$ are used in three dimensions. The motion step is carried out by evolving the ODE system

$$\frac{dx}{dt} = v$$

with a step of forward Euler. In all examples we use a uniform Cartesian grid with $\Delta x = \Delta y = \Delta z = h$.

### 5.2.1 Mean curvature motion on a circle and a sphere

We begin by considering mean curvature motion on a circle and a sphere. The velocity is given by

$$v = -\kappa n$$

where $\kappa$ is the mean curvature and $n$ is the outward unit normal vector. The analytical solution may be found by integrating

$$\dot{R} = \kappa$$

with $\kappa = 1/R$ for the circle and $\kappa = 2/R$ for the sphere. For all times $t \geq 0$, this gives

$$R_{\text{exact}}(t) = \sqrt{R_0^2 - 2t}$$

for the case of a circle and

$$R_{\text{exact}}(t) = \sqrt{R_0^2 - 4t}$$

for the sphere.

In our discretization, we select a spatial step-size of $\Delta x = 0.00625$ for the circle and $\Delta x = 0.0125$ for the sphere. A forward Euler time discretization is chosen with step-
size $\Delta t = 0.4\Delta x^2$. In both cases, the computed results give a good match to the exact solution. See Figure 5.3 for details. In addition, Figure 5.4 shows the $\infty$-norm error of the approximate radius in comparison to the exact radius $R_{exact}(t)$ over time. The method is second-order convergent away from the final time of the evolution for both cases.

5.2.2 Vortex flow with rewind

To test area conservation, we follow [53] and apply a vortex flow motion followed by a reversal of the velocity field. Starting from a circle centered at $(0.5, 0.75)$ with radius 0.15, we apply the velocity $\mathbf{v} = (v_1, v_2)$, where

$$v_1 = -\sin^2(\pi x) \sin(2\pi y) \cos(\frac{\pi t}{T}),$$
$$v_2 = \sin(2\pi x) \sin^2(\pi y) \cos(\frac{\pi t}{T}),$$

and $T = 4$. This vortex flow with rewind motion yields the original surface at the prescribed final time $T$. In this example, different parts of the surface get close to one another, so a check on the consistency of the Lagrangian information is applied. Specifically, we require that

$$\mathbf{n}_0 \cdot \mathbf{n} > \cos\left(\frac{\pi}{2}\right)$$

for every footpoint added in the collection of $m$ footpoints, where $\mathbf{n}_0$ is the unit normal vector of the closest footpoint to the reference grid point and $\mathbf{n}$ is the unit normal of the candidate footpoint.

We compute to the final time $T = 4$ using a grid spacing $\Delta x = 0.0015625$ and a time step-size $\Delta t = 0.8\Delta x$. Results at various times are displayed in Figure 5.5. A good visual match is observed between the initial and final contours. Indeed, the mean radius of the final contour is $R = 0.1505$, which is a 0.33% change from the initial circle.
Figure 5.5: The evolution of a circle under vortex flow with rewind at various times $t$ (from top left to bottom right).
5.2.3 Merging circles

We now consider an example that includes topological change. Starting from two circles with radius 0.15 centered at (0.4, 0.4) and (0.6, 0.6), we evolve outwards in the normal direction with constant unit speed. Under this flow, the two circles intersect and merge to form a single closed curve.

Evolving to time $T = 0.1$ using the modified GBPM with a mesh spacing $\Delta x = 0.0015625$ and a time step-size $\Delta t = 0.5\Delta x$ yields the results displayed in Figure 5.6. In agreement with [51], we observe that the surfaces intersect to form a single curve which continues its outward normal expansion. In this example, it was necessary to deactivate grid points and their corresponding footpoints to ensure the merging of the surface; see Section 4.2.1 for further details.

5.2.4 Vortex flow with rewind in 3D

Volume-conserving flows are also of considerable interest. We consider next vortex flow with rewind applied to an initial sphere with radius 0.15 centered at (0.35, 0.35, 0.35). The
velocity is given by $v = (v_1, v_2, v_3)$, where

$$v_1 = 2 \sin^2(\pi x) \sin(2\pi y) \sin(2\pi z) \cos\left(\frac{\pi t}{T}\right),$$

$$v_2 = -\sin(2\pi x) \sin^2(\pi y) \sin(2\pi z) \cos\left(\frac{\pi t}{T}\right),$$

$$v_3 = -\sin(2\pi x) \sin(2\pi y) \sin^2(\pi z) \cos\left(\frac{\pi t}{T}\right),$$

and $T = 1.5$. In this example, different segments of the surface get close to one another so Lagrangian consistency information is used. Similar to vortex flow with rewind in 2D, for this example we insist that

$$\mathbf{n}_0 \cdot \mathbf{n} > \cos\left(\frac{\pi}{3}\right)$$

for every footpoint added in the collection of $m$ footpoints, where $\mathbf{n}_0$ is the unit normal vector of the closest footpoint to the reference grid point and $\mathbf{n}$ is the unit normal of the candidate footpoint.

Selecting a mesh spacing $\Delta x = 0.0083$ and a time step-size $\Delta t = 0.8\Delta x$, we obtain the evolution displayed in Figure 5.7. We find that our final shape ($t = 1.5$) is in good agreement with the initial sphere ($t = 0$). The mean radius of the final shape is $R = 0.1534$ with a standard deviation of 0.0048.

### 5.2.5 Topological change in 3D

We conclude our tests for geometric motion with an example exhibiting topological change in three dimensions.

Start from an initial dumbbell shape, the surface evolves inwards by its mean curvature in the normal direction, i.e. the velocity is given by

$$v = -\kappa \mathbf{n}.$$ 

Taking discretization step-sizes $\Delta x = 0.0333$ and $\Delta t = 0.4\Delta x^2$, and a final time $T = 0.03$, we obtain the results displayed in Figure 5.8. The modified GBPM captures the split that arises in this well-known example. An examination of the results indicates that the topological change occurs at approximately $t = 0.021$.

In this example, the deactivation of grid points was necessary to ensure the splitting of the dumbbell; see Section 4.2.1 for details. Therefore, the modified grid based particle method is not capable of constructing a closest point representation that maps all the grid points to their closest points on the surface in a neighborhood of the surface when topological changes occur.
Figure 5.7: The evolution of a sphere under vortex flow with rewind at selected times $t$. The final time is $T = 1.5$. 
Figure 5.8: The evolution of a dumbbell under mean curvature motion at selected times $t$. 
5.3 A coupled method

We now introduce our coupled method. The proposed method employs a simple coupling of the closest point method and the modified grid based particle method for solving PDEs on moving surfaces.

To begin, we apply the initialization step of the CPM using a tube radius $\gamma$ (which we shall define later). This gives a closest point representation over the computational tube. Nodal values are initialized using a closest point extension of initial surface values: $u(x) = u(cp(x), 0)$.

After initialization, the system is evolved in time. Each time step of size $\Delta t$ consists of two steps:

1. **CPM evolution**: Steps 1-2 of the CPM are applied using a time step-size $\Delta t$ and a tube radius $\gamma$. (See Section 4.1 for details on the CPM.)

2. **Modified GBPM evolution**: We now think of the closest point representation on the computational tube as a collection of footpoints defined on a set of active grid points. Steps 1-3 of the modified GBPM are applied using a time step-size $\Delta t$ and a tube radius $\gamma$. This gives a closest point representation of the time-evolved surface over a tube of radius $\gamma$. (See Section 5.1.1 for details on the modified GBPM.)

We now discuss the tube radius $\gamma$. Let $\Gamma_n$ be the computed surface after $n$ time steps, and $S_\gamma^n$ be the corresponding discrete computational tube (i.e., the set of grid points that are within a distance $\gamma$ of $\Gamma_n$). The approximation of $u$ (from the CPM evolution) will be defined on the set $S_{n-1}^{\gamma}$, while the closest point representation (from the modified GBPM evolution) will be defined on the set $S_n^{\gamma}$. The CPM step uses both these quantities over a tube of radius $\gamma_{CPM}$ (see Section 4.1). This suggests setting the tube radius

$$\gamma = \gamma_{CPM} + \Delta t \cdot v_{max}^{n}$$  \hspace{1cm} (5.1)

where $v_{max}^{n}$ is a bound on the normal speed of a footpoint at time step $n$. While (5.1) does not guarantee a complete tube of nodal values in the CPM step, it was not observed to fail in our numerical tests.

Condition (5.1) is too conservative and in some cases results in a larger computational tube than necessary. In practice, we select $\gamma$ adaptively. Specifically, we set $\gamma = \gamma_{CPM}$ and check for violations of the tube radius. Such events are rare, and when they arise we simply apply a larger computational tube (e.g., we compute using (5.1)).

For the advection-diffusion equation (3.5) on a moving surface, the closest point method coupled with the modified grid based particle method consists of the following algorithmic steps:
• Construct a grid \( \{z_j\} \) that contains the initial surface \( \Gamma_0 \) and compute a closest point representation of the surface \( \text{cp}_{\Gamma_0} \) to map all the grid points \( \{z_j\} \) to their closest points \( \{x_j\} \) on the surface. A computational tube \( S_0^0 \) is constructed with \( \gamma = \gamma_{CPM} \).

• For each iteration \( n = 1, ..., N \) of time step-size \( \Delta t \):
  
  – Apply the resampling step of the original GBPM as described in Section 4.2.1. In case a grid point is flagged for deactivation, apply the modification of the GBPM (see Section 5.1.1) to find its new footpoint. The velocity \( v^n_j \) is found for each new footpoint.

  – Update the computational tube according to the original GBPM (see Section 4.2). In particular:
    1. The activation phase is applied: all the grid points neighboring to active grid points are activated and the resampling step is applied. The velocity of each footpoint \( v^n_j \) is calculated.
    2. Check whether the maximum velocity \( \max_j(v^n_j)\Delta t \) exceeds the computational tube radius \( \gamma \). In such occurrence, update the tube radius \( \gamma \) using Equation (5.1) and repeat step 1, otherwise continue.

  The deactivation phase using the tube radius \( \gamma \) is applied. The new computational tube \( S_n^0 \) is constructed.

  – A closest point extension step is applied to extend the solution to the computational tube \( S_n^0 \).

  – The finite differences matrices of the Laplacian \( \Delta_h \) and the first derivatives \( D_x \) and \( D_y \) using centered schemes are found.

  – Solve the PDE for one time step using either the explicit or the implicit closest point method.

  – Restore the value of \( \gamma \) to its initial value \( \gamma_{CPM} \) to ensure that a computational tube of minimum size is constructed at each time step.

Note that an initial step of the modified GBPM is applied before the application of the CPM because the advection-diffusion equation (3.5) requires the calculation of the mean curvature.

