

Solving Variational Problems and Partial Differential Equations Mapping Between Manifolds via the Closest Point Method

by

Nathan David King

B.Sc. (Hons.), Memorial University of Newfoundland, 2013

Thesis Submitted in Partial Fulfillment
of the Requirements for the Degree of

Master of Science

in the

Department of Mathematics

Faculty of Science

© Nathan David King 2015

SIMON FRASER UNIVERSITY

Spring 2015

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

Name: Nathan David King
Degree: Master of Science
Title: Solving Variational Problems and Partial Differential Equations Mapping Between Manifolds via the Closest Point Method

Examining Committee: **Chair:** Dr. Ben Adcock
Assistant Professor, Department of Mathematics
Simon Fraser University

Dr. Steven Ruuth

Senior Supervisor
Professor, Department of Mathematics
Simon Fraser University

Dr. Mirza Faisal Beg

Supervisor
Professor, School of Engineering Science
Simon Fraser University

Dr. Robert Russell

Internal Examiner/External Examiner
Professor Emeritus, Department of Mathematics
Simon Fraser University

Date Approved: April 9th, 2015

Partial Copyright Licence



The author, whose copyright is declared on the title page of this work, has granted to Simon Fraser University the non-exclusive, royalty-free right to include a digital copy of this thesis, project or extended essay[s] and associated supplemental files (“Work”) (title[s] below) in Summit, the Institutional Research Repository at SFU. SFU may also make copies of the Work for purposes of a scholarly or research nature; for users of the SFU Library; or in response to a request from another library, or educational institution, on SFU’s own behalf or for one of its users. Distribution may be in any form.

The author has further agreed that SFU may keep more than one copy of the Work for purposes of back-up and security; and that SFU may, without changing the content, translate, if technically possible, the Work to any medium or format for the purpose of preserving the Work and facilitating the exercise of SFU’s rights under this licence.

It is understood that copying, publication, or public performance of the Work for commercial purposes shall not be allowed without the author’s written permission.

While granting the above uses to SFU, the author retains copyright ownership and moral rights in the Work, and may deal with the copyright in the Work in any way consistent with the terms of this licence, including the right to change the Work for subsequent purposes, including editing and publishing the Work in whole or in part, and licensing the content to other parties as the author may desire.

The author represents and warrants that he/she has the right to grant the rights contained in this licence and that the Work does not, to the best of the author’s knowledge, infringe upon anyone’s copyright. The author has obtained written copyright permission, where required, for the use of any third-party copyrighted material contained in the Work. The author represents and warrants that the Work is his/her own original work and that he/she has not previously assigned or relinquished the rights conferred in this licence.

Simon Fraser University Library
Burnaby, British Columbia, Canada

revised Fall 2013

Abstract

Many applications from fields such as mathematical physics, image processing, computer vision and medical imaging require computation of maps between manifolds. A numerical framework is introduced for solving variational problems and partial differential equations that map from a source manifold \mathcal{M} to a target manifold \mathcal{N} . The numerics rely on the closest point representations of \mathcal{M} and \mathcal{N} . Using the closest point representation produces simple algorithms for handling complex surface geometries, since standard Cartesian numerical methods can be used.

The framework is illustrated with harmonic maps and a straightforward algorithm is given for this case. Harmonic maps are important in applications such as texture mapping, brain image regularization and colour image denoising. Moreover, the harmonic mapping energy is part of numerous energy functionals. The algorithm is justified theoretically and shown to be first order accurate. It is implemented in two applications: removing noise from texture maps and colour image enhancement.

Keywords: harmonic maps; closest point method; texture mapping; chroma enhancement; brain mapping; manifold mapping

*To my parents and grandparents
Rupert, Donna, Reginald, Gordon, Annie and Alice*

“The desire to fly is an idea handed down to us by our ancestors who, in their grueling travels across trackless lands in prehistoric times, looked enviously on the birds soaring freely through space, at full speed, above all obstacles, on the infinite highway of the air.”

—WILBUR WRIGHT

Acknowledgements

This work was partially supported by a Postgraduate Scholarship from the Natural Sciences and Engineering Research Council of Canada.

I thank Dr. Steven Ruuth for his suggestion of an intriguing research project and his most beneficial insight throughout. His enthusiasm for every direction I desired to take my research was invaluable. I further extend my gratitude to Dr. Mirza Faisal Beg and his research group for assisting with medical imaging applications. I also thank all other professors and fellow students who made this graduate experience exceptional.

A special thank you to Jillian, whose support throughout my post secondary education cannot be understated.

Contents

Approval	ii
Partial Copyright License	iii
Abstract	iv
Dedication	v
Quotation	vi
Acknowledgements	vii
Contents	viii
List of Tables	x
List of Figures	xi
1 Introduction	1
2 The Closest Point Method on One Manifold	5
2.0.1 Heat equation on the unit circle	8
3 The Closest Point Method for Maps Between Manifolds	12
3.1 The closest point method for Euler-Lagrange equations of harmonic maps .	15
3.1.1 Euler-Lagrange equations for liquid crystals	17
3.1.2 Geodesic computation on implicit manifolds using harmonic maps .	18
3.1.3 A note on general variational problems and partial differential equations	18
3.2 Projections onto target manifold \mathcal{N}	19

3.2.1	Applications in medical imaging	22
3.3	Justification of the approach	25
3.3.1	Decrease of harmonic energy	26
3.3.2	Equivalence to projection onto target manifold \mathcal{N}	32
4	Numerical Results	34
4.1	Identity maps	34
4.2	Diffusion of noisy texture maps	36
4.2.1	Texture mapping by multidimensional scaling	36
4.2.2	Harmonic maps from the plane	39
4.2.3	Harmonic maps from a cylinder to a submanifold of S^2	41
4.3	Denosing colour images	44
5	Conclusion	49
	Bibliography	52

List of Tables

4.1	Convergence study for the computation of a unit sphere identity map with Algorithm 3.1.	35
4.2	Convergence study for the computation of an ellipsoid identity map using Algorithm 3.1. The ellipsoid was constructed with minor axis of length 0.75 and major axis of length 1.25.	35
4.3	Convergence study of a torus identity map computation using Algorithm 3.1. The torus was constructed with minor radius 0.4 and major radius 1. . . .	35
4.4	Convergence study using errors between a reference solution \mathbf{u}_{ref} and harmonic map \mathbf{u} from a plane to S^2	43

List of Figures

2.1	Example of a computational domain Ω_c (indicated by \bullet) in a band around the unit circle S^1 . The ghost points in G (indicated by \circ) are also shown around the edge of Ω_c . Example stencils for barycentric Lagrange interpolation, with $p = 3$, at two points \diamond on S^1 are shown in the shaded regions. Five point Laplacian stencils are also depicted for two different points in Ω_c	10
3.1	Comparison of anatomical surfaces of the brain (left) and optic nerve head (right). Notice the convoluted nature of the brain compared to the optic nerve head, which is much smoother. Anatomical surface data courtesy of the Medical Image Analysis Lab at Simon Fraser University.	24
4.1	The original mapping \mathbf{w} of the image is given in (a)-(d) for two viewing angles with the two different planar images. The noisy initial map $\mathbf{u}_0(\mathbf{x})$ from a plane to the unit sphere is shown in (e)-(h). The harmonic map $\mathbf{u}(\mathbf{x})$ computed with Algorithm 3.1, after 60 time steps, is displayed on the unit sphere in (i)-(l).	40
4.2	The original mapping \mathbf{w} of the image is given in (a)-(d) for two viewing angles with the two different planar images. The noisy initial map $\mathbf{u}_0(\mathbf{x})$ from a plane to a pig is shown in (e)-(h). The harmonic map $\mathbf{u}(\mathbf{x})$ computed using Algorithm 3.1 with 15 time steps is displayed on the pig in (i)-(l). . .	42
4.3	The original mapping (described in text) of the image is given in (a)-(d) with two planar images and viewed at two different angles for each. The corresponding noisy initial map $\mathbf{u}_0(\mathbf{x})$ from the cylinder to unit sphere is shown in (e)-(h). The harmonic map $\mathbf{u}(\mathbf{x})$ computed with Algorithm 3.1 after 300 time steps is displayed on the sphere in (i)-(l).	45

4.4	Isotropic diffusion of chromaticity noise for two 256×256 pixel images: cartoon of Newfoundland row houses [19] (first row) and natural scene of flowers (second row). The noisy image (b) was denoised to give (c) after 25 time steps. The noisy flower image in (e) took 30 time steps to give the denoised image in (f).	46
4.5	Anisotropic diffusion of 256×256 pixel images with chromaticity noise. A cartoon of Newfoundland row houses [19] (a) with noise added to the chromaticity in (b) was denoised to give (c) after 55 time steps. The flower image (d) with noise in (e) took 100 time steps to give the denoised image in (f). .	47
4.6	Comparison of isotropic and anisotropic diffusion of a simple 256×256 pixel red, green and blue image. Chromaticity noise is added in (b) and denoised with 30 time steps in (c) using isotropic diffusion. Anisotropic diffusion is applied for 125 time steps to give the denoised image (d). Notice the blur of colours in (c) and the preservation of the edges between colours in (d). . .	48

Chapter 1

Introduction

The need to compute maps between a source manifold \mathcal{M} and a target manifold \mathcal{N} is present in mathematical physics, image processing, computer vision and medical imaging. In this thesis we are interested in maps between \mathcal{M} and \mathcal{N} that are defined by variational problems and partial differential equations (PDEs). Our primary concern is solving PDEs derived from the variational problems, i.e., Euler-Lagrange equations. The algorithms introduced extend however to handle more general PDEs.