Our coupled method uses the same time step-size for both the evolution of the surface and the solution of the PDE. In order to avoid stability issues, we use the smallest time step-size that satisfies the CFL conditions that arise from the motion law and the PDE. Further analysis needs to be done in case of different time step-size options for each method.
5.4 Numerical experiments

In this section, the coupled method is applied to a number of examples in two and three dimensions. Unless stated otherwise, we solve Equation (3.5) with a flux \( q = \nabla \Gamma u \). This corresponds to diffusion with diffusivity parameter \( D = 1 \). We note that the use of the CPM extension step (see Section 4.1) in our coupled method leads to a vanishing normal derivative, \( \partial u / \partial n = 0 \), and a corresponding simplification in Equation (3.5):

\[
    u_t - V \kappa u + \nabla \cdot (u(\mathbf{cp}_\Gamma)\mathbf{T}) - \Delta u(\mathbf{cp}_\Gamma) = 0.
\] (5.2)

Unless stated otherwise, forward Euler is used for the discretization in time and centered finite difference schemes for the discretization in space.

5.4.1 Diffusion on an expanding circle

In our first experiment, we approximate diffusion on an expanding circle (cf. [24]). This example illustrates the convergence of our coupled method in two dimensions. The surface starts from a unit circle and evolves outwards with a constant velocity given by \( \mathbf{v} = 5\mathbf{n} \).

The exact solution of the homogeneous PDE (5.2) on our evolving circle is

\[
    u(\theta, t) = e^{4/(5r(t))} \frac{\cos \theta \sin \theta}{r(t)}
\]

where \( r(t) = 1 + 5t \) is the radius of the circle at time \( t \). The result may be verified by substituting \( u \) into the PDE (3.6) and noting that

\[
    \nabla \Gamma = (-\sin \theta, \cos \theta) \frac{\partial}{\partial \theta} \left\| \mathbf{x}_t(\theta, t) \right\|,
\]

where \( \Gamma(t) \) is the moving surface and \( \mathbf{x}(\theta, t) = r(t)(\cos(\theta), \sin(\theta)) \) is the parametrization of the circle. Because there are no tangential terms in the velocity, diffusion dominates the flow. The evolution of the surface and the solution are depicted in Figures 5.9 and 5.10 using a spatial step of \( \Delta x = 0.025 \) and a time step-size of \( \Delta t = 0.4 \Delta x^2 \).

Using a uniform grid of \( \Delta x = 0.025 \), the total computational time for our coupled method’s algorithm after 5 time steps is 23.97 seconds on a 4-core laptop in MATLAB. The total time of the modified GBPM algorithm alone after 5 time steps is 23.61 seconds using a parallel implementation. The surface is sampled by 1,372 points at the completion of the 5th time step. The computational time for the calculation of the solution of the linear matrix system is negligible, despite the choice of the explicit or implicit closest point method.
Figure 5.9: The graph of the approximate solution of the diffusion model on the evolving circle at selected times $t$. The solution is depicted in relation to the surface parameter $\theta$. 
Figure 5.10: The graph of the approximation of the solution of the diffusion model on the evolving circle at selected times $t$. The solution is shown in relation to the $x$ and $y$ coordinates.
We compute the absolute error as measured in the infinity norm at various times \( k\Delta t \), i.e., we compute
\[
\max_i |u(x_i, k\Delta t) - U^k_i|
\]
where \( u(x_i, k\Delta t) \) denotes the analytical solution at node \( x_i \) and \( U^k_i \) denotes the corresponding numerical result. As shown in Figure 5.11, errors accumulate in time and are reduced by mesh refinement.

To clarify the convergence rates, Table 5.1 provides the results and a numerical convergence study for selected times \( t \). In all four cases, a clear second-order spatial convergence is observed.

### 5.4.2 Advection-diffusion on an oscillating sphere

Moving to three dimensions, the convergence of our coupled method is examined on the solution of the advection-diffusion PDE on an oscillating sphere. We apply the velocity
\[
v = \frac{a'(t)}{2a(t)}(x_1, 0, 0),
\]
where
\[ a(t) = 1 + \sin(2t) \]
to the unit sphere centered at the origin. Following [24], the exact solution of the surface at all times \( t \geq 0 \) is
\[
\mathbf{x}(\theta, \phi, t) = (\sqrt{a(t)} \cos(\theta) \cos(\phi), \sin(\theta) \cos(\phi), \sin(\phi))
\]
where \( \theta \) and \( \phi \) correspond to the azimuth and the elevation of the sphere respectively. Notice that this velocity causes the sphere to oscillate along the \( x \)-axis with period \( \pi/2 \).

To form our surface PDE, we insert \( u(x, t) = e^{-6t} x_1 x_2 \) into the non-homogeneous PDE (3.6) and compute the right-hand side \( f \). This gives
\[
f = \left( -6 + \frac{a'(t)}{a(t)} \left( 1 - \frac{x_1^2}{2N} \right) + \frac{1}{N} + \frac{a(t)}{N^2} \left( x_1^2 + a(t)(x_2^2 + x_3^2) \right) \right) u,
\]
with
\[ N = x_1^2 + a(t)(x_2^2 + x_3^2). \]

Using a mesh spacing \( \Delta x = 0.1 \) and a time step-size \( \Delta t = 0.1 \Delta x^2 \), we obtain the numerical solution displayed in Figures 5.12 and 5.13 at the final time \( T = \pi/2 \). The relative error at the final time is less than 5\% as measured in the infinity norm.

For our three dimensional example, using a uniform grid of \( \Delta x = 0.2 \), the total computational time for our coupled method’s algorithm after 5 time steps is 207.21 seconds on a 4-core laptop in MATLAB. The total time of modified GBPM algorithm alone after 5 time steps is 203.46 seconds using a parallel implementation. The surface is sampled by 3,232 points at the completion of the 5th time step. The computational time for the calculation of the solution of the linear matrix system is negligible, despite the choice of the explicit or implicit closest point method.

Figure 5.14 gives a plot of the error as a function of the elapsed time for various mesh sizes. We observe that the error accumulates in time, and that the error decreases with
Figure 5.12: The graph of the numerical solution of an advection-diffusion model on an oscillating ellipsoid at various times \( t \). The solution is plotted as a graph of the surface parameters \( \theta \) and \( \phi \). Yellow corresponds to large solution values and dark blue corresponds to small values.
Figure 5.13: The graph of the numerical solution of an advection-diffusion model on an oscillating ellipsoid at various times $t$. The solution is visualized on the evolving surface. Yellow corresponds to large solution values and dark blue corresponds to small values.

Figure 5.14: The absolute error of the numerical solution of the advection-diffusion model on the oscillating ellipsoid over time for various mesh spacings.
mesh refinement. Table 5.2 gives a convergence study for times $t = 0.01, 0.02, 0.03$ and $0.04$. In all four cases, second-order convergence is observed.

### 5.4.3 Strongly coupled flow on an evolving torus

Next, we evolve an initial torus according to a velocity

$$ \mathbf{v} = (0.1\kappa + 5u)\mathbf{n} $$

where $\kappa$ is the mean curvature, $u$ is the solution of the PDE (3.5) and $\mathbf{n}$ is the unit normal vector. Because the surface velocity depends on the solution of the PDE, we call this system *strongly coupled*. We select

$$ u(x, 0) = 1 + 20x_1x_2x_3 $$

for the initial condition.

Using a mesh spacing $\Delta x = 0.05$ and a time step-size $\Delta t = 0.2\Delta x^2$, we obtain the evolution displayed in Figure 5.15. In the initial flow, the torus experiences some rapid changes in the normal direction from the variation in the solution $u$. Over time, diffusion causes $u$ to become more uniform and the surface motion slows.

### 5.4.4 A model for solid tumor growth

Our final example considers solid tumor growth in the avascular growth phase, as presented in [26]. The mathematical model for this example has the form

$$ \frac{Du}{Dt} = \Delta_\Gamma u - u\nabla_\Gamma \cdot \mathbf{v} + f_1(u, w), $$

$$ \frac{Dw}{Dt} = \mathcal{D}\Delta_\Gamma w - w\nabla_\Gamma \cdot \mathbf{v} + f_2(u, w) $$

where $u$ and $w$ are scalar quantities on an evolving surface $\Gamma(t)$, $\frac{D}{Dt}$ is the material derivative, $\mathbf{v}$ is the velocity of $\Gamma(t)$, $\mathcal{D}$ is a positive constant and $f_1, f_2$ are functions that couple the solutions of the PDEs. For this example, we set

$$ f_1(u, w) = 100(0.1 - u + u^2w), $$

$$ f_2(u, w) = 100(0.9 - u^2w) $$

and

$$ \mathbf{v} = (0.01\kappa + 0.4u)\mathbf{n} $$
Figure 5.15: A visualization of the numerical solution of the strongly coupled flow at selected times $t$. The initial surface is a torus.
Figure 5.16: A visualization of the numerical solution $u$ of the tumor growth model (5.3) at selected times $t$. The initial surface is a sphere.

where $n$ is the unit normal vector and $\kappa$ is the mean curvature. Finally, we set the diffusivity constant $D = 10$. Two identical initial conditions are chosen:

$$u(x, 0) = 1 + 2x_1x_2x_3 = w(x, 0).$$

Using a mesh spacing of $\Delta x = 0.05$ and a time step-size $\Delta t = 0.02\Delta x^2$ yields the results displayed in Figures 5.16 and 5.17. We observe tumor growth with preferred directions according to the solution $u$.

A smaller time step-size of $\Delta t = 0.02\Delta x^2$ is used here to ensure stability for a relatively large diffusivity parameter $D = 10$. While this was sufficient for our purposes, stable, implicit methods are also available. See [58] for details.
Figure 5.17: A visualization of the numerical solution $w$ of the tumor growth model (5.3) at selected times $t$. The initial surface is a sphere.
Chapter 6

An explicit closest point method using RBF-FD

In this chapter, we introduce an explicit closest point method using finite differences derived from radial basis functions (see Section 2.4). A detailed description of the proposed method is provided and numerical experiments show the convergence of the method for different types of surface PDEs.

6.1 An explicit formulation of the closest point method using RBF-FD

We state an explicit formulation of the closest point method for solving PDEs on surfaces using RBF-FD. We propose the use of RBF-FD stencils that consist of \( m \) closest neighboring grid points and we provide two different approaches for the calculation of a computational tube around the surface.

6.1.1 Method’s description

In the explicit closest point method [70], surface differential operators \( L_\Gamma \) on a surface \( \Gamma \) are replaced by the corresponding operators \( L \) in the Cartesian coordinate system. Applying Principles 1 and 2 and a closest point representation of the surface \( \text{cp}_\Gamma \)

\[
L_\Gamma u(x) = Lu(\text{cp}_\Gamma(x))
\]

on the surface \( \Gamma \). Suppose that \( \{z_j\} \) are \( k \) Cartesian grid points around the surface \( \Gamma \) and \( \{\text{cp}_\Gamma(z_j)\} \equiv \{x_j\} \) are their corresponding closest points on the surface \( \Gamma \) using a closest point representation \( \text{cp}_\Gamma \). A discretization, \( L_h \), of the operator \( L \) can be obtained using an
RBF approximation as in Equation (2.9):

\[ L_h = BA^{-1}, \]

where \( A = [\phi(\|z_i - z_j\|)]_{k \times k} \), \( B = [L\phi(\|z_i - z_j\|)]_{k \times k} \) and \( h \) is the spatial grid size. Also, an extension matrix \( E \) can be found that assigns a value \( u(\mathbf{c}\mathbf{p}_\Gamma(z_j)) = u(x_j) \) to the value \( u(z_j) \) using a modified version of Equation (2.9), i.e.

\[ E = DA^{-1}, \]  

where \( D = [\phi(\|x_i - z_j\|)]_{k \times k} \). Then, using a closest point representation \( \mathbf{c}\mathbf{p}_\Gamma \) of the surface \( \Gamma \), the discretization of the operator \( L\Gamma \) takes the form

\[ L\Gamma u(x) = L[u(\mathbf{c}\mathbf{p}_\Gamma(x))] \approx L_h E u, \]

on the surface \( \Gamma \), where \( L_h \) is a discretization of the operator \( L \). Notice that the explicit closest point method evaluates the derivatives of the function \( u(\mathbf{c}\mathbf{p}_\Gamma(x)) \) on the grid points \( \{z_j\} \) in the embedding space. Our RBF-FD method instead calculates the derivatives directly on the closest points \( \{x_j\} \) on the surface via

\[ L\Gamma u(x) = Lu|_{\mathbf{c}\mathbf{p}_\Gamma(x)} \approx L_h^{RBF} u. \]

By applying the differentiation on the surface points \( \{x_j\} \), the approximation of the differential operator takes the form

\[ L_h^{RBF} = FA^{-1} \]

where \( F = [L\phi(\|x_i - z_j\|)]_{k \times k} \) and \( A \) is defined as before. Thus, our RBF-FD implementation of the closest point method uses the closest point function to represent the surface, and to update the solution in the embedding space using quantities evaluated on the surface. In particular, a direct evaluation of the derivatives on the closest points \( \{x_j\} \) on the surface is obtained without the calculation of derivatives on the grid nodes \( \{z_j\} \). The matrices \( F \) and \( A \) are computed using the stable RBF-GA method [34] (see Section 2.4.1).

The RBF finite difference stencil used in this paper consists of the \( m \) closest grid points \( z_j, j = 1, ..., m \), for each closest point \( x_i \) on the surface. Methods that use RBF-FD stencils of \( m - 1 \) nearest neighbors on scattered nodes require the use of quasi-uniform nodes on the surface [74]. By using RBF-FD stencils on the grid nodes, we avoid the ill-conditioning of the RBF-FD matrices that arises from the clustering of nodes on the surface.

In our algorithm, a preprocessing step is required to calculate the RBF-FD matrices. This involves a call of the RBF-GA method for each point on the surface. Specifically, for each surface node \( x_i \):
1. Find the $m$ closest grid points $\{z_j\}$ to the closest point $x_i$ on the surface.

2. Use the RBF-GA algorithm to find the finite difference weights of the discretized operator $L_{h,RB}^{RF}$ evaluated at the closest points $x_i$ on the surface. The grid points $\{z_j\}$ are treated as the RBF centers.

These two steps are repeated for each closest point $x_i$ on the surface, $i = 1, \ldots, k$. On fine grids, this process can be computationally expensive, however a parallel implementation is possible as there is no interaction between computations for each node.

To solve time dependent PDEs on surfaces, the RBF-FD formulation of the explicit closest point method requires a time discretization. For illustration purposes, consider the PDE

$$u_t = L_\Gamma u$$

on a surface $\Gamma$. Using forward Euler time stepping and an RBF-FD discretization of the operator $L_\Gamma$, the discretized equation takes the form

$$U^{n+1} = EU^n + \Delta t L_{h,RB}^{RF} U^n$$

where $U$ is the approximate solution and $E$ is the extension matrix defined as before.

### 6.1.2 Parameters

There are two parameters appearing in the RBF-FD formulation of the explicit closest point method. These are the shape parameter $\epsilon$ of the Gaussian RBFs and the number of points $m$ in the stencil used locally for the RBF interpolation. In this section, we analyze the dependence of the numerical method on these parameters.

We consider two test problems to measure the error of the discrete Laplace-Beltrami operator in comparison to the exact. In both test cases we consider a uniform Cartesian grid with $\Delta x = \Delta y = \Delta z = h$. The first test problem that we consider is the approximation of the Laplace-Beltrami operator applied to the function $u(\theta) = \sin(\theta)$ on the unit circle.

We discretize the Laplace-Beltrami operator using the RBF-FD approximation $\Delta h^{RB}$ and a grid size $\Delta x$. The relative error is measured in the infinity norm as

$$\|\Delta h^{RB} u + u\|_\infty,$$

using the exact solution $\Delta_\Gamma u = -u$ and $u = [\sin(\theta)]$. We choose $m = 20$ points in the stencil and a spatial grid size of $\Delta x = 0.05$. 

58
In our second problem, we approximate the Laplace-Beltrami operator applied to the function \( u(\theta, \phi) = \sin(\phi) \) on the unit sphere \( \Gamma \)

\[
\Gamma = \{ x : x(\theta, \phi) = (\cos(\theta) \cos(\phi), \sin(\theta) \cos(\phi), \sin(\phi)), -\pi \leq \theta < \pi, -\frac{\pi}{2} \leq \phi \leq \frac{\pi}{2} \}.
\]

The Laplace-Beltrami operator is discretized using the RBF-FD approximation \( \Delta^{RBF}_h \), where \( \Delta x \) is the grid size. The relative error is measured in the infinity norm as

\[
\frac{\| \Delta^{RBF}_h u + 2u \|_\infty}{2},
\]

using the exact solution \( \Delta_\Gamma u = -2u \) and \( u = [\sin(\phi_j)] \). We select \( m = 60 \) points and a grid size of \( \Delta x = 0.2 \).

To begin, we consider varying the shape parameter \( \epsilon \) that describes the flatness of the Gaussian RBF \( \phi(r) = e^{-(\epsilon r)^2} \). The smaller the parameter \( \epsilon \), the more flat the Gaussian functions are, forming an ill-condition basis to expand in. On the other hand, higher accuracy can be obtained using a small parameter \( \epsilon \). Using the RBF-GA algorithm in the calculation of a well-conditioned basis for expansion (see Section 2.4.1), the impact of the shape parameter in our RBF-FD closest point method appears to be minimal: Figure 6.1 shows that in both test problems the change in the \( \infty \)-norm relative error is very small, provided \( \epsilon \) is not too close to 1. For this reason, we select \( \epsilon = 0.5 \) in our numerical tests. We further note that the solution does not diverge as \( \epsilon \) gets small because of the use of the stable RBF-GA method.

We can also test how the number of points \( m \) in the finite difference stencil affects the order of convergence of the RBF-FD method. Without the use of a computational tube, Figure 6.2 shows the convergence of the RBF-FD approximation of the Laplace-Beltrami
operator for various test cases. Observe that the order of the method increases as the number of points in the finite difference stencil increases.

### 6.1.3 Computational tube

In the original formulation of the closest point method, computations are restricted to a computational tube around the surface [70]. Based on the accuracy required, a formula (4.1) is provided for the computational tube radius depending on the degree of the interpolation polynomial, the grid spacing and the dimension of the embedding space.

The RBF-FD stencils considered in this thesis are constructed differently than the standard finite difference schemes. For each node on the surface, the finite difference stencil consists of its $m$ closest grid points. Therefore, a computational tube around the surface should include these $m$ grid points for any node on the surface. Figure 6.3 shows a 13 point stencil in red circles for the green surface node.

A kd-tree structure [72] seems a natural choice for the construction of a computational tube around the surface. Specifically, for each surface node $x_i$, the kd-tree stores the indices of the $m$ closest grid points $\{z_j\}$ that appear in the RBF-FD stencil of the node $x_i$.

An alternative approach to the calculation of the computational tube radius $\gamma$ uses the famous Gauss circle problem. In its original formulation, the Gauss circle problem is posed as follows: Find the number of integer lattice points $m$ inside a circle with radius $r$ centered at the origin. The problem can also be formulated as: Find the number of ordered pairs $(x, y)$, with integers $x, y \geq 0$, such that

$$x^2 + y^2 \leq q$$
Figure 6.3: An example of a 13-point stencil (black points with red circles) for the approximation of a differential operator for a point on the interface (green point). The red transparent circle centered at the surface node contains the closest grid points to the reference surface node.

<table>
<thead>
<tr>
<th>$q$</th>
<th>$m$ (2D)</th>
<th>$m$ (3D)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>7</td>
</tr>
<tr>
<td>2</td>
<td>9</td>
<td>19</td>
</tr>
<tr>
<td>3</td>
<td>9</td>
<td>27</td>
</tr>
<tr>
<td>4</td>
<td>13</td>
<td>33</td>
</tr>
<tr>
<td>5</td>
<td>21</td>
<td>57</td>
</tr>
</tbody>
</table>

Table 6.1: The number of lattice points $m$ contained in a ball with radius $r = \sqrt{q}$ in two and three dimensions.

where the radius of the circle is chosen as $r = \sqrt{q}$, for a fixed integer $q \geq 0$. Generalizations to higher dimensions are also available. In three dimensions, the problem uses a sphere centered at the origin: Find the number of ordered triplets $(x, y, z)$, with integers $x, y, z \geq 0$, such that

$$x^2 + y^2 + z^2 \leq q$$

where the radius of the sphere is $r = \sqrt{q}$, for an integer $q \geq 0$.

The integer solutions of the Gauss circle problem and its generalization to three dimensions are given by sequences A057655 (two dimensions) and sequences A117609 (three dimensions) in the On-Line Encyclopedia of Integer Sequences [2]. Table 6.1 shows some of the integer solutions for the Gauss circle problem in two and three dimensions. See Appendix A for more information on the Gauss circle problem.

For an RBF-FD stencil that uses the $m$ closest grid points to a surface point, a circle can be constructed centered at the surface node that contains these $m$ grid points. Figure 6.3
shows an example of a circle centered at the green surface node (red transparent circle) that contains the 13 red circled grid points that appear in the RBF-FD stencil. In order to construct a computational tube around the surface using the Gauss circle problem, we need to consider two cases regarding the position of the surface node and the surrounding grid points. In the first case, the surface node lies on a grid point. In such an occurrence, the Gauss circle problem can be applied directly, scaled properly with the grid size $\Delta x$. In the second case, the surface node does not lie on a grid point. In order to use the Gauss circle problem, the distance between the surface node and its closest grid point needs to be added to the radius $r$ of the Gauss circle problem. The worst case appears in Figure 6.4, where the surface node lies midway between all four surrounding grid points.

Taking into consideration these two cases, a relation between the computational tube radius $\gamma$ (corresponding to the circle/sphere radius $r$) and the number of points $m$ in the RBF-FD stencil can be found; see Table 6.2.

### 6.2 Numerical experiments

In this section, we test our method on a number of examples in two and three dimensions. For problems involving the Laplace-Beltrami operator, we opt for a second-order approximation by choosing $m = 13$ in two dimensions and $m = 57$ in three dimensions (cf. Figure 6.2). The radius of the computational tube is set according to the values specified in Table 6.2. The RBF shape parameter is chosen to be $\epsilon = 0.5$, though other small values of $\epsilon$ could be chosen without any effect on the accuracy of the computational results.
Table 6.3: Relative error of the approximate solution at time $t = 1$ for the heat equation on the unit circle as measured in the infinity norm and the convergence rates.