In mathematical physics these types of maps occur in the study of liquid crystals [77], micromagnetic materials [48], biomembranes [78] and superconductors [14] (see [12] and references within for more details). Applications in image processing and computer vision arise in colour image enhancement by chromaticity diffusion [74], directional diffusion [73,76] and texture mapping during shape transformations [30]. The field of medical imaging contains applications such as brain image regularization [55] and brain mapping [68]. This thesis introduces a numerical framework for solving variational problems and PDEs that define maps between \mathcal{M} and \mathcal{N} .

There are many possible approaches to solve variational problems and PDEs posed on a single manifold. One popular approach is to use a smooth coordinate system or parameterization on the surface. There is however some difficulty in expressing differential operators within this coordinate system. In general a substantial complication of the surface PDE arises, usually involving non constant coefficients and additional derivative terms. Possibly more detrimental are the artificial singularities introduced by some coordinate systems, e.g., the stereographic projection. Parameterizations of the surface will almost always cause distortions in either angles or some region of the surface [35].

A second approach is to solve the PDE on a triangulated representation of the surface. For certain classes of equations this technique is satisfactory. As a general approach, however, there are numerous difficulties when using triangulated surfaces (see for example [16, 17, 54]). The discretization of the variational problem or PDE on triangulated surfaces is non-trivial. There is no general agreement on the approach to compute geometric primitives, such as tangents, normals, principal directions and curvatures. Furthermore, the convergence of numerical methods on triangulated surfaces is less understood compared to methods on Cartesian grids [40].

A more novel approach to solve variational problems and PDEs on a manifold is to embed the problem in a higher dimensional space. For example, if the heat equation $u_t = \Delta_{\mathcal{S}}u$ is posed on the unit sphere, then we somehow embed the PDE in all of \mathbb{R}^3 . This embedding PDE is constructed in such a way that its solution, when restricted to the surface, is the solution to the original surface PDE of interest. Note that even though the PDE is embedded in all of \mathbb{R}^3 , in practice one computes in a band surrounding the surface. Hence the increase in dimension of the problem does not introduce a sizable difference in the amount of computational work. The goal of an embedding approach is to allow for the treatment of complex surface geometry while using standard numerical methods on Cartesian grids. When using standard numerical methods on Cartesian grids one benefits from well established convergence results, see for example [49].

Two main types of embedding methods have been developed: the closest point method and the level set method. These embedding methods are based on two types of implicit representations of manifolds. The level set method for variational problems and PDEs on a single manifold was proposed by Bertalmío, Cheng, Osher and Sapiro in [16]. The level set method represents the surface of interest as the zero level set of a higher dimensional function. The surface PDE can then be embedded in this higher dimensional space, e.g., \mathbb{R}^3 for a 2-dimensional surface, and discretized using standard Cartesian numerical methods. The level set method was also extended by Méholi, Sapiro and Osher [54] to solve variational problems and PDEs that define maps between manifolds \mathcal{M} and \mathcal{N} . On the other hand, the closest point method, first introduced by Ruuth and Merriman [61], implicitly represents manifolds using closest point functions. It has been developed for surface PDEs posed on a single manifold \mathcal{S} . It is the goal of this thesis to present a numerical framework, using

closest point representations, for solving variational problems and PDEs that define maps between \mathcal{M} and \mathcal{N} .

Development of this framework using the closest point representation of \mathcal{M} and \mathcal{N} instead of the level set representation is well warranted. The most obvious limitation of the level set representation is that open surfaces with boundaries or objects of codimension two or higher are not immediately accommodated. It is however, in principle, possible to use multiple level set functions to represent the latter more general surfaces. Another limitation occurs during the practical implementation of the level set method. One only ever computes in a band surrounding the surface of interest instead of all of \mathbb{R}^3 . For the level set method artificial boundary conditions must be imposed on the edge of the band. The best approach to construct these boundary conditions is not apparent. Consequently, a degradation of the convergence order with the level set method is seen when banding is introduced, e.g., in diffusive problems [40].

Two main features of the closest point method allow the above limitations of the level set method to be overcome. The first is the use of a closest point representation of the manifold, which automatically accommodates more general manifolds. Implementing the closest point representation of the manifold also results in a simple embedding PDE, involving only standard Cartesian derivatives. Secondly, instead of solving the embedding PDE for all time, one solves the embedding PDE for one time step of the numerical scheme (or one stage of a Runge-Kutta method). This allows the computation in the embedding space to be constrained to a band around the surface without degrading the convergence order, since artificial boundary conditions are not propagated for all time. One important advantage when solving problems that map between manifolds \mathcal{M} and \mathcal{N} is the ability to represent submanifolds with the closest point representation. A second level set function had to be introduced to handle submanifolds when using the level set method [54].

This thesis is organized as follows. A review of the original closest point method for surface PDEs on a single manifold is given next in Chapter 2. Chapter 3 introduces our numerical framework for variational problems and PDEs that define maps between \mathcal{M} and \mathcal{N} . The framework is illustrated using the example of harmonic maps, which is important in many of the applications discussed above. Theoretical justification of the proposed algorithm is also given in Chapter 3. The behaviour and performance of our method is shown with numerical examples in Chapter 4. Convergence studies of the computation of identity

maps verify first order convergence of the algorithm. Noisy texture maps between a plane and different target manifolds \mathcal{N} are denoised. Colour image denoising is also accomplished using the framework introduced within. The thesis concludes with a discussion of possible further work and applications in Chapter 5.

Chapter 2

The Closest Point Method on One Manifold

A main component of our general algorithm is the closest point method on a single manifold. A review for explicit and implicit time-stepping discretizations of surface PDEs is provided below. For more in-depth implementation details of the closest point method see [51, 61]. The closest point method relies on two simple principles and the extension of surface data using the closest point function. Implicit time-stepping requires a slight modification to ensure stability. This modification is not drastic and is therefore attractive when widely varying scales are present (e.g., for problems requiring adaptivity).

The closest point representation of manifolds is of great importance to the closest point method. A closest point representation of the manifold \mathcal{S} , in the embedding space \mathbb{R}^d , assumes that for every $\mathbf{x} \in \mathbb{R}^d$ there exists a point $\text{cp}_{\mathcal{S}}(\mathbf{x}) \in \mathcal{S}$. That is, the closest point $\text{cp}_{\mathcal{S}}(\mathbf{x})$ to \mathbf{x} in Euclidean distance. Definition 1 defines the closest point function of \mathcal{S} , $\text{cp}_{\mathcal{S}}$.

Definition 1 *Let \mathbf{x} be some point in the embedding space \mathbb{R}^d . Then,*

$$\text{cp}_{\mathcal{S}}(\mathbf{x}) = \arg \min_{\mathbf{z} \in \mathcal{S}} \|\mathbf{x} - \mathbf{z}\|_2$$

is the closest point function of the manifold \mathcal{S} .

The point $\text{cp}_{\mathcal{S}}(\mathbf{x})$ may be non unique; however, for a smooth manifold \mathcal{S} it is unique if \mathbf{x} is within a narrow band around \mathcal{S} [52, 57]. The closest point function $\text{cp}_{\mathcal{S}}$ can then be uniquely defined within the narrow band and is a map that takes points $\mathbf{x} \in \mathbb{R}^d$ to points $\text{cp}_{\mathcal{S}}(\mathbf{x}) \in \mathcal{S}$. The width of the band will depend on geometry of the manifold \mathcal{S} , e.g., the size

of its principal curvatures. Properties of the closest point function and calculus involving $\text{cp}_{\mathcal{S}}$ have been investigated further by März and Macdonald in [52]. They also extend the definition of the closest point function to involve more general definitions of distance other than $\|\cdot\|_2$.

The closest point method is an embedding method: it extends the problem defined on a surface \mathcal{S} to the embedding space \mathbb{R}^d surrounding it. A function extending the function of surface data, u , is easily accomplished by composition with the closest point function. That is, $u(\text{cp}_{\mathcal{S}}(\mathbf{y}))$ is the closest point extension of u at the point $\mathbf{y} \in \mathbb{R}^d$. Note that u represents a scalar function in this Chapter only; throughout the rest of the thesis \mathbf{u} is a vector-valued function representing a map from \mathcal{M} to \mathcal{N} . The extension step allows the intrinsic surface gradient $\nabla_{\mathcal{S}}$ and surface divergence ($\nabla_{\mathcal{S}}\cdot$) operators to be replaced by the “standard” gradient ∇ and divergence ($\nabla\cdot$) via the following principles [61]:

Principle 1 *Let v be any function on \mathbb{R}^d that is constant along normal directions of \mathcal{S} . Then, at the surface, intrinsic gradients are equivalent to standard gradients, $\nabla_{\mathcal{S}}v = \nabla v$.*

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

Higher order derivatives can be handled by combining Principles 1 and 2 with extensions into the embedding space. For example, consider approximating the Laplace-Beltrami operator $\Delta_{\mathcal{S}}u = \nabla_{\mathcal{S}}\cdot(\nabla_{\mathcal{S}}u)$. If u is a function defined on \mathcal{S} , then $u(\text{cp}_{\mathcal{S}})$ is constant along normal directions of \mathcal{S} and therefore $\nabla_{\mathcal{S}}u = \nabla u(\text{cp}_{\mathcal{S}})$ by Principle 1. Principle 2 indicates that $\nabla_{\mathcal{S}}\cdot(\nabla_{\mathcal{S}}u) = \nabla\cdot(\nabla u(\text{cp}_{\mathcal{S}}))$ since $\nabla_{\mathcal{S}}u$ is always tangent to the level sets of the distance function of \mathcal{S} . In this fashion a PDE of interest is embedded in the surrounding space \mathbb{R}^d and is written in terms of standard Cartesian derivatives.