<table>
<thead>
<tr>
<th>$\Delta x$</th>
<th>Rel. error ($t = 1$)</th>
<th>Conv. rates</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.2</td>
<td>$4.30 \times 10^{-3}$</td>
<td>-</td>
</tr>
<tr>
<td>0.1</td>
<td>$9.86 \times 10^{-4}$</td>
<td>2.14</td>
</tr>
<tr>
<td>0.05</td>
<td>$1.87 \times 10^{-4}$</td>
<td>2.40</td>
</tr>
<tr>
<td>0.025</td>
<td>$2.31 \times 10^{-5}$</td>
<td>3.02</td>
</tr>
<tr>
<td>0.0125</td>
<td>$7.53 \times 10^{-6}$</td>
<td>1.62</td>
</tr>
<tr>
<td>0.00625</td>
<td>$1.20 \times 10^{-6}$</td>
<td>2.65</td>
</tr>
<tr>
<td>0.003125</td>
<td>$2.37 \times 10^{-7}$</td>
<td>2.34</td>
</tr>
</tbody>
</table>

according to Figure 6.1. Unless stated otherwise, a uniform Cartesian grid is used with $\Delta x = \Delta y = \Delta z = h$ and forward Euler is used in the discretization of the time derivatives with time step-size $\Delta t = 0.1 \Delta x^2$.

A sample MATLAB implementation of the RBF-FD formulation of the explicit closest point method is provided in Appendix B.1.

6.2.1 Heat equation on a circle

In our first experiment, we are interested in the second-order approximation of the second surface derivative using RBF-FD stencils. Consider the surface heat equation

$$u_t = \Delta_\Gamma u$$

on the unit circle $\Gamma$. Following [70], for an initial profile $u(\theta, 0) = \sin \theta$, the exact solution is

$$u(\theta, t) = e^{-t} \sin \theta.$$

Table 6.3 shows the relative errors as well as the convergence rates for different grid sizes $\Delta x$. Second-order convergence is observed.

6.2.2 Advection equation on an ellipse

The next example is the advection equation on an ellipse. We are interested in the second-order approximation of first derivatives using RBF-FD stencils. Following [70], the equation

$$u_t + u_s = 0,$$

with $s$ being the arclength, is imposed on an ellipse with major axis $b = 1.25$ along the $y$-axis and minor axis $a = 0.75$ along the $x$-axis. By [70], application of the closest point principles to the surface PDE gives

$$u_t + \mathbf{T}(x, y) \cdot \nabla u = 0$$
Table 6.4: Relative error of the approximate solution at time $t = 1$ for the advection equation on an ellipse as measured in the infinity norm and the convergence rates.

<table>
<thead>
<tr>
<th>$\Delta x$</th>
<th>Rel. error ($t = 1$)</th>
<th>Conv. rates</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.2</td>
<td>$6.47 \times 10^{-2}$</td>
<td>-</td>
</tr>
<tr>
<td>0.1</td>
<td>$8.80 \times 10^{-3}$</td>
<td>2.88</td>
</tr>
<tr>
<td>0.05</td>
<td>$2.20 \times 10^{-3}$</td>
<td>2.02</td>
</tr>
<tr>
<td>0.025</td>
<td>$5.36 \times 10^{-4}$</td>
<td>2.02</td>
</tr>
<tr>
<td>0.0125</td>
<td>$1.34 \times 10^{-4}$</td>
<td>1.99</td>
</tr>
<tr>
<td>0.00625</td>
<td>$3.38 \times 10^{-5}$</td>
<td>1.99</td>
</tr>
</tbody>
</table>

with

$$T(x, y) = \frac{(-y/b^2, x/a^2)}{\sqrt{y^2/b^4 + x^2/a^4}}.$$ 

For an initial solution $u(s, 0) = \sin^3(2\pi s/L)$, the exact solution for all times $t$ is

$$u(s, t) = \sin^3(2\pi (s - t)/L),$$

where $L$ is the length of the perimeter of the ellipse. Table 6.4 shows the error for the final time $t = 1$ and the estimated order of convergence of the method for various grid spacings. Due to the generic centered nature of the RBF-FD stencils used for approximating the first order derivatives, the TVD-RK3 scheme [41] is chosen for the time discretization with a time step-size $\Delta t = 0.5\Delta x$. In this example, the bandwidth is $\gamma = (\sqrt{2} + \sqrt{2}/2)\Delta x$ with $m = 9$ points in the finite difference stencil. Second-order convergence is observed.

### 6.2.3 Advection-diffusion equation on an ellipse

Our next example is the advection-diffusion equation on an ellipse. We are interested in the second-order approximation of the first and second derivatives using RBF-FD stencils for a computational tube radius induced by the second derivatives. The equation

$$u_t + u_s = u_{ss},$$

with $s$ being the arclength, is imposed on an ellipse with major axis $b = 1.25$ along the $y$-axis and minor axis $a = 0.75$ along the $x$-axis. Similar to the previous example, application of the closest point principles to the surface PDE gives

$$u_t + T(x, y) \cdot \nabla u = \Delta u$$

with

$$T(x, y) = \frac{(-y/b^2, x/a^2)}{\sqrt{y^2/b^4 + x^2/a^4}}.$$
Table 6.5: Relative error of the approximate solution at time $t = 1$ for the advection-diffusion equation on an ellipse as measured in the infinity norm and the convergence rates.

<table>
<thead>
<tr>
<th>$\Delta x$</th>
<th>Rel. error ($t = 1$)</th>
<th>Conv. rates</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.2</td>
<td>$7.20 \times 10^{-3}$</td>
<td>-</td>
</tr>
<tr>
<td>0.1</td>
<td>$1.70 \times 10^{-3}$</td>
<td>2.11</td>
</tr>
<tr>
<td>0.05</td>
<td>$3.76 \times 10^{-4}$</td>
<td>2.14</td>
</tr>
<tr>
<td>0.025</td>
<td>$9.25 \times 10^{-5}$</td>
<td>2.02</td>
</tr>
<tr>
<td>0.0125</td>
<td>$1.83 \times 10^{-5}$</td>
<td>2.33</td>
</tr>
<tr>
<td>0.00625</td>
<td>$5.13 \times 10^{-6}$</td>
<td>1.84</td>
</tr>
<tr>
<td>0.003125</td>
<td>$1.12 \times 10^{-6}$</td>
<td>2.19</td>
</tr>
</tbody>
</table>

Table 6.6: Relative error of the approximate solution at time $t = 1$ for the heat equation on the unit sphere as measured in the infinity norm and the convergence rates.

<table>
<thead>
<tr>
<th>$\Delta x$</th>
<th>Rel. error ($t = 1$)</th>
<th>Conv. rates</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.2</td>
<td>$5.40 \times 10^{-3}$</td>
<td>-</td>
</tr>
<tr>
<td>0.1</td>
<td>$1.40 \times 10^{-3}$</td>
<td>1.95</td>
</tr>
<tr>
<td>0.05</td>
<td>$3.79 \times 10^{-4}$</td>
<td>1.89</td>
</tr>
<tr>
<td>0.025</td>
<td>$9.60 \times 10^{-5}$</td>
<td>1.98</td>
</tr>
<tr>
<td>0.0125</td>
<td>$2.39 \times 10^{-5}$</td>
<td>2.00</td>
</tr>
</tbody>
</table>

For an initial solution $u(s,0) = \sin(2\pi s/L)$, the exact solution has the form

$$u(s,t) = e^{-2\pi t/L} \sin(2\pi(s-t)/L),$$

where $L$ is the length of the perimeter of the ellipse. Table 6.5 shows the relative error of the approximate solution compared to the exact as well as the estimated order of convergence. A second-order of convergence is observed.

6.2.4 Heat equation on a sphere

For our first three dimensional example, we are interested in the second-order approximation of the Laplace-Beltrami operator using RBF-FD stencils. Consider the heat equation

$$u_t = \Delta_{\Gamma} u$$

on the unit sphere $\Gamma$. For an initial profile $u(\theta, \phi, 0) = \sin \phi$, the exact solution for all times $t$ is

$$u(\theta, \phi, t) = e^{-2t} \sin \phi.$$

Table 6.6 shows the relative errors as well as the convergence rates for different grid sizes $\Delta x$. Second-order convergence is observed.
Table 6.7: Relative error of the approximate solution at time $t = 1$ for the advection equation on a torus as measured in the infinity norm and the convergence rates.

<table>
<thead>
<tr>
<th>$\Delta x$</th>
<th>Rel. error ($t = 1$)</th>
<th>Conv. rates</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>1.74 x 10^{-2}</td>
<td>-</td>
</tr>
<tr>
<td>0.05</td>
<td>2.90 x 10^{-3}</td>
<td>2.59</td>
</tr>
<tr>
<td>0.025</td>
<td>4.81 x 10^{-4}</td>
<td>2.59</td>
</tr>
<tr>
<td>0.0125</td>
<td>9.40 x 10^{-5}</td>
<td>2.36</td>
</tr>
<tr>
<td>0.00625</td>
<td>1.78 x 10^{-5}</td>
<td>2.40</td>
</tr>
</tbody>
</table>

### 6.2.5 Advection on a torus

In this example, we are interested in the second-order approximation of the first surface derivatives using RBF-FD stencils. Consider the advection equation on a torus. Following [42], for a torus defined as

$$x(\phi, \theta) = \left( \left( \frac{1}{2} \cos(\phi) + 1 \right) \cos(\theta), \left( \frac{1}{2} \cos(\phi) + 1 \right) \sin(\theta), \frac{1}{2} \sin(\phi) \right)$$

with $-\pi \leq \theta, \phi < \pi$, the advection equation is given by

$$u_t + u_\phi = 0.$$

By considering an initial solution of the form

$$u(\phi, \theta, 0) = f(\phi) = \begin{cases} g\left(\frac{2 + \pi}{\pi}\right), & -\pi \leq \phi \leq 0; \\ g\left(\frac{\pi - \phi}{\pi}\right), & 0 < \phi < \pi. \end{cases}$$

where

$$g(x) = \frac{e^{1/(x-1)} - e^{-1/x}}{e^{-1/x} + e^{1/(x-1)}},$$

the exact solution at time $t$ is

$$u(\phi, \theta, t) = f(\phi - t).$$

In this example, the computational tube radius is $\gamma = (2 + \sqrt{3}/2)\Delta x$ and the stencil uses $m = 33$ points. The stable TVD-RK3 scheme is chosen for the discretization in time with step-size $\Delta t = 0.5\Delta x$. Table 6.7 shows the error and the convergence rate of the method for various grid sizes $\Delta x$. Second-order of convergence is observed.

### 6.2.6 Image denoising on a sphere

Our next example employs image denoising on a texture image on the unit sphere. Following [13], the Perona-Malik equation on a surface $\Gamma$ is used to denoise images on surfaces. The equation has the form

$$u_t = \nabla_\Gamma \cdot (g(|\nabla_\Gamma u|)\nabla_\Gamma u)$$
where $g$ is the diffusion coefficient given by

$$g(s) = \frac{1}{1 + (s/\lambda)^2}$$

and $\lambda$ is a coefficient. The parameter $\lambda$ and the final time $t$ of the computation control the denoising of an image.

For our example, we consider the image of two birds and we add Gaussian noise with zero mean and 0.2 standard deviation to the image of the two birds. The noisy image is scaled to the interval $[0, 1]$. Figure 6.5 shows the results after denoising the image, using $\lambda = 5$ and running the code for 120 time steps with $\Delta t = 0.2\Delta x^2$, where the grid size $\Delta x = 0.005$. The choice of a small parameter $\lambda$ preserves edges in the denoised image. The computational tube radius is chosen to be $\gamma = (2 + \sqrt{3}/2)\Delta x$ with $m = 33$ points in the stencil. The number of points in the image is 1280598.

### 6.2.7 Reaction-diffusion systems

Our final example considers the Gray-Scott reaction-diffusion model on surfaces. Following [43], the model describes the chemical reaction

$$U + 2V \rightarrow 3V$$

$$V \rightarrow P,$$

where $U$, $V$ and $P$ are chemicals. The surface model has the form

$$u_t = F(1 - u) - uv^2 + D_u \Delta_S u$$

$$v_t = -(F + k)v + uv^2 + D_v \Delta_S v$$

where $u$, $v$ are the concentrations of the chemicals, $D_u$, $D_v$ are the diffusion rates, $k$ is the conversion rate from $V$ to $P$ and $F$ is the feed rate of $U$.

For parameter choices of $D_u = 5 \times 10^{-5}$ and $D_v = 2.5 \times 10^{-5}$ and different values for $F$ and $k$ a variety of patterns are observed. Figure 6.6 shows two of these patterns on the surface of the Stanford Bunny [78]. The final time is 15000 and the time step-size is $\Delta t = (0.1/D_u)\Delta x^2$ for a spatial grid size of $\Delta x = 0.025$. The patterns in Figure 6.6 are similar to the plane patterns formed in [43].
Figure 6.5: The initial image (top) is warped on a sphere and Gaussian noise is added. The noisy image (left) and the denoised image (right) are shown after 120 iterations on 1280598 points.
Figure 6.6: The solution of the Gray-Scott reaction-diffusion model for parameters \((k, F) = (0.06, 0.037)\) (left) and \((k, F) = (0.062, 0.03)\) (right).
Chapter 7

An implicit RBF-FD discretization of the closest point method

In this chapter, we provide an implicit formulation of the closest point method. Our approach employs an RBF-FD discretization and a least squares method to obtain stable, accurate results. We test the method on a variety of PDEs on static surfaces using implicit and implicit-explicit time stepping methods.

In addition, we perform a simple coupling of our method with the grid based particle method (see Section 4.2) to approximate the solution of PDEs on moving surfaces. Numerical experiments show the convergence of the coupled method for PDE systems, including the approximation of the solution of the strongly coupled Cahn-Hilliard PDE on a moving ellipsoid.

7.1 An implicit formulation of the closest point method using RBF-FD

In Section 6.1, an RBF-FD formulation of the explicit closest point method is presented. Numerical results show the potential of this method. The advantage of this method is the reduction of the size of the computational tube by 25% relative to the standard finite difference implementation of the closest point method for the case of the Laplace-Beltrami operator. In this section, we derive an implicit formulation of the closest point method using RBF-FD.

For illustration purposes, consider the heat equation on a surface $\Gamma$,

$$ u_t = \Delta_\Gamma u. $$  \hspace{1cm} (7.1)
An RBF-FD discretization for the Laplace-Beltrami operator (see Section 6.1), yields the ODE system

\[ \dot{U} \approx \Delta_h^{RBF} U, \]  

(7.2)

where \( U \) is the semi-discretized solution in space.

To obtain an improvement in efficiency over our explicit methods (see Section 6.1), we considered implicit time stepping methods. Unfortunately, these did not yield an improvement in the stability time stepping restriction. We also tried to stabilize (7.2) by enforcing the constant extension of the solution in the normal direction in a single equation, leading to the approximation

\[ \dot{U} \approx \Delta_h^{RBF} U - c(U - EU), \]

where \( c \) is a positive constant and \( E \) is the extension matrix defined in Equation (6.1) (cf. [56, 80]). There are several \( c \) values that stabilize the system above, i.e. \( c \approx \max_j |\lambda_j| \), where \( \lambda_j \) are the eigenvalues of the RBF-FD approximation of \( \Delta_h \). This approach can be efficient, however we are interested in developing an implicit RBF-CPM that can be applied on computational tubes with holes.

Inspired by [68], we propose an alternative approach for stabilizing the method. Specifically, we enforce the solution to be constant in the normal direction using an extra equation, \( u = u(cp) \). The semi-discrete system of equations becomes

\[
\begin{align*}
\dot{U} &\approx \Delta_h^{RBF} U, \\
U &\approx EU.
\end{align*}
\]  

(7.3)

An implicit time stepping method is employed leading to an over-determined system. For example, choosing a backward Euler time stepping method, the system takes the form

\[
\begin{align*}
U^{n+1} &= EU^n + \Delta t \Delta_h^{RBF} U^{n+1}, \\
U^{n+1} &= EU^{n+1},
\end{align*}
\]  

(7.4)

where \( U \) is the discretized approximate solution \( u \). The system (7.4) is solved using the least squares method, and was found to be unconditionally stable in practice for the problems considered in this thesis.

The use of the least squares method also provides flexibility in the selection of computational tubes around the surface. For example, a computational tube might not contain a closest point representation of all the grid points that lie within the tube radius. We refer to this type of tubes as computational tubes with holes. Our numerical examples show that the proposed stabilization is convergent for computational tubes of this form. Further analysis is required in order to explore the extent of the flexibility in the choices of the computational tube that the least squares method provides.
7.2 Numerical experiments on static surfaces

In this section, we present numerical experiments for the solution of PDEs on static surfaces. For the solution of the sparse matrices least squares systems, we use the MATLAB code Factorize\(^1\) \cite{16} which is a powerful tool for solving sparse matrix systems for any matrix sparsity pattern that appears in our numerical examples. Unless stated otherwise, a computational tube of radius \(\gamma = 2 + \sqrt{2}/2\) is chosen with \(m = 13\) points in the RBF finite difference stencil in two dimensions and \(\gamma = \sqrt{5} + \sqrt{3}/2\) with \(m = 57\) points in three dimensions.

A sample MATLAB implementation of the implicit RBF-FD formulation of the closest point method appears in Appendix B.2.

7.2.1 Heat equation on a circle

Our first example considers the heat equation on the unit circle. We are interested in applying large time step-sizes for the approximation of the Laplace-Beltrami operator using the implicit RBF-closest point method. The equation

\[ u_t = \Delta_G u \]

is solved using \(u(\theta, 0) = \sin \theta\) as an initial condition. The exact solution at all times \(t > 0\) is

\[ u(\theta, t) = e^{-t} \sin \theta. \]

The second-order backward differentiation formula (BDF2) is chosen as the time discretization. This yields the over-determined system

\[
\begin{cases}
U^{n+1} &= \frac{4}{3} EU^n - \frac{1}{3} EU^{n-1} + \frac{2}{3} \Delta t \Delta_h^{RBF} U^{n+1}, \\
U^{n+1} &= EU^{n+1},
\end{cases}
\]

(7.5)

where \(\Delta_h^{RBF}\) is the RBF-FD discretization of the Laplace-Beltrami operator (see Section 6.1), \(U\) is the discretized solution and \(E\) is the closest point extension matrix that enforces the solution \(U^{n+1}\) to be constant in the normal direction. To obtain \(U^1\), we use backward Euler.

Table 7.1 shows the error of the approximate solution compared to the exact in the \(l_\infty\)-norm at time \(t = 1\) using \(\Delta t = 0.5\Delta x\). A clear second-order convergence in \(\Delta x\) is observed.

\(^1\)Available online at http://www.mathworks.com/matlabcentral/fileexchange/24119-don-t-let-that-inv-go-past-your-eyes-to-solve-that-system-factorize-
<table>
<thead>
<tr>
<th>$\Delta x$</th>
<th>Rel. error ($t = 1$)</th>
<th>Conv. rates</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.2</td>
<td>$2.08 \times 10^{-2}$</td>
<td>-</td>
</tr>
<tr>
<td>0.1</td>
<td>$8.50 \times 10^{-3}$</td>
<td>1.30</td>
</tr>
<tr>
<td>0.05</td>
<td>$2.70 \times 10^{-3}$</td>
<td>1.63</td>
</tr>
<tr>
<td>0.025</td>
<td>$6.68 \times 10^{-4}$</td>
<td>2.03</td>
</tr>
<tr>
<td>0.0125</td>
<td>$1.64 \times 10^{-4}$</td>
<td>2.03</td>
</tr>
<tr>
<td>0.00625</td>
<td>$3.50 \times 10^{-5}$</td>
<td>2.23</td>
</tr>
</tbody>
</table>

Table 7.1: Relative error of the approximate solution at time $t = 1$ for the heat equation on the unit circle as measured in the infinity norm and the convergence rates.

Figure 7.1: The mean value of the relative error as measured in the $\infty$-norm for various spatial step sizes $\Delta x$.

The implicit RBF-closest point method can be applied on different computational tubes including tubes with holes that arise from the application of the grid based particle method’s resampling step (see Section 4.2.1). To demonstrate the convergence of the method for computational tubes with holes, we randomly remove 1% of the points in the computational tube and apply the method to the heat equation. For each spatial discretization level $\Delta x$, 50 experiments are carried out by removing 1% of the grid points within the computational tube along with corresponding closest points on the surface. The mean value of the relative error over the 50 experiments for different grid spacings $\Delta x$ is shown in Figure 7.1. Second-order of convergence is observed for the mean value of the relative error without any changes in the choice of the computational tube radius and the number of points in the PBF-FD stencil.

### 7.2.2 Heat equation on a sphere

In this example, large time step-sizes for the approximation of the Laplace-Beltrami operator are applied using the implicit RBF-closest point method. Consider the heat equation on
Table 7.2: Relative error of the approximate solution at time $t = 1$ for the heat equation on the unit sphere as measured in the infinity norm and the convergence rates.