To solve surface PDEs, a narrow computational domain, Ω_c , surrounding \mathcal{S} is chosen. Initial surface data is extended onto Ω_c using the closest point function. The explicit closest point method for PDEs on a single manifold alternates between the following two steps:

- Use standard finite differences on a Cartesian mesh in Ω_c to solve the embedded PDE for one time step.
- Replace u by $u(\text{cp}_{\mathcal{S}})$ for all $\mathbf{x} \in \Omega_c$. That is, extend the current solution on \mathcal{S} to the computational domain.

Note that the closest point extension in the second step involves interpolation. Interpolation is needed since $\text{cp}_{\mathcal{S}}(\mathbf{x})$ is not necessarily a grid point in Ω_c . Appropriate interpolation order is therefore necessary to ensure the expected accuracy is achieved. We use barycentric Lagrange interpolation [15] in a dimension-by-dimension fashion, which can be written in matrix form. The finite differences are also written in matrix form and thus a matrix system of ODEs is solved in the end. These matrices for finite differences and extending the solution are discussed in Section 2.0.1.

The implicit closest point method is nearly identical except for a small modification to ensure stability. To illustrate the difference, consider the PDE describing heat flow within the surface \mathcal{S} , $u_t = \Delta_{\mathcal{S}}u$. Using Principles 1 and 2 it was shown above that $\nabla_{\mathcal{S}} \cdot (\nabla_{\mathcal{S}}u) = \nabla \cdot (\nabla u(\text{cp}_{\mathcal{S}}))$, so this heat equation can be written as

$$\begin{cases} u_t(\mathbf{x}, t) = \Delta u(\text{cp}_{\mathcal{S}}(\mathbf{x}), t), \\ u(\mathbf{x}, 0) = u_0(\text{cp}_{\mathcal{S}}(\mathbf{x})), \end{cases} \quad (2.1)$$

where we have included the initial condition $u_0(\text{cp}_{\mathcal{S}}(\mathbf{x}))$, i.e., the extension of u_0 onto Ω_c . Equation (2.1) is the embedding PDE and has the property that it agrees with the original surface PDE at the surface.

Now for simplicity consider that \mathcal{S} is a curve embedded in \mathbb{R}^2 . Applying a second order centred finite difference scheme to the Laplacian operator in (2.1) yields

$$\begin{cases} \frac{\partial u}{\partial t} = \frac{1}{\Delta x^2} \left(-4u(\text{cp}_{\mathcal{S}}(x, y), t) + u(\text{cp}_{\mathcal{S}}(x + \Delta x, y), t) + u(\text{cp}_{\mathcal{S}}(x - \Delta x, y), t) \right. \\ \qquad \qquad \qquad \left. + u(\text{cp}_{\mathcal{S}}(x, y + \Delta x), t) + u(\text{cp}_{\mathcal{S}}(x, y - \Delta x), t) \right) + \mathcal{O}(\Delta x^2), \\ u(x, y, 0) = u_0(\text{cp}_{\mathcal{S}}(x, y)), \end{cases} \quad (2.2)$$

where Δx is the uniform grid spacing of Ω_c . Note that (2.2) is not a full discretization since it still involves continuous variables t , x and y . To finish the discretization in space one approximates the solution $u(x, y, t)$ at points $\{(x_i, y_j)\} \in \Omega_c$ such that $u_{ij}(t) \approx u(x_i, y_j, t)$. Discretizing (2.2) onto grid points $\{(x_i, y_j)\}$ involves computing $u(\text{cp}_{\mathcal{S}}(x_i, y_j), t)$. This is where the interpolation step mentioned above is needed. One replaces $u(\text{cp}_{\mathcal{S}}(x_i, y_j), t)$ by a linear combination of the solution at nearby grid points to approximate $u_{ij}(t)$.

The total discretization is then accomplished by discretizing in time. The explicit closest point method uses some explicit time-stepping scheme and has no problem with stability. On the other hand, with an implicit time-stepping method one must slightly modify (2.2)

because computations will become unstable [51]. The modification of (2.2) drops the redundant closest point operators, which occur in the diagonal terms of the discretization, to give

$$\begin{cases} \frac{\partial u}{\partial t} = \frac{1}{\Delta x^2} \left(-4u(x, y, t) + u(\text{cp}_{\mathcal{S}}(x + \Delta x, y), t) + u(\text{cp}_{\mathcal{S}}(x - \Delta x, y), t) \right. \\ \qquad \qquad \qquad \left. + u(\text{cp}_{\mathcal{S}}(x, y + \Delta x), t) + u(\text{cp}_{\mathcal{S}}(x, y - \Delta x), t) \right) + \mathcal{O}(\Delta x^2), \\ u(x, y, 0) = u_0(\text{cp}_{\mathcal{S}}(x, y)). \end{cases} \quad (2.3)$$

The closest point operator is redundant for the diagonal terms since the discretized PDE (2.3) still agrees with the original PDE, $u_t = \Delta_{\mathcal{S}}u$, at the surface; $\text{cp}_{\mathcal{S}}(x, y) = (x, y)$ for all $(x, y) \in \mathcal{S}$. It is shown in [51] that the discretization in (2.3) enjoys improved stability. The matrix form of (2.3), given in Section 2.0.1, benefits further from being more diagonally dominant. Also see [51] for a more general description of the implicit closest point method and examples of other PDEs.

2.0.1 Heat equation on the unit circle

To provide a concrete understanding of the closest point method, implementation for the heat equation on a circle is discussed. The choice of a computational domain Ω_c in a band around the surface \mathcal{S} is described. The matrix form of the closest point method is introduced in this section. Furthermore, implementations of the two alternating steps of the closest point method are detailed.

Of course the first component one needs is a closest point function corresponding to \mathcal{S} . Here, for the unit circle S^1 , an analytic formula for the closest point function is simply

$$\text{cp}_{\mathcal{S}}(\mathbf{x}) = \frac{\mathbf{x}}{\|\mathbf{x}\|_2}. \quad (2.4)$$

The embedding PDE is then found using Principles 1 and 2. This gives a PDE in Cartesian coordinates of \mathbb{R}^2 , since surface gradients are replaced by standard gradients in \mathbb{R}^2 . The embedding PDE for the heat equation on S^1 is the same as (2.1), i.e., heat flow in \mathbb{R}^2 .

Now one must choose the computational domain Ω_c . A uniform square grid around \mathcal{S} could be chosen. For improved efficiency, however, a narrow band around \mathcal{S} is desirable. That is,

$$\Omega_c = \{\mathbf{x} : \|\mathbf{x} - \text{cp}_{\mathcal{S}}(\mathbf{x})\|_2 \leq \lambda\},$$

where λ is the bandwidth. The bandwidth depends on the degree of the interpolating polynomial p , the order of the PDE derivatives r , the order of the differencing scheme q and the dimension of the embedding space d . It is shown in [61] that for second order centred differences discretizing the Laplacian

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

Interpolation order depends on the derivative order r and differencing scheme order q . The interpolation order should be large enough not to produce errors greater than the differencing scheme, i.e., $q + r$ or higher. In our numerical examples $q = r = 2$, so $p = 3$ is taken to acquire order 4 interpolation. Writing code using this narrow band around \mathcal{S} is simple. One first constructs a uniform square grid around \mathcal{S} and then uses an indexing array to access points within the band.

Consider now writing the interpolation, necessary to extend solution data onto Ω_c , in matrix form. Barycentric Lagrange interpolating polynomials of degree p use a linear combination of values at $p+1$ neighbouring grid points in each dimension. The interpolation stencil for some point on S^1 is therefore a square consisting of $(p+1)^2$ grid points. A discrete ordered list of points $L = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\}$ is constructed. The list L consists of all grid points that can appear in the interpolation stencil for any point on S^1 . Note that the points in L are an ordered list of the points that comprise Ω_c , which are shown in Figure 2.1 as solid points. The open points in Figure 2.1 make up a second list of ordered points $G = \{\mathbf{x}_{N+1}, \mathbf{x}_{N+2}, \dots, \mathbf{x}_{N+N_g}\}$, which is disjoint from L . The N_g points in G are “ghost points” that preside along the edge of the computational band. The points in G are needed for the 5-point stencil of the Laplacian discretization, but are not propagated in time.