<table>
<thead>
<tr>
<th>$\Delta x$</th>
<th>Rel. error ($t = 1$)</th>
<th>Conv. rates</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.2</td>
<td>$6.53 \times 10^{-2}$</td>
<td>-</td>
</tr>
<tr>
<td>0.1</td>
<td>$2.75 \times 10^{-2}$</td>
<td>1.25</td>
</tr>
<tr>
<td>0.05</td>
<td>$7.40 \times 10^{-3}$</td>
<td>1.89</td>
</tr>
<tr>
<td>0.025</td>
<td>$2.20 \times 10^{-3}$</td>
<td>1.78</td>
</tr>
<tr>
<td>0.0125</td>
<td>$5.95 \times 10^{-4}$</td>
<td>1.86</td>
</tr>
</tbody>
</table>

the unit sphere. For the parametrization of the sphere

$$\mathbf{x}(\theta, \phi) = (\cos \theta \cos \phi, \sin \theta \cos \phi, \sin \phi)$$

and an initial profile

$$u(\theta, \phi, 0) = \sin \phi$$

the exact solution for all times $t > 0$ is given as

$$u(\theta, \phi, t) = e^{-2t} \sin \phi.$$

Using the second-order backward differentiation formula (BDF2) described in Equation (7.5) and $\Delta t = 0.5 \Delta x$, the $l_{\infty}$-error of the approximation at time $t = 1$ is shown in Table 7.2. The method is convergent and the convergence rate is close to 2. Based on Figure 6.2, the convergence rate is expected to be second-order.

Similar to the previous example, we randomly remove 1% of the points in the computational tube and approximate the solution of the heat equation. The mean value of the relative error of each experiment for different spatial step sizes $\Delta x$ is shown in Figure 7.2. Second-order of convergence is observed for the mean value of the relative error without any changes in the choice of the computational tube radius and the number of points in the PBF-FD stencil.

### 7.2.3 Reaction-diffusion systems

This example considers a reaction-diffusion system on a bumpy torus [1]. We use large time step-sizes by employing our implicit RBF-closest point method to reach the steady state solution of the system. The system has the form

$$
\begin{align*}
\frac{u_t}{\mu} &= \nu \Delta u + f(u, v) \\
\frac{v_t}{\mu} &= \mu \Delta v + g(u, v)
\end{align*}
$$

(7.6)

with

$$f(u, v) = a - (b + 1)u + u^2v, \quad g(u, v) = bu - u^2v,$$
Figure 7.2: The mean value of the relative error as measured in the $\infty$-norm for various spatial step sizes $\Delta x$.

where $a, b, \nu$ and $\mu$ are constants. This reaction-diffusion system is called the Brusselator [83]. The use of the SBDF2 scheme is recommended for the solution of reaction-diffusion equations when second-order centered finite difference schemes are applied to the diffusion operator [69]. Applying the SBDF2 scheme (see Section 2.5.3), yields

$$
\begin{aligned}
\frac{1}{2\Delta t} (3U^{n+1} - 4EU^n + EU^{n-1}) &= \nu \Delta_h^{RBF} U^{n+1} + 2Ef(U^n, V^n) - Ef(U^{n-1}, V^{n-1}), \\
\frac{1}{2\Delta t} (3V^{n+1} - 4EV^n + EV^{n-1}) &= \mu \Delta_h^{RBF} V^{n+1} + 2Ef(U^n, V^n) - Ef(U^{n-1}, V^{n-1}), \\
U^{n+1} &= EU^{n+1}, \\
V^{n+1} &= EV^{n+1},
\end{aligned}
$$

where $\Delta_h^{RBF}$ is the RBF-FD discretization of the Laplace-Beltrami operator (see Section 6.1) and $U, V$ are the discretized solutions. $E$ is the closest point extension matrix that enforces the solutions $U^{n+1}, V^{n+1}$ to be constant in the normal direction. An iteration of the implicit-explicit Euler discretization (see Section 2.5.3) is applied to initialize the system.

The approximation $U$ is shown in Figure 7.3. In this numerical experiment, a random selection of 1% of points in the computational tube is removed.

7.2.4 Cahn-Hilliard on a sphere

The final example of PDEs on static surfaces considers the Cahn-Hilliard equation on the unit sphere. The Cahn-Hilliard PDE contains a stiff biharmonic operator which is treated implicitly in time. Therefore, large time step-sizes can be applied for the approximation of
the solution of the PDE. The equation has the form

\[ u_t - \frac{1}{Pe} \nabla \Gamma \cdot \left( \nu \nabla \Gamma \left( \frac{\partial g}{\partial u} \right) \right) + \frac{Cn^2}{Pe} \nabla \Gamma \cdot \left( \nu \nabla \Delta \Gamma u \right) = 0, \]

where \( Cn \) is the Cahn number, \( Pe \) is the surface Peclet number, \( \nu \) is the mobility and \( g \) is a double well potential. This formulation of the Cahn-Hilliard equation appears in [39].

Considering the mobility \( \nu \) to be constant, the biharmonic operator intrinsic to the surface is computed by applying the Laplace-Beltrami operator twice, i.e.

\[ \nabla \Gamma \cdot \left( \nu \nabla \Delta \Gamma u \right) = \nu \Delta \Gamma \Delta \Gamma u. \]

To complete the discretization, we apply the IMEX Euler method with an implicit treatment of the biharmonic operator. Specifically,

\[
\begin{cases}
U^{n+1} + \Delta t \frac{Cn^2}{Pe} \nu \Delta_h^{RBF} \Delta_h^{RBF} U^{n+1} = EU^n + \Delta t \frac{\nu}{Pe} \Delta_h^{RBF} \frac{\partial g}{\partial u}(U^n), \\
U^{n+1} = EU^{n+1},
\end{cases}
\tag{7.7}
\]

where \( \Delta_h^{RBF} \) is the discretized Laplace-Beltrami operator using RBF-FD (see Section 6.1) and \( E \) is the closest point extension matrix.

Consider the unit sphere with an initial profile \( u(x,0) \) defined as a random small perturbation around 0.5. For the parameter choice of \( \nu = 1, Pe = 1, Cn = 0.06 \) and a double well potential function \( g(u) = u^2(1 - u)^2 \), the experimental results for a spatial grid size \( \Delta x = 0.03536 \) and for a time step-size \( \Delta t = 0.001 \) are shown in Figure 7.4. As expected, the initially randomly distributed low and high concentrations on the surface tend to separate in two distinct areas. The mixing is fast initially and it gets slower as time evolves.
Figure 7.4: The solution of the Cahn-Hilliard equation on the unit sphere for various times $t$. Yellow and blue colors correspond to high and low concentrations respectively.
7.3 A coupled method

To solve PDEs on moving surfaces, the implicit RBF-FD formulation of the closest point method is coupled with the grid based particle method described in Section 4.2. Numerical examples showed the potential of the implicit RBF-closest point method to be applied on computational tube with holes that arise in the original grid based particle method.

First, the initialization step of the closest point method takes place, using a closest point representation of the surface over a computational tube of radius $\gamma$. The initial solution is extended using the closest point representation $u(x) = u(cp(x), 0)$.

After the initialization step, for each time step of size $\Delta t$, two steps are alternated:

1. **Closest point method evolution**: The PDE system is evolved by applying the implicit RBF-FD discretization of the closest point method using a time step-size $\Delta t$ and a tube radius $\gamma$. (See Section 7.1 for details.)

2. **GBPM evolution**: Steps 1-3 of the GBPM are applied for a time step-size $\Delta t$ and a computational tube radius $\gamma$. The GBPM evolution provides a closest point representation of the time-evolved surface over a tube of radius $\gamma$. (See Section 4.2 for details on the GBPM.)

Conceptually, the calculation of the computational tube radius $\gamma$ is similar to Section 5.3. Using Table 6.2, for a choice of computational tube radius $\bar{\gamma}$ and $m$ points on the RBF-FD stencil, a safe choice for the tube radius is

$$\gamma = \bar{\gamma} + \Delta t \cdot v^{\text{max}}_n$$  \hspace{1cm} (7.8)

where $v^{\text{max}}_n$ is an upper bound on the speed of a footpoint in the normal direction at time step $n$. In practice, we select $\gamma$ adaptively by setting $\gamma = \bar{\gamma}$ and checking for violations of the tube radius. A larger computational tube is applied in an event of the tube violation, while keeping the same number of points $m$ in the RBF-FD stencil.

For the advection-diffusion equation (3.5) on a moving surface, the implicit RBF-closest point method coupled with the grid based particle method consists of the following algorithmic steps:

- Construct a grid $\{z_j\}$ that contains the initial surface $\Gamma_0$ and compute a closest point representation of the surface $cp_{\Gamma_0}$ to map all the grid points $\{z_j\}$ to their closest points $\{x_j\}$ on the surface. A computational tube $S_{\Gamma_0}$ is constructed with $\gamma = \gamma_{CPM}$.
- For each iteration $n = 1, \ldots, N$ of time step-size $\Delta t$:
  - Apply the resampling step of the GBPM as described in Section 4.2.1. The velocity $v^n_j$ is found for each new footpoint.
Update the computational tube step of the GBPM (see Section 4.2). In particular:

1. The activation phase is applied: all the grid points neighboring to active grid points are activated and the resampling step is applied. The velocity of each footpoint $v_{nj}$ is calculated.

2. Check whether the maximum velocity $\max_j(v_{nj})\Delta t$ exceeds the computational tube radius $\gamma$. In such occurrence, update the tube radius $\gamma$ using Equation (5.1) and repeat step 1, otherwise continue.

The deactivation phase using the tube radius $\gamma$ is applied. The new computational tube $S^\gamma_n$ is constructed.

- A closest point extension step is applied to extend the solution to the computational tube $S^\gamma_n$.
- The finite differences matrices of the Laplacian $\Delta_h$ and the first derivatives $D_x$ and $D_y$ using centered schemes are calculated using RBF-GA algorithm.
- Solve the PDE for one time step using either the explicit or the implicit closest point method.
- Restore the value of $\gamma$ to its initial value $\gamma_{CPM}$ to ensure that a computational tube of minimum size is constructed at each time step.

Similarly to the first coupled method, an initial step of the modified GBPM is applied before the application of the CPM for the calculation of the mean curvature.

Our coupled method uses the same time step-size for both the evolution of the surface and the solution of the PDE. Even though we use the implicit RBF-closest point method which appeared to be unconditionally stable in practice, we follow the stability constraints introduced by the grid based particle method. Further analysis needs to be done in order to avoid stability constraints that arise from the grid based particle method.

### 7.4 Numerical experiments on moving surfaces

In this section, we present our numerical experiments for the solution of PDEs on moving surfaces. For the solution of the sparse matrices least squares systems, we use the MATLAB code Factorize [16]. Unless stated otherwise, we approximate the solution of the advection-diffusion PDE (3.5) with diffusivity parameter $D = 1$. Due to our use of a closest point representation, the normal derivative vanishes in equation (3.5), $\partial u/\partial n = 0$. We use the implicit formulation of the closest point method with RBF-FDs as defined in Section 7.1.
Figure 7.5: The relative error of the numerical solution of the diffusion model on the expanding circle over time. Errors are computed using the analytical solution.

with an implicit-explicit Euler scheme in time (see Section 2.5.3). Specifically,

$$
\begin{align*}
U^{n+1} - \Delta t \Delta_h^{RBF} U^{n+1} &= EU^n + \Delta t (Ef(U^n) - EV^n \kappa^n U^n + \nabla_h^{RBF} \cdot (U^n T^n)), \\
U^{n+1} &= EU^{n+1},
\end{align*}
$$

(7.9)

where \(U\) is the discretized solution, \(E\) is the closest point extension matrix and \(\Delta_h^{RBF}\) and \(\nabla_h^{RBF}\) are the RBF-FD discretizations of the Laplace-Beltrami and the divergence operators (see Section 6.1) respectively. As before, the second equation imposes the constant normal extension of the approximation \(U^{n+1}\) at time \(t_{n+1}\).