Now let the vector $\vec{u} \in \mathbb{R}^N$ denote the approximation to the solution u at all the points in L . That is, the vector \vec{u} has entries $u_i \approx u(\mathbf{x}_i)$ for all $\mathbf{x}_i \in L$. Also let $u(\text{cp}_{\mathcal{S}}(L)) \in \mathbb{R}^N$ and $u(\text{cp}_{\mathcal{S}}(G)) \in \mathbb{R}^{N_g}$ denote vectors with components $u(\text{cp}_{\mathcal{S}}(\mathbf{x}_i))$ for $\mathbf{x}_i \in L$ and $u(\text{cp}_{\mathcal{S}}(\mathbf{x}_{N+i}))$ for $\mathbf{x}_{N+i} \in G$, respectively. The closest point extension extends data on the surface \mathcal{S} to the computational domain by assigning a value of $u(\text{cp}_{\mathcal{S}}(\mathbf{x}))$ to $u(\mathbf{x})$. The extension step can be accomplished by matrix multiplication with the extension matrix \mathbf{E} .

Definition 2 *Let \vec{u} , $u(\text{cp}_{\mathcal{S}}(L))$ and $u(\text{cp}_{\mathcal{S}}(G))$ be defined as described above. Choose an interpolation scheme that uses a linear combination of values at neighbouring grid points.*

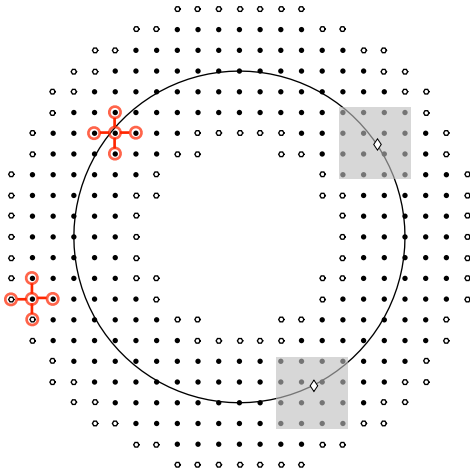


Figure 2.1: Example of a computational domain Ω_c (indicated by \bullet) in a band around the unit circle S^1 . The ghost points in G (indicated by \circ) are also shown around the edge of Ω_c . Example stencils for barycentric Lagrange interpolation, with $p = 3$, at two points \diamond on S^1 are shown in the shaded regions. Five point Laplacian stencils are also depicted for two different points in Ω_c .

Then the extension matrix is an $(N + N_g) \times N$ matrix operator \mathbf{E} such that

$$\begin{pmatrix} u(\text{cp}_{\mathcal{S}}(L)) \\ u(\text{cp}_{\mathcal{S}}(G)) \end{pmatrix} \approx \mathbf{E}\vec{u} = \begin{matrix} \text{[Large Gray Box]} \\ \mathbf{E} \end{matrix} \begin{matrix} \text{[Small Gray Box]} \\ \vec{u} \end{matrix}.$$

The chosen interpolation scheme for $u(\text{cp}_{\mathcal{S}}(\mathbf{x}_i))$ determines the nonzero entries of the i -th row of \mathbf{E} . More explicitly, the components of \mathbf{E} are

$$\{\mathbf{E}\}_{ij} = \begin{cases} w_j & \text{if } \mathbf{x}_j \text{ is in the interpolation stencil for } \text{cp}_{\mathcal{S}}(\mathbf{x}_i), \\ 0 & \text{otherwise,} \end{cases}$$

where w_j is the weight of grid point \mathbf{x}_j in the interpolation scheme for the point $\text{cp}_{\mathcal{S}}(\mathbf{x}_i)$.

The explicit and implicit closest point methods can be written using the extension matrix \mathbf{E} . The two steps of the explicit closest point method are simply:

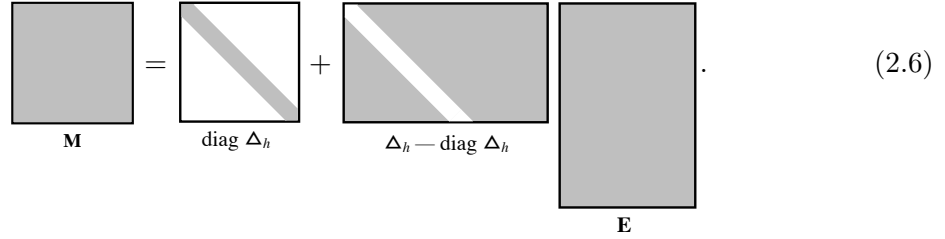
- Solve the embedding PDE for one time step of size ε , e.g., with forward Euler time stepping $\vec{w}^{n+1} = \vec{w}^n + \varepsilon \Delta_h \vec{w}^n$, where Δ_h is an $N \times (N + N_g)$ matrix that approximates the Laplacian operator in \mathbb{R}^3 .
- Perform the closest point extension $\vec{w}^{n+1} = \mathbf{E}\vec{w}^{n+1}$.

Alternatively, writing the two steps together one has $\bar{u}^{n+1} = \mathbf{E}\bar{u}^n + \varepsilon\Delta_h\mathbf{E}\bar{u}^n$.

For the implicit closest point method one must remove the redundant interpolation to achieve stability. One instead splits the diagonal part of the matrix Δ_h out before multiplying by the extension matrix \mathbf{E} . That is, define a new stable version of the discretization of the Laplacian operator as

$$\mathbf{M} = \text{stab}(\Delta_h, \mathbf{E}) \equiv \text{diag}\Delta_h + (\Delta_h - \text{diag}\Delta_h)\mathbf{E}, \quad (2.5)$$

or pictorially



$\mathbf{M} = \text{diag } \Delta_h + (\Delta_h - \text{diag } \Delta_h) \mathbf{E}.$
(2.6)

Hence the spatial discretization for the implicit closest point method is $\bar{u}_t = \mathbf{M}\bar{u}$. Using, for example, backward Euler time stepping the complete discretization of the surface heat equation is

$$\bar{u}^{n+1} = \bar{u}^n + \varepsilon\mathbf{M}\bar{u}^{n+1}.$$

Chapter 3

The Closest Point Method for Maps Between Manifolds

In this chapter, we introduce our framework for solving variational problems and PDEs that define maps between a source manifold \mathcal{M} and a target manifold \mathcal{N} . To give a clear, explicit presentation, the method is developed for the case of harmonic maps. Harmonic maps are important in many applications such as texture mapping [30], regularization of brain images [55] and colour image enhancement [74]. Some properties of harmonic maps will be discussed first. Section 3.1 and 3.2 then describe the numerical framework. Section 3.3 gives justification of the algorithm introduced in Section 3.2.

With most problems in mathematics there are two important questions one must first ask: does a solution exist and is the solution unique? It is important to know the answers to these two questions when using numerical methods for any problem. It is meaningless to have a numerical method to “solve” a problem if the problem does not have a solution. One may also be deceived if it is assumed that there is a unique solution and the numerical method returns multiple solutions. There are however exceptions here since sometimes numerics are used to prove the existence of solutions, e.g., in rigorous computing, or to give insight on a problem.

Existence and uniqueness for harmonic maps is outside the scope of this thesis. The question of regularity (or partial regularity) of solutions is also a difficult subject. There has been substantial analysis of harmonic maps between manifolds. There are far too many results to give a comprehensive summary here. We therefore only include references. Most of the work has been done assuming the manifolds \mathcal{M} and \mathcal{N} are Riemannian [21,23,27,32–

34, 38, 43, 45, 62–65, 70–72]. There has however been some work involving other manifolds, such as Kähler manifolds [22, 37] and Finsler manifolds [67]. Some good books that study the subject of harmonic maps are [47, 57, 66].

We now give the definition of a harmonic map between two Riemannian manifolds \mathcal{M} and \mathcal{N} of dimension m and l , respectively. Harmonic maps have the desirable property of being smooth, since intuitively a map is harmonic when \mathcal{M} is constrained to \mathcal{N} in elastic equilibrium. Denote the metric tensor of \mathcal{M} as $(\gamma_{\alpha\beta})_{\alpha,\beta=1,\dots,m}$ in local coordinates. Similarly for \mathcal{N} its metric tensor is $(g_{ij})_{i,j=1,\dots,l}$. The inverses of these metric tensors are denoted $\gamma^{\alpha\beta}$ and g^{ij} .

Definition 3 *Harmonic maps $\mathbf{u} : \mathcal{M} \rightarrow \mathcal{N}$ are the critical points of the Dirichlet energy*

$$E[\mathbf{u}] = \int_{\mathcal{M}} e[\mathbf{u}] dv_{\mathcal{M}}, \quad (3.1)$$

where

$$e[\mathbf{u}] = \frac{1}{2} |d\mathbf{u}|^2 = \frac{1}{2} \sum_{\alpha,\beta=1}^m \sum_{i,j=1}^l \gamma^{\alpha\beta}(\mathbf{x}) g_{ij}(\mathbf{u}(\mathbf{x})) \frac{\partial u^i}{\partial x^\alpha} \frac{\partial u^j}{\partial x^\beta}, \quad (3.2)$$

is the energy density and $dv_{\mathcal{M}} = (\det \gamma_{\alpha\beta})^{\frac{1}{2}} d\mathbf{x}$ is the volume element of \mathcal{M} .

The map $\mathbf{u} : \mathcal{M} \rightarrow \mathcal{N}$ must be a C^1 map so that $e[\mathbf{u}]$ is well-defined. Throughout this thesis we will assume that $\mathbf{u} \in C^1$ unless otherwise stated. Furthermore, by the Nash embedding theorem [58, 59] any Riemannian manifold \mathcal{N} can be isometrically embedded in a Euclidean space \mathbb{R}^n . Thus throughout this thesis think of $\mathbf{u} = (u^1, u^2, \dots, u^n)^T$ with point-wise constraint $\mathbf{u}(\mathbf{x}) \in \mathcal{N}$ for any $\mathbf{x} \in \mathcal{M}$.

The corresponding Euler-Lagrange equations of (3.1) are

$$\Delta_{\mathcal{M}} u^k + \sum_{\alpha,\beta=1}^m \sum_{i,j=1}^n \gamma^{\alpha\beta} \Gamma_{ij}^k(\mathbf{u}) \frac{\partial u^i}{\partial x^\alpha} \frac{\partial u^j}{\partial x^\beta} = 0, \quad \text{for } k = 1, 2, \dots, n, \quad (3.3)$$

where

$$\Gamma_{ij}^k = \frac{1}{2} \sum_{l=1}^n g^{kl} (g_{il,j} + g_{jl,i} - g_{ij,l})$$

are the Christoffel symbols for the target manifold \mathcal{N} with

$$g_{jl,i} \equiv \frac{\partial}{\partial u^i} g_{jl}.$$

Any map \mathbf{u} satisfying the Euler-Lagrange equations (3.3) is a critical point of the harmonic energy (3.1). Note that the energy (3.1) and its corresponding Euler-Lagrange equations (3.3) are given for explicit representations of \mathcal{M} and \mathcal{N} above. That is, everything

has been defined in terms of local coordinates $\{x^\alpha\}_{\alpha=1,2,\dots,m}$ on \mathcal{M} and $\{u^i\}_{i=1,2,\dots,n}$ on \mathcal{N} . The idea of the numerical framework here is to instead reformulate problems in terms of an implicit representation using the closest point function. This idea is shown and discussed further for harmonic maps in Section 3.1.

Instead of discussing the existence, uniqueness and regularity of harmonic maps we list some special cases of harmonic maps. This is to give the reader some intuition for what a harmonic map is. Three special cases are:

1. Consider $\dim(\mathcal{M}) = 1$, then harmonic maps give precisely the geodesics of \mathcal{N} [32].

When $\dim(\mathcal{M}) = 1$ the Euler-Lagrange equations (3.3) become

$$\ddot{u}^k + \sum_{i,j=1}^n \Gamma_{ij}^k(\mathbf{u}) \dot{u}^i \dot{u}^j = 0, \quad \text{for } k = 1, 2, \dots, n, \quad (3.4)$$

which are the familiar Euler-Lagrange equations defining geodesics on \mathcal{N} .

2. Take $\mathcal{N} = \mathbb{R}^n$ and thus $g_{ij} = \delta_{ij}$. In this case the map $\mathbf{u} : \mathcal{M} \rightarrow \mathcal{N}$ is harmonic if and only if each component u^k , $k = 1, 2, \dots, n$, is a harmonic function [66], i.e., $\Delta_{\mathcal{M}} u^k = 0$. When $\mathcal{N} = \mathbb{R}^n$ the Christoffel symbols are identically zero since $\frac{\partial}{\partial u^i} \delta_{jl} = 0$, which reduces (3.3) to $\Delta_{\mathcal{M}} u^k = 0$ for $k = 1, 2, \dots, n$.
3. The identity map $\text{id} : \mathcal{M} \rightarrow \mathcal{M}$ of any Riemannian manifold is a harmonic map [47].

To make this apparent, we remark that the Laplace-Beltrami operator can be written explicitly as

$$\Delta_{\mathcal{M}} u^k = \sum_{\alpha,\beta=1}^m \gamma^{\alpha\beta} \left(\frac{\partial^2 u^k}{\partial x^\alpha \partial x^\beta} - \sum_{\delta=1}^m C_{\alpha\beta}^\delta \frac{\partial u^k}{\partial x^\delta} \right),$$

where $C_{\alpha\beta}^\delta$ are the Christoffel symbols for the source manifold \mathcal{M} . For the identity map $C_{\alpha\beta}^\delta = \Gamma_{ij}^k$ and $\mathbf{u}(\mathbf{x}) = \mathbf{x}$ so $\partial u^k / \partial x^\alpha = \delta_{k\alpha}$. It is henceforth easy to check that (3.3) is satisfied.

A physical intuition for harmonic maps, given by Eells and Lemaire [32], is possibly more instructive. Denote the *tension field* of \mathbf{u} by $\tau(\mathbf{u}) = \text{div}(d\mathbf{u})$ (see [32] for definition of div). One can then imagine that the source manifold \mathcal{M} is made of rubber and the target manifold \mathcal{N} is made of marble. The map \mathbf{u} will constrain \mathcal{M} to lie on \mathcal{N} . For each point $\mathbf{x} \in \mathcal{M}$, there is a vector $\tau(\mathbf{u}(\mathbf{x})) = \text{div}(d\mathbf{u}(\mathbf{x}))$ that corresponds to the ‘‘tension’’ in the rubber at the point $\mathbf{u}(\mathbf{x})$. The map \mathbf{u} is harmonic if and only if $\tau(\mathbf{u}(\mathbf{x})) = 0$, for all

$\mathbf{x} \in \mathcal{M}$. That is, \mathbf{u} is harmonic if and only if \mathbf{u} constrains the manifold \mathcal{M} to \mathcal{N} in elastic equilibrium by minimizing (3.1).

Work on numerical methods for harmonic maps is more recent. Most numerical schemes were developed for the special case of $\mathcal{N} = S^{n-1}$, the unit hypersphere. A number of algorithms for finding stable solutions of harmonic maps, into $\mathcal{N} = S^{n-1}$, were originally proposed in [24,25,29,50]. One of the first numerical algorithms that was proven to converge in a continuous setting was introduced by Alouges in [7]. The latter algorithm was then proven to converge in a finite element setting with acute triangles by Bartels [11]. Further finite element methods have been developed for more difficult problems, e.g., [10] for p -harmonic maps and [8] for the Landau-Lifschitz-Gilbert equation. A finite element method for more general target manifolds has also been introduced by Bartels [13].

A different approach was taken by Vese and Osher [76] for p -harmonic maps with $\mathcal{N} = S^{n-1}$. Instead of using a finite element approach a parametric approach was taken. Their method was successfully used to denoise colour images; however, it is restricted to $\mathcal{N} = S^{n-1}$. The work of Mémoli et al. [54] is the most similar to our approach illustrated next. In our approach, the problem is embedded in the surrounding space using the closest point representations of \mathcal{M} and \mathcal{N} . On the other hand, Mémoli et al. [54] embed the problem in the surrounding space using level set representations. These embedding methods share the advantage of working for general \mathcal{M} and \mathcal{N} . By adopting the closest point formulation instead of the level set formulation, geometric flexibility is gained (the surfaces can be open or closed and need not be oriented). Moreover, as will be seen in Section 3.2, our algorithm is simpler and therefore more straightforward to implement.

3.1 The closest point method for Euler-Lagrange equations of harmonic maps

Assume \mathcal{M} and \mathcal{N} are Riemannian manifolds and $\mathbf{u} : \mathcal{M} \rightarrow \mathcal{N}$ is a C^1 map. The source manifold \mathcal{M} is of degree m and \mathcal{N} is isometrically embedded in \mathbb{R}^n . Denote the signed distance functions of \mathcal{M} and \mathcal{N} as $d_{\mathcal{M}}$ and $d_{\mathcal{N}}$ [52], respectively. The signed distance function $d_{\mathcal{S}}(\mathbf{x})$ gives the Euclidean distance from \mathbf{x} to the manifold \mathcal{S} with a + or - sign depending on which side of \mathcal{S} the point \mathbf{x} is located. The signed distance function has the important property that $\|\nabla d_{\mathcal{S}}\|_2 = 1$. The energy (3.1) defining harmonic maps can be

rewritten as [54, 55]

$$E[\mathbf{u}] = \frac{1}{2} \int_{\mathcal{M}} \|\mathbf{J}_{\mathbf{u}}^{\text{d}\mathcal{M}}\|_{\mathcal{F}}^2 dv_{\mathcal{M}}, \quad (3.5)$$

where $\|\cdot\|_{\mathcal{F}}$ is the Frobenius norm and $\mathbf{J}_{\mathbf{u}}^{\text{d}\mathcal{M}}$ is the Jacobian of \mathbf{u} intrinsic to \mathcal{M} . The intrinsic Jacobian is $\mathbf{J}_{\mathbf{u}}^{\text{d}\mathcal{M}} = \mathbf{J}_{\mathbf{u}} \Pi_{\nabla \text{d}\mathcal{M}}$, where $\Pi_{\nabla \text{d}\mathcal{M}} = \mathbf{I} - \nabla \text{d}\mathcal{M} \nabla \text{d}\mathcal{M}^T$ is the projection operator onto the tangent space of \mathcal{M} .