### 7.4.1 Diffusion on an expanding circle

In our first example, we investigate the convergence of our method in two dimensions. Following [67], consider the homogenous PDE (3.5) on a circle centered at the origin with radius \(r_0 = 0.75\). The velocity of the circle is constant in the normal direction, \(v = 5n\). The exact solution has the form

$$u(\theta, t) = e^{4/(5r(t))} \frac{\cos \theta \sin \theta}{r(t)}$$

where \(r(t) = r_0 + 5t\) is the exact radius of the circle at time \(t\).

Using the discretization (7.9), Figure 7.5 shows the relative error of the approximate solution against the analytical solution over time for different grid sizes \(\Delta x\). Table 7.3 shows the relative error and the convergence rate for selected times and grid sizes. Second-order convergence is observed for a time step-size of \(\Delta t = 4\Delta x^2\).
Using a uniform grid of $\Delta x = 0.025$, the total computational time for our coupled method’s algorithm after 5 time steps is 26.4 seconds on a 4-core laptop in MATLAB. The total time of GBPM algorithm alone after 5 time steps is 17.64 seconds using a parallel implementation. The total time of the calculation of the RBF-FD weights alone using the RBF-GA algorithm after 5 time steps is 8.08 seconds. The surface is sampled by 1,064 points at the completion of the 5th time step. The computational time for the calculation of the solution of the linear over-determined matrix system using Factorize is negligible using the implicit RBF-closest point method.

### 7.4.2 Advection-diffusion on an oscillating sphere

Next, consider the advection-diffusion equation on an oscillating sphere. We are interested in the convergence of our coupled method in three dimensions. Following the example in [67], consider the solution of the non-homogenous PDE (3.5) on the unit sphere. The velocity of the sphere is defined purely on the x-axis as

$$v = \frac{a'(t)}{2a(t)}(x_1, 0, 0),$$

where

$$a(t) = 1 + \sin(2t).$$

The exact solution of the evolution of the surface is given by

$$x(\theta, \phi, t) = (\sqrt{a(t)} \cos(\theta) \cos(\phi), \sin(\theta) \cos(\phi), \sin(\phi))$$

where $\theta$ is the azimuth and $\phi$ is the elevation of the sphere. The forcing term of the PDE chosen is given by

$$f = u(x, t) \cdot \left( -6 + \frac{a'(t)}{a(t)} \left( 1 - \frac{x_1^2}{2N} \right) + \frac{1 + 5a(t) + 2a^2(t)}{N} - \frac{1 + a(t)}{N^2} (x_1^2 + a^3(t)(x_2^2 + x_3^2)) \right).$$

<table>
<thead>
<tr>
<th>$\Delta x$</th>
<th>$t = 0.04$ e.o.c.</th>
<th>$t = 0.08$ e.o.c.</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>9.96x10^-2</td>
<td>1.55x10^-1</td>
</tr>
<tr>
<td>0.05</td>
<td>2.27x10^-2</td>
<td>3.63x10^-2</td>
</tr>
<tr>
<td>0.025</td>
<td>5.60x10^-3</td>
<td>9.00x10^-3</td>
</tr>
<tr>
<td>0.0125</td>
<td>1.40x10^-3</td>
<td>2.30x10^-3</td>
</tr>
<tr>
<td>0.00625</td>
<td>3.51x10^-4</td>
<td>5.63x10^-4</td>
</tr>
<tr>
<td>0.003125</td>
<td>8.75x10^-5</td>
<td>1.41x10^-4</td>
</tr>
</tbody>
</table>

Table 7.3: Relative errors as measured in the infinity norm and the estimated order of convergence (e.o.c.) at various times $t$.


Figure 7.6: The relative error of the numerical solution of the diffusion model on the oscillating sphere over time. Errors are computed using the analytical solution.

<table>
<thead>
<tr>
<th>$\Delta x$</th>
<th>$t = 0.08$</th>
<th>e.o.c.</th>
<th>$t = 0.16$</th>
<th>e.o.c.</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.2</td>
<td>$8.89 \times 10^{-2}$</td>
<td>-</td>
<td>$1.76 \times 10^{-1}$</td>
<td>-</td>
</tr>
<tr>
<td>0.1</td>
<td>$2.6 \times 10^{-2}$</td>
<td>1.77</td>
<td>$4.91 \times 10^{-2}$</td>
<td>1.84</td>
</tr>
<tr>
<td>0.05</td>
<td>$6.80 \times 10^{-3}$</td>
<td>1.94</td>
<td>$1.26 \times 10^{-2}$</td>
<td>1.96</td>
</tr>
<tr>
<td>0.025</td>
<td>$1.70 \times 10^{-3}$</td>
<td>1.99</td>
<td>$3.20 \times 10^{-3}$</td>
<td>1.99</td>
</tr>
</tbody>
</table>

Table 7.4: Relative errors as measured in the infinity norm and the estimated order of convergence (e.o.c.) at various times $t$.

with

$$N = x_1^2 + a^2(t)(x_2^2 + x_3^2).$$

The solution of the PDE for all times $t > 0$ is given by

$$u(x, t) = e^{-6t}x_1x_2.$$
tational time for the calculation of the solution of the over-determined matrix system alone using Factorize is 1.5 seconds.

7.4.3 A cross-diffusion reaction-diffusion system

This example considers a reaction-diffusion system with cross-diffusion terms [59]. The system of equations has the form

\[
\frac{D u}{D t} = \Delta u + d_u \Delta w - u \nabla \cdot v + f_1(u, w),
\]

\[
\frac{D w}{D t} = D \Delta w + d_u \Delta u - w \nabla \cdot v + f_2(u, w)
\]

where \( D, d_u, d_w > 0 \) are scalar quantities. The coupling functions that we consider are

\[
f_1(u, w) = 200(0.1 - u + u^2w),
\]

\[
f_2(u, w) = 200(0.9 - u^2w).
\]

The velocity considered is purely in the normal direction and is given as

\[
v = (0.01\kappa + 0.4u)n
\]

where \( \kappa \) is the mean curvature and \( n \) is the unit normal vector. In our experiment, we consider a small random perturbation around 0.5 on a torus. The solution is expected to develop different patterns on the surface for different parameter choices [59]. Figure 7.7 shows the results for selected times for \( D = 10 \) and \( d_u = d_w = 1 \). An implicit-explicit scheme (IMEX) is chosen in time, specifically,

\[
\begin{aligned}
U^{n+1} - \Delta t \Delta_h^{RBF} U^{n+1} &= EU^n + \Delta t(E f_1(U^n, W^n) - EV^n \kappa^n U^n - \nabla_h^{RBF} \cdot (U^n T^n) + d_w \Delta_h^{RBF} W^n), \\
W^{n+1} - \Delta t \Delta_h^{RBF} W^{n+1} &= EW^n + \Delta t(E f_2(U^n, W^n) - EV^n \kappa^n W^n - \nabla_h^{RBF} \cdot (W^n T^n) + d_u \Delta_h^{RBF} U^n), \\
U^{n+1} &= EU^{n+1} \\
W^{n+1} &= EW^{n+1}
\end{aligned}
\]

with time step-size \( \Delta t = 0.2\Delta x^2 \) and spatial grid size \( \Delta x = 0.05 \).

Note that the patterns that the solution \( u \) develops on the torus are similar to the patterns that appears in [59] for the chosen parameters.
Figure 7.7: A visualization of the numerical solution $u$ of the cross-diffusion reaction-diffusion system (7.10) at selected times $t$. 
Figure 7.8: A visualization of the numerical solution \( u \) of the Cahn-Hilliard PDE (3.7) at selected times \( t \).

### 7.4.4 Cahn-Hilliard equation on an ellipsoid

Our final example considers the homogenous Cahn-Hilliard PDE (3.7). An initial ellipsoid centered at the origin with axis \( r_x = 1 \), \( r_y = 0.8 \) and \( r_z = 0.5 \) is evolved in the normal direction according to the velocity

\[
v = (0.01\kappa + 0.4u)n
\]

where \( \kappa \) is the mean curvature and \( n \) is the unit normal vector pointing outwards. A double well potential of the form

\[
\psi(t) = t^2(1-t)^2
\]

is chosen and the parameter values are \( \mu = 0.03^2 \) and \( D = 1 \). The discretization of the Cahn-Hilliard PDE is carried in the same way as in Equation (7.7). For an initial profile \( u \) given as a small random perturbation around 0.5, we obtain the numerical results displayed in Figure 7.8. The spatial step-size is \( \Delta x = 0.05 \) with a time step-size \( \Delta t = 0.0005 \). We observe that the surface grows outwards at the high concentration areas of the solution \( u \).
Chapter 8

Conclusions

The closest point method is an embedding method for solving PDEs on static surfaces. The goal of this thesis is to extend the closest point method to approximate the solution of PDEs on moving surfaces. A motion tracking method is required to capture the motion of the surface. The grid based particle method is chosen, which provides a closest point representation of the surface at each time step. However, some difficulties prevent the natural coupling of the two methods. The main adversity arises in the closest point mapping that the grid based particle method provides at each time step. Specifically, grid point deactivation may occur leading to a computational tube with holes.

In order to overcome this difficulty, we use two different approaches. In the first approach, a modification is introduced in the grid based particle method to map the deactivated points to their closest points on the surface using locally defined circles for curves and spheres for surfaces. Numerical results display the potential of this modification in tracking the motion of a surface. The coupled method obtained by combining the closest point method and the modified grid based particle method provides a second-order approximation of the solution of the conservation law described in Section 3.2 for $\Delta t = O(\Delta x^2)$ and for geometric motions in the normal and tangent direction. Because the deactivation of grid points is inevitable in certain situations (e.g., topological change in 3D example in Section 5.2) the method is suitable for solving PDEs on smooth moving surfaces.

We also propose to overcome the difficulty that arises from holes in the computational tube by introducing an explicit formulation of the closest point method using finite differences derived from radial basis functions (RBF-FD). To approximate the surface derivatives, a finite difference stencil that uses the $m$ grid points closest to a surface point is introduced that uses the closest point concept and the RBF-GA algorithm. A computational tube radius relation with the $m$ points on the stencil is given based on the solution to the Gauss circle problem. An advantage of our algorithm is the reduction of the computational tube radius (and the reduction of the number of points on the surface) for a second-order fi-
nite difference approximation of the Laplace-Beltrami operator. Numerical experiments are provided to validate the method for different types of PDEs on surfaces.

Next, an implicit formulation of the closest point method using RBF-FD is proposed. In this method, the constant normal extension of the approximate solution in the embedding space is imposed using a separate equation. The PDE and the constant extension in the normal direction system is solved using a least squares formulation. The least squares method provides the flexibility to use a variety of computational tubes around the surface, including ones that have holes. Numerical examples illustrate the convergence of the method on computational tubes missing 1% of the grid points (and their corresponding closest points on the surface) that lie within.