To motivate the definition (3.5) of the *harmonic energy* consider both \mathcal{M} and \mathcal{N} to be Euclidean. For Euclidean \mathcal{M} and \mathcal{N} the metrics $\gamma_{\alpha\beta} = K_1 \delta_{\alpha\beta}$ and $g_{ij} = K_2 \delta_{ij}$ are proportional to the identity, for some constants $K_1, K_2 \in \mathbb{R}$. The energy density (3.2) simplifies to

$$e[\mathbf{u}] = \frac{K}{2} \sum_{\alpha=1}^m \sum_{i=1}^n \left(\frac{\partial u^i}{\partial x^\alpha} \right)^2 \propto \|\mathbf{J}_{\mathbf{u}}\|_{\mathcal{F}}^2.$$

Therefore the energy (3.1) for Euclidean \mathcal{M} and \mathcal{N} becomes

$$E[\mathbf{u}] = \frac{K}{2} \int_{\mathcal{M}} \|\mathbf{J}_{\mathbf{u}}\|_{\mathcal{F}}^2 dv_{\mathcal{M}}.$$

The definition of the energy (3.5) for general \mathcal{M} and \mathcal{N} instead uses the intrinsic Jacobian $\mathbf{J}_{\mathbf{u}}^{\text{d}\mathcal{M}}$. The constraint that $\mathbf{u}(\mathbf{x}) \in \mathcal{N}$ incorporates the general target manifold \mathcal{N} .

Mémoli et al. [54] derived the Euler-Lagrange equations for (3.5) in terms of the level set representation of \mathcal{N} assuming \mathcal{M} is flat and open. The same calculation is carried out by Moser [57] in terms of the closest point representation of \mathcal{N} . There the closest point function is called the nearest point projection by Moser and is used to prove regularity results of harmonic maps (see Chapter 3 of [57]). Moser shows that the Euler-Lagrange equations corresponding to (3.5), assuming \mathcal{M} is flat and open, are

$$\Delta \mathbf{u} - \sum_{\alpha=1}^m \mathbf{H}_{\text{cp}_{\mathcal{N}}(\mathbf{u})} \left[\frac{\partial \mathbf{u}}{\partial x^\alpha}, \frac{\partial \mathbf{u}}{\partial x^\alpha} \right] = 0, \quad (3.6)$$

where the notation $\mathbf{A}[\mathbf{x}, \mathbf{y}] = (\mathbf{y}^T \mathbf{A}^1 \mathbf{x}, \mathbf{y}^T \mathbf{A}^2 \mathbf{x}, \dots, \mathbf{y}^T \mathbf{A}^n \mathbf{x})^T$ is used. The matrix $\mathbf{H}_{\text{cp}_{\mathcal{N}}(\mathbf{u})}$ denotes the Hessian of $\text{cp}_{\mathcal{N}}(\mathbf{u})$, i.e., the Hessian of each component of $\text{cp}_{\mathcal{N}}(\mathbf{u})$ is $\mathbf{H}_{\text{cp}_{\mathcal{N}}(\mathbf{u})}^i$ for $i = 1, 2, \dots, n$.

The key identity that Moser derives is that the second fundamental form of \mathcal{N} can be written in terms of the Hessian of $\text{cp}_{\mathcal{N}}$. In local coordinates the same identity involves the Christoffel symbols. Let $\Gamma(\mathbf{u}) = (\Gamma_{ij}^1(\mathbf{u}), \Gamma_{ij}^2(\mathbf{u}), \dots, \Gamma_{ij}^n(\mathbf{u}))^T$; then for all $\mathbf{u} \in \mathcal{N}$ and X, Y in the tangent space, $T_{\mathbf{u}}\mathcal{N}$, of \mathcal{N} at the point \mathbf{u} , we have

$$\mathbf{H}_{\text{cp}_{\mathcal{N}}(\mathbf{u})} [X, Y] = -\Gamma(\mathbf{u}) [X, Y]. \quad (3.7)$$

Using (3.7) the Euler-Lagrange equations (3.3) for general \mathcal{M} and \mathcal{N} can be wrote in terms of $\text{cp}_{\mathcal{N}}$ as

$$\Delta \mathbf{u}(\text{cp}_{\mathcal{M}}(\mathbf{x})) - \sum_{\alpha, \beta=1}^m \{\Pi_{T_{\mathbf{u}}\mathcal{M}}\}_{\alpha\beta} \mathbf{H}_{\text{cp}_{\mathcal{N}}(\mathbf{u})} \left[\frac{\partial \mathbf{u}}{\partial x^\alpha}, \frac{\partial \mathbf{u}}{\partial x^\beta} \right] = 0, \quad (3.8)$$

where the correspondence $\gamma^{\alpha\beta} = \{\Pi_{T_{\mathbf{u}}\mathcal{M}}\}_{\alpha\beta}$ from [54] is used.

A solution to (3.6) or (3.8) can be obtained by evolving the corresponding gradient descent flow to steady state (cf. [54, 57]). The gradient descent flow is a PDE which introduces an artificial time variable and evolves in the direction of maximal decrease of the energy. Numerically one would discretize the gradient descent flow and evolve the solution until some long time t_f . For numerical examples in Chapter 4, we instead use a different approach for harmonic maps between general \mathcal{M} and \mathcal{N} . This simple algorithm is described in Section 3.2, which is based on rewriting (3.8) in a different form.

3.1.1 Euler-Lagrange equations for liquid crystals

One can verify the Euler-Lagrange equations (3.6) by considering the case of liquid crystals, i.e., \mathcal{M} is a flat, open subset of \mathbb{R}^m and $\mathcal{N} = S^{n-1}$. Note that the closest point function can be written as

$$\text{cp}_{\mathcal{S}}(\mathbf{y}) = \mathbf{y} - \text{d}_{\mathcal{S}}(\mathbf{y}) \nabla \text{d}_{\mathcal{S}}(\mathbf{y}). \quad (3.9)$$

We must therefore compute $\nabla \text{d}_{\mathcal{N}}$ to construct $\text{cp}_{\mathcal{N}}$ and then calculate the component-wise Hessian of $\text{cp}_{\mathcal{N}}$. The signed distance function for the unit hypersphere is

$$\text{d}_{\mathcal{N}}(\mathbf{y}) = \|\mathbf{y}\|_2 - 1, \text{ for all } \mathbf{y} \in \mathbb{R}^n, \quad (3.10)$$

which gives $\nabla \text{d}_{\mathcal{N}}(\mathbf{y}) = \mathbf{y}/\|\mathbf{y}\|_2$. Next, the gradient of the i -th component of $\text{cp}_{\mathcal{N}}$ is

$$\begin{aligned} \nabla \text{cp}_{\mathcal{N}}^i(\mathbf{y}) &= \nabla y^i - \nabla \text{d}_{\mathcal{N}}(\mathbf{y}) \frac{\partial}{\partial y^i} \text{d}_{\mathcal{N}}(\mathbf{y}) - \text{d}_{\mathcal{N}}(\mathbf{y}) \nabla \left(\frac{\partial}{\partial y^i} \text{d}_{\mathcal{N}}(\mathbf{y}) \right), \\ &= \nabla y^i - y^i \frac{\mathbf{y}}{\|\mathbf{y}\|_2^2} - (\|\mathbf{y}\|_2 - 1) \frac{\nabla y^i}{\|\mathbf{y}\|_2}, \\ &= \frac{\nabla y^i}{\|\mathbf{y}\|_2} - y^i \frac{\mathbf{y}}{\|\mathbf{y}\|_2^2}. \end{aligned}$$

Through a similar calculation

$$\left\{ \mathbf{H}_{\text{cp}_{\mathcal{N}}}^i(\mathbf{y}) \right\}_{jk} = 3 \frac{y^i y^j y^k}{\|\mathbf{y}\|_2^5} - \frac{y^k \delta_{ij} + y^j \delta_{ik} + y^i \delta_{jk}}{\|\mathbf{y}\|_2^3}. \quad (3.11)$$

The target manifold constraint that $\mathbf{u} \in S^{n-1}$ simplifies (3.11) since $\|\mathbf{u}\|_2 = 1$. Another simplification that will be needed is an important identity to realize. One has that

$$0 = \partial d_{\mathcal{N}}(\mathbf{u}(\mathbf{x}))/\partial x_j = \nabla d_{\mathcal{N}}(\mathbf{u}(\mathbf{x})) \cdot \partial \mathbf{u}(\mathbf{x})/\partial x_j, \quad (3.12)$$

which is due to the fact that $d_{\mathcal{N}}(\mathbf{u}(\mathbf{x})) = 0$ for all $\mathbf{x} \in \mathcal{M}$ and thus $\partial d_{\mathcal{N}}(\mathbf{u}(\mathbf{x}))/\partial x_j = 0$. With the signed distance function for the unit hypersphere we have $\nabla d_{\mathcal{N}}(\mathbf{u}(\mathbf{x})) \cdot \partial \mathbf{u}(\mathbf{x})/\partial x^\alpha = \sum_j u^j (\partial u^j / \partial x^\alpha)$. Carrying out the matrix-vector multiplication in (3.6) yields

$$\begin{aligned} \sum_{\alpha} \mathbf{H}_{\text{cp}_{\mathcal{N}}(\mathbf{u})}^i \left[\frac{\partial \mathbf{u}}{\partial \mathbf{x}^\alpha}, \frac{\partial \mathbf{u}}{\partial \mathbf{x}^\alpha} \right] &= \sum_{\alpha} \sum_j \sum_k \left(3u^i u^j u^k - u^k \delta_{ij} - u^j \delta_{ik} - u^i \delta_{jk} \right) \frac{\partial u^j}{\partial x^\alpha} \frac{\partial u^k}{\partial x^\alpha}, \\ &= - \sum_{\alpha} \sum_j u^i \left(\frac{\partial u^j}{\partial x^\alpha} \right)^2, \\ &= -u^i \|\mathbf{J}_{\mathbf{u}}\|_{\mathcal{F}}^2. \end{aligned}$$