Finally, a simple coupling of the implicit formulation of the closest point method using RBF-FD and the grid based particle method is proposed to approximate the solution of the conservation law described in Section 3.2. Numerical results show second-order convergence for time step-sizes $\Delta t = \mathcal{O}(\Delta x^2)$. In addition, the coupled method is tested on strongly coupled systems including a cross-diffusion reaction-diffusion model and the Cahn-Hilliard equation.

Much work needs to be done to test the coupled method, including examples of PDE models on moving open surfaces and on surfaces with topological changes. An interesting application would employ the conservation law on two circles expanding in the normal direction and merging into one curve (see the merging of circles example in Figure 5.6). The least squares formulation of the implicit RBF-FD implementation of the closest point method should be capable of dealing with the discontinuity that arises from the merging of the two circles.

Furthermore, the implicit formulation of the closest point method using RBF-FD should be tested for the case of adaptive computational tubes surrounding the surface. Adaptivity is often required to capture fine features of a surface, i.e. areas with high curvature. The grid based particle method contains an optional adaptivity step for such occurrences (see Section 4.2.1). It is expected that the RBF-FD implementation of the closest point method will provide a simple way of developing schemes for adaptive tubes.

In addition, a simple coupling of the closest point method and the grid based particle method is performed in this thesis to solve the PDE on moving surfaces model described in Section 3.2 by alternating the two methods for each time step. The coupled method is capable of approximating the solution of general PDE systems with first-order accuracy in time. Second-order coupling techniques in time are of interest as well. A different coupling of the two methods that involves the interaction of the closest point method and the grid based particle method is required.

Another interesting direction for solving PDEs on moving surfaces employs meshfree techniques that use the closest point concept. Such methods have been developed for
approximating the solution of surface PDEs [68, 15], however no one until this thesis used RBF approximations to solve PDEs on moving surfaces.
Bibliography


Appendix A

The Gauss circle problem

Following [81], the Gauss circle problem is to determine the number of integer lattice points that lie within a circle centered at the origin and with radius $r$. In other words, for integers $x, y$ and $r$, find the number of ordered pairs $(x, y)$ satisfying

$$x^2 + y^2 \leq r^2,$$

where $x, y, r \geq 0$. A generalization considers circles of radius $r = \sqrt{q}$ where $q$ is a non-negative integer: For non-negative integers $x, y$ and $q$ find the number of ordered pairs $(x, y)$ satisfying

$$x^2 + y^2 \leq q.$$

The exact solution for the alternative formulation takes the form

$$m(q) = 1 + 4 \sum_{k=0}^{\lfloor \sqrt{q} \rfloor} \lfloor \sqrt{q - k^2} \rfloor$$

The first few integer solutions appear in Table 6.1. Figure A.1 gives illustrations for circles of radius $r = 1$ and $r = 2$.

Figure A.1: Left: Five lattice points lie within a circle of radius 1. Right: Thirteen lattice points lie within a circle of radius 2.
The Gauss circle problem can be extended to higher dimensions. We consider the extension in three dimensions: Find the number of integer lattice points that lie within a sphere centered at the origin with radius \( r = \sqrt{q} \), \( q = 0, 1, 2, \ldots \). For \( x, y, z \) and \( q \) non-negative integers, we must find the number of ordered triplets \((x, y, z)\) satisfying

\[
x^2 + y^2 + z^2 \leq q.
\]

The exact solution that is built on the result for the two dimensional problem [62] and takes the form

\[
m_{3D}(q) = m(q) + 2 \sum_{k=1}^{\lfloor \sqrt{q} \rfloor} m(q - k^2)
\]

for \( q = 1, 2, \ldots \) and \( m_{3D}(0) = 1 \). Some integer solutions appear in Table 6.1.
Appendix B

Sample code

We now provide sample MATLAB codes for the RBF-FD formulation of the explicit closest point method and the implicit RBF-FD closest point method.

B.1 Sample MATLAB code for the RBF-FD implementation of the CPM

In this section, a sample MATLAB implementation of the explicit closest point method using RBF-FD is provided for the heat equation on the unit circle. We make use of the RBF-GA algorithm [34] (see Section 2.4.1). For proper functionality, make sure that the arguments within the RBF-GA algorithm do not have any white space, i.e. use '0' instead of '0 '.

```matlab
% Sample MATLAB code of the explicit RBF-FD closest point method for the solution of the heat equation on the unit circle.

rad = 1; % Radius of the circle
dx = 0.1; % Grid size

% Construct a Cartesian grid
[xx,yy] = meshgrid(-2+dx/2:dx:2-dx/2,-2+dx/2:dx:2-dx/2);
xx = xx(:); yy = yy(:);

% Polar coordinates
[s,r] = cart2pol(xx,yy);

bw = sqrt(4)+sqrt(2)/2; % Computational tube radius
m = 13; % Number of points on the RBF-FD stencil

d = find(abs(r-rad)<=bw*dx);
s = s(d); % Angles within the computational tube
```
% Change to Cartesian coordinates on the unit circle
[x,y] = pol2cart(s,rad*ones(size(d)));

RBFCen = [xx(d) yy(d)]; % Cartesian grid points z_j
ctrs = [x,y]; % Closest points on the circle x_j

Tf = 1; % Final time
dt = 0.1*dx^2; % Time step-size
nt = round(Tf/dt); % Number of time steps to final time

% Initial solution
u = sin(s);

scale = 0.5; % Shape parameter

% Neighborhood size for the search of closest grid points
% (optional; for performance purposes)
Nbh = 1.5*bw;

% Initialization of the Laplace RBF-FD matrix L and
% the extension matrix I
L = spalloc(size(ctrs,1),size(RBFCen,1),m*size(ctrs,1));
I = L;

% Loop for all surface points (change to for loop
% if parallel toolbox is not available)
parfor j = 1:size(ctrs,1)
    % temporary variable to store weights for surface node x_j
    temp = spalloc(1,size(RBFCen,1),m);

    % Neighborhood for the search of closest grid points
    % (optional; for performance purposes)
    N = find(sqrt((ctrs(j,1)-RBFCen(:,1)).^2+...
               (ctrs(j,2)-RBFCen(:,2)).^2)<=Nbh*dx);

    % Sort grid points by distance to x_j
    [~,l] = sort(sqrt((ctrs(j,1)-RBFCen(N,1)).^2+...
                       (ctrs(j,2)-RBFCen(N,2)).^2));

    % Choose the first m closest grid points
    N = N(l);
    N = N(1:m);

    % Calculate the RBF-FD weights for the Laplacian and
    % the extension matrices
    K = rbfga_weights({'0','L'},scale,RBFCen(N,:),ctrs(j,:));

98
% Add weights to j-th row of the extension matrix
temp(N) = K(1,:);
I(j,:) = temp;

% Add weights to j-th row of the extension matrix
temp(N) = K(2,:);
L(j,:) = temp;

% Clear temporary variable (memory reasons)
temp = []; 
end

% Solve the PDE for nt time steps
for j = 1:nt
    u = I*u+dt*L*u;
end

% Calculate the relative error of the approximate
% solution vs the exact
max(abs(u-exp(-nt*dt)*sin(s)))/max(exp(-nt*dt)*sin(s))

B.2 Sample MATLAB code for the implicit RBF-FD discretization of the CPM

In this section, a sample MATLAB implementation of the implicit RBF-FD formulation of
the closest point method is provided for the heat equation on the unit circle. We make use
of the RBF-GA algorithm [34] (see Section 2.4.1) and the use of the Factorize algorithm [16]
(see Section 7.2) for the solution of the sparse matrices least squares system. For proper
functionality, include Factorize folder to the path in MATLAB and make sure that the
arguments within the RBF-GA algorithm do not have any white space, i.e. use '0' instead
of '0'.

% Sample MATLAB code of the implicit RBF-FD closest point
% method for the solution of the heat equation on the
% unit circle.

rad = 1; % Radius of the circle
dx = 0.1; % Grid size

% Construct a Cartesian grid
[xx,yy] = meshgrid(-2+dx/2:dx:2-dx/2,-2+dx/2:dx:2-dx/2);
xx = xx(:); yy = yy(:);

% Polar coordinates
[s,r] = cart2pol(xx,yy);
bw = sqrt(4)+sqrt(2)/2; % Computational tube radius
m = 13; % Number of points on the RBF-FD stencil

% Construction of computational tube
d = find(abs(r-rad)<=bw*dx);
s = s(d); % Angles within the computational tube

% Change to Cartesian coordinates on the unit circle
[x,y] = pol2cart(s, rad*ones(size(d)));

RBFCen = [xx(d) yy(d)]; % Cartesian grid points z_j
ctrs = [x,y]; % Closest points on the circle x_j

Tf = 1; % Final time
dt = 0.5*dx; % Time step-size
nt = round(Tf/dt); % Number of time steps to final time

% Initial solution
u = sin(s);

scale = 0.5; % Shape parameter

% Neighborhood size for the search of closest grid points
% (optional)
Nbh = 1.5*bw;

% Initialization of the Laplace RBF-FD matrix L and
% the extension matrix I
L = spalloc(size(ctrs,1), size(RBFCen,1)*m*size(ctrs,1));
I = L;

% Loop for all surface points (change to for loop
% if parallel toolbox is not available)
parfor j = 1:size(ctrs,1)
    % temporary variable to store weights for surface node x_j
    temp = spalloc(1, size(RBFCen,1), m);

    % Neighborhood for the search of closest grid points
    % (optional)
    N = find(sqrt((ctrs(j,1)-RBFCen(:,1)).^2+...
                (ctrs(j,2)-RBFCen(:,2)).^2)<=Nbh*dx);

    % Sort grid points by distance to x_j
    [~,1] = sort(sqrt((ctrs(j,1)-RBFCen(N,:)).^2+...
               (ctrs(j,2)-RBFCen(N,:)).^2));
% Choose the first m closest grid points
N = N(1);
N = N(1:m);

% Calculate the RBF-FD weights for the Laplacian and
% the extension matrices
K = rbfga_weights({'0','L'}, scale, RBFCen(N,:), ctrs(j,:));

% Add weights to j-th row of the extension matrix
temp(N) = K(1,:);
I(j,:) = temp;

% Add weights to j-th row of the extension matrix
temp(N) = K(2,:);
L(j,:) = temp;

% Clear temporary variable (memory reasons)
temp = [];
end

% Setting the initial solution
u0 = u;

% Initialize BDF2 using a step of backward Euler
u1 = factorize([speek(size(I))-dt*L; I-speek(size(I))])... 
\([I*u0; zeros(size(I,1),1)];\)

% Solve the PDE using BDF2 for nt time steps
for l = 2:nt
    u = factorize([speek(size(I))-2/3*dt*L;... 
        (I-speek(size(I)))]\... 
    [4/3*I*u1-1/3*I*u0; zeros(size(I,1),1)];
    u0 = u1;
    u1 = u;
end

% Calculate the relative error of the approximate
% solution vs the exact
max(abs(u-exp(-nt*dt)*sin(s)))/max(exp(-nt*dt)*sin(s))