Substituting into (3.6) gives the Euler-Lagrange equations for liquid crystals [77]

$$\Delta \mathbf{u} + \|\mathbf{J}_{\mathbf{u}}\|_{\mathcal{F}}^2 \mathbf{u} = 0.$$

3.1.2 Geodesic computation on implicit manifolds using harmonic maps

As mentioned at the start of this chapter, geodesics of the target manifold \mathcal{N} satisfy the Euler-Lagrange equations (3.3) for harmonic maps when $\dim(\mathcal{M}) = 1$. Geodesic maps ensue when $\dim(\mathcal{M}) = 1$ because (3.3) simplifies to (3.4). The geodesics on manifolds given by a closest point representation will therefore satisfy (3.6). That is, geodesics (parameterized by arc-length) on \mathcal{N} are the solution to

$$\ddot{u}^k - \sum_{i,j=1}^n \mathbf{H}_{\text{cp}_{\mathcal{N}}(\mathbf{u})}^k \dot{u}^i \dot{u}^j = 0. \quad (3.13)$$

This equation could be used to obtain geodesic curves on \mathcal{N} , which is an important problem that arises in numerous applications.

3.1.3 A note on general variational problems and partial differential equations

More general PDEs that map between manifolds can be embedded by rewriting all intrinsic geometric quantities in terms of the appropriate closest point functions. This can

be accomplished using Principles 1 and 2 for quantities on \mathcal{M} combined with identities in [57] for \mathcal{N} , e.g., the identity (3.7) for the second fundamental form of \mathcal{N} . It is difficult to give steps to handle the most general of PDEs. One may need to derive new identities besides the ones mentioned here. The general idea is however the same in all cases: rewrite geometric quantities intrinsic to \mathcal{M} and \mathcal{N} in terms of $\text{cp}_{\mathcal{M}}$ and $\text{cp}_{\mathcal{N}}$, respectively.

For a concrete example, consider what was done above for harmonic maps. When using an explicit representation of the manifolds (i.e., local coordinates) the Euler-Lagrange equations for harmonic maps are given by (3.3). Using Principles 1 and 2 one can rewrite $\Delta_{\mathcal{M}}\mathbf{u} = \Delta\mathbf{u}(\text{cp}_{\mathcal{M}})$. Then using the identity (3.7) one rewrites the Christoffel symbols in (3.3): $\Gamma(\mathbf{u})[\partial_{x^\alpha}\mathbf{u}, \partial_{x^\beta}\mathbf{u}] = -\mathbf{H}_{\text{cp}_{\mathcal{N}}(\mathbf{u})}[\partial_{x^\alpha}\mathbf{u}, \partial_{x^\beta}\mathbf{u}]$. The final component to change from an explicit manifold representation to an implicit one, is to associate the metric tensor on \mathcal{M} with the projection operator onto the tangent space of \mathcal{M} , i.e., $\gamma^{\alpha\beta} = \{\Pi_{T_{\mathbf{u}}\mathcal{M}}\}_{\alpha\beta}$.

It is sometimes also possible to simplify the method by realizing a different form for the PDE of interest. The next section gives details for the case of harmonic maps between general manifolds \mathcal{M} and \mathcal{N} . In this case the Euler-Lagrange equations (3.8) are equivalent to $\Pi_{T_{\mathbf{u}}\mathcal{N}}(\Delta_{\mathcal{M}}\mathbf{u}) = 0$.

3.2 Projections onto target manifold \mathcal{N}

A numerical method for harmonic maps is realized by rewriting the Euler-Lagrange equations (3.8), or equivalently (3.3), in a different form. The Euler-Lagrange equations are rewritten as $\Pi_{T_{\mathbf{u}}\mathcal{N}}(\Delta_{\mathcal{M}}\mathbf{u}) = 0$ [54], where $\Pi_{T_{\mathbf{u}}\mathcal{N}}$ is the projection operator at the point \mathbf{y} onto the tangent space of \mathcal{N} . As mentioned in [66], the second term in (3.3) is by definition $(\Delta_{\mathcal{M}}\mathbf{u})^\perp$, the normal component of $\Delta_{\mathcal{M}}\mathbf{u}$. One therefore realizes that

$$0 = \Delta_{\mathcal{M}}\mathbf{u} - (\Delta_{\mathcal{M}}\mathbf{u})^\perp = (\Delta_{\mathcal{M}}\mathbf{u})^\text{T},$$

where $(\Delta_{\mathcal{M}}\mathbf{u})^\text{T}$ is the tangential component of $\Delta_{\mathcal{M}}\mathbf{u}$. Hence $\Pi_{T_{\mathbf{u}}\mathcal{N}}(\Delta_{\mathcal{M}}\mathbf{u}) = 0$.

When \mathcal{M} is a flat, open manifold in \mathbb{R}^m it is easy to see that the second term in (3.6) is the normal component. First notice that for any $Y \in T_{\mathbf{u}}\mathcal{N}$

$$\Pi_{T_{\mathbf{u}}\mathcal{N}}(Y) = Y.$$

It is shown in both [52, 57] that the Jacobian of the closest point function is equal to $\Pi_{T_{\mathbf{u}}\mathcal{N}}$ at the surface of \mathcal{N} . Hence, one has that

$$\mathbf{J}_{\text{cp}_{\mathcal{N}}(\mathbf{u})} \frac{\partial \mathbf{u}}{\partial x^\alpha} = \frac{\partial \mathbf{u}}{\partial x^\alpha}.$$

Differentiating the above in the direction of $\frac{\partial \mathbf{u}}{\partial x^\alpha}$ gives

$$\mathbf{H}_{\text{cp}_{\mathcal{N}}(\mathbf{u})} \left[\frac{\partial \mathbf{u}}{\partial x^\alpha}, \frac{\partial \mathbf{u}}{\partial x^\alpha} \right] + \mathbf{J}_{\text{cp}_{\mathcal{N}}(\mathbf{u})} \Delta_{\mathcal{M}} \mathbf{u} = \Delta_{\mathcal{M}} \mathbf{u}, \quad (3.14)$$

as shown by Moser [57] during the derivation of (3.7). Now from (3.14) it is apparent that

$$\mathbf{H}_{\text{cp}_{\mathcal{N}}(\mathbf{u})} \left[\frac{\partial \mathbf{u}}{\partial x^\alpha}, \frac{\partial \mathbf{u}}{\partial x^\alpha} \right] = (\Delta_{\mathcal{M}} \mathbf{u})^\perp$$

since $\mathbf{J}_{\text{cp}_{\mathcal{N}}(\mathbf{u})} \Delta_{\mathcal{M}} \mathbf{u} = \Pi_{T_{\mathbf{u}}\mathcal{N}}(\Delta_{\mathcal{M}} \mathbf{u}) = (\Delta_{\mathcal{M}} \mathbf{u})^\top$.

The corresponding gradient descent flow is

$$\begin{cases} \frac{\partial \mathbf{u}}{\partial t} = \Pi_{T_{\mathbf{u}}\mathcal{N}}(\Delta_{\mathcal{M}} \mathbf{u}), \\ \mathbf{u}(\mathbf{x}, 0) = \mathbf{u}_0(\mathbf{x}), \\ \mathbf{J}_{\mathbf{u}}^{\text{d}_{\mathcal{M}}} \mathbf{n}|_{\partial \mathcal{M}} = 0, \end{cases} \quad (3.15)$$

where $\mathbf{u}_0(\mathbf{x})$ is a given initial map. The Neumann boundary conditions are justified in Appendix A of [54]. Our idea is to construct a numerical method for (3.15) by splitting the evolution into two steps. First, evolve the gradient descent flow solely on \mathcal{M} for one time step of size ε to give $\tilde{\mathbf{u}}(\mathbf{x}, t + \varepsilon)$. That is, solve (3.15) while omitting $\Pi_{T_{\mathbf{u}}\mathcal{N}}$. The second step obtains the solution to (3.15), at time $t + \varepsilon$, by projecting $\tilde{\mathbf{u}}(\mathbf{x}, t + \varepsilon)$ onto \mathcal{N} using the closest points of $\tilde{\mathbf{u}}(\mathbf{x}, t + \varepsilon)$ on \mathcal{N} . The explicit steps are given in Algorithm 3.1. Note that because

Algorithm 3.1 Closest point method for harmonic maps

1. Using the closest point method for a single manifold solve

$$\begin{cases} \frac{\partial \tilde{\mathbf{u}}}{\partial t} = \Delta_{\mathcal{M}} \tilde{\mathbf{u}}, \\ \tilde{\mathbf{u}}(\mathbf{x}, t) = \mathbf{u}(\mathbf{x}, t), \\ \mathbf{J}_{\tilde{\mathbf{u}}}^{\text{d}_{\mathcal{M}}} \mathbf{n}|_{\partial \mathcal{M}} = 0, \end{cases} \quad (3.16)$$

for one time step of size ε .

2. Set $\mathbf{u}(\mathbf{x}, t + \varepsilon) = \text{cp}_{\mathcal{N}}(\tilde{\mathbf{u}}(\mathbf{x}, t + \varepsilon))$.
-

$\mathbf{u}(\mathbf{x}, t)$ is the solution computed at the previous time step it satisfies $\text{d}_{\mathcal{N}}(\mathbf{u}(\mathbf{x}, t)) = 0$ and $\text{cp}_{\mathcal{N}}(\mathbf{u}(\mathbf{x}, t)) = \mathbf{u}(\mathbf{x}, t)$ for all $\mathbf{x} \in \mathcal{M}$.

Solving (3.15) by splitting the evolution into the two steps of Algorithm 3.1 is advantageous because it eliminates the projection operator $\Pi_{T_{\mathbf{u}\mathcal{N}}}$. This simplifies solving (3.15) by reducing it to a PDE on a single surface \mathcal{M} ; now the PDE does not involve quantities on both \mathcal{M} and \mathcal{N} . A further benefit is efficiency due to not building a discrete form of the projection operator $\Pi_{T_{\mathbf{u}\mathcal{N}}}$ at every time step. The PDE (3.16) is also more simple in form compared to the PDE obtained from the level set approach of Mémoli et al. [54]. The level set method's PDE involves two projection operators $\Pi_{\nabla\phi}$ and $\Pi_{\nabla\psi}$, which can be written in terms of the level set function ϕ and ψ representing \mathcal{M} and \mathcal{N} , respectively. These projections operators are undesirable since they complicate the numerical stability of the method.

This method can be applied to any PDE of the form

$$\frac{\partial \mathbf{u}}{\partial t} = \eta \Pi_{T_{\mathbf{u}\mathcal{N}}}(\mathbf{F}(\mathbf{x}, \mathbf{u}, \mathbf{J}_{\mathbf{u}}^{\mathbf{d}\mathcal{M}}, \mathbf{H}_{\mathbf{u}}^{\mathbf{d}\mathcal{M}}, \dots)), \quad (3.17)$$

where $\eta \in \mathbb{R}$ is a scalar. That is, our splitting approach in Algorithm 3.1 can solve PDEs with intrinsic geometric terms on \mathcal{M} that are projected onto the tangent space of \mathcal{N} . In this situation, one solves the PDE

$$\partial \tilde{\mathbf{u}} / \partial t = \eta \mathbf{F}(\mathbf{x}, \tilde{\mathbf{u}}, \mathbf{J}_{\tilde{\mathbf{u}}}^{\mathbf{d}\mathcal{M}}, \mathbf{H}_{\tilde{\mathbf{u}}}^{\mathbf{d}\mathcal{M}}, \dots)$$

for one time step with the closest point method for a single manifold (see Chapter 2). This is followed by the projection step $\mathbf{u}(\mathbf{x}, t + \varepsilon) = \text{cp}_{\mathcal{N}}(\tilde{\mathbf{u}}(\mathbf{x}, t + \varepsilon))$. The algorithm for solving (3.17) is given in Algorithm 3.2. It is shown in Section 3.3.2 that these two steps

Algorithm 3.2 Closest point method for $\partial \mathbf{u} / \partial t = \eta \Pi_{T_{\mathbf{u}\mathcal{N}}}(\mathbf{F})$

1. Using the closest point method for a single manifold solve

$$\begin{cases} \frac{\partial \tilde{\mathbf{u}}}{\partial t} = \eta \mathbf{F}(\mathbf{x}, \tilde{\mathbf{u}}, \mathbf{J}_{\tilde{\mathbf{u}}}^{\mathbf{d}\mathcal{M}}, \mathbf{H}_{\tilde{\mathbf{u}}}^{\mathbf{d}\mathcal{M}}, \dots), \\ \tilde{\mathbf{u}}(\mathbf{x}, t) = \mathbf{u}(\mathbf{x}, t), \\ \mathbf{J}_{\tilde{\mathbf{u}}}^{\mathbf{d}\mathcal{M}} \mathbf{n}|_{\partial \mathcal{M}} = 0, \end{cases}$$

for one time step of size ε .

2. Set $\mathbf{u}(\mathbf{x}, t + \varepsilon) = \text{cp}_{\mathcal{N}}(\tilde{\mathbf{u}}(\mathbf{x}, t + \varepsilon))$.
-

are equivalent to the original PDE (3.17) as $\varepsilon \rightarrow 0$.

A PDE that is a special case of (3.17) comes from the gradient descent flow for p -harmonic maps.

Definition 4 *The critical points $\mathbf{u} : \mathcal{M} \rightarrow \mathcal{N}$ of the energy*

$$E_p[\mathbf{u}] = \int_{\mathcal{M}} e_p[\mathbf{u}] dv_{\mathcal{M}}, \quad (3.18)$$

with $1 \leq p < \infty$ and

$$e_p[\mathbf{u}] = \frac{1}{p} \|\mathbf{J}_{\mathbf{u}}^{d_{\mathcal{M}}}\|_{\mathcal{F}}^p,$$

are called *p-harmonic maps*.

There has been less work on the study of existence, uniqueness and regularity of *p*-harmonic maps when $p \neq 2$. Some examples are works by Chen et al. [23] and Giaquinta et al. [38]. Chen et al. [23] take $1 < p < \infty$ and assume that \mathcal{M} is a compact Riemannian manifold without boundary and \mathcal{N} is the unit sphere S^{n-1} . The work of Giaquinta et al. [38] handles the case $p = 1$ mapping onto $\mathcal{N} = S^1$.

The gradient descent flow for the Euler-Lagrange equations of (3.18), with $1 \leq p < \infty$, is [54]

$$\begin{cases} \frac{\partial \mathbf{u}}{\partial t} = p^{1-\frac{2}{p}} \Pi_{T_{\mathbf{u}}\mathcal{N}} \left(\nabla \cdot \left((e_p[\mathbf{u}])^{1-\frac{2}{p}} \mathbf{J}_{\mathbf{u}}^{d_{\mathcal{M}}} \right) \right), \\ \mathbf{u}(\mathbf{x}, 0) = \mathbf{u}_0(\mathbf{x}), \\ \mathbf{J}_{\mathbf{u}}^{d_{\mathcal{M}}} \mathbf{n}|_{\partial \mathcal{M}} = 0, \end{cases} \quad (3.19)$$

which can be handled using Algorithm 3.2. Note that the divergence of the matrix here is defined as the divergence of each row of the matrix. Numerical implementation is more involved for (3.19) and difficulties are expected to arise for $p < 2$ [76]. Numerical implementation with $p = 1$ for the application of colour image enhancement [74] is discussed in Chapter 4. The colour image enhancement is achieved by diffusing the chromaticity of the image, which is a map from $\mathcal{M} \subset \mathbb{R}^2$ to $\mathcal{N} = S^{n-1}$.

3.2.1 Applications in medical imaging

There is immense use of maps between surfaces in medical imaging. The maps obtain a correspondence between two surfaces to enable comparison. Initially, images are acquired using one of a number of possible methods such as radiography, tomography, magnetic resonance imaging (MRI), etc. The anatomical structure of interest in each image is then segmented to produce a population of surfaces. A reference surface can be chosen at random for comparison. A map from each surface in the population to the reference is created. The map allows matching of functional and/or anatomical regions to the reference. Using the

maps one can then describe specific subpopulations of the surfaces, measure their variability and/or characterize structural differences [75].

The variability in shape and size of anatomical structures, and the convoluted nature of some, cause numerical difficulties for classical surface matching methods such as the iterative closest point method [18]. Therefore mappings that fall into this problem category are conventionally handled using an intermediate parametrization to the plane (or the sphere). That is, one assigns a 2D coordinate to each point on the surfaces by mapping to the plane (or sphere) first. These 2D coordinates then allow for indirect mapping between the two surfaces via mappings between planes (or spheres).

For closed surfaces the difficulty with intermediate parameterizations to the plane is that one must artificially cut open the surface to flatten it. A standard approach for cutting open surfaces is hard to establish with surfaces that have varying shapes and sizes. Another disadvantage of intermediate parameterizations is that two surfaces will generally have different parameterizations, leading to different coordinates systems. In this situation one has to perform a warping procedure while enforcing anatomically meaningful constraints. For more information see [68] and references within.

Shi et al. [68] overcome the difficulties due to an intermediate parametrization by abandoning this step completely. They introduce a direct surface mapping method for MRI images of the brain cortex. The map is constructed by solving PDEs that define maps between the source and target cortices using the level set approach of Mémoli et al. [54]. Instead one can apply the closest point method introduced above to the cortical mapping problem. Numerical discretization schemes must however be developed to incorporate constraints of anatomical features. One anatomical constraint feature could be the curves that follow the “valley” structure of the cortex, called sulcal curves. One approach to handle sulcal curves is given in [68].

One could also use the same ideas for registering optic nerve head images. A method which performs an intermediate parametrization to the sphere was introduced by Gibson et al. [39]. The optic nerve head is not convoluted like the cortex (compare surfaces in Figure 3.1). Algorithm 3.1 can therefore be immediately used to construct a harmonic map between optic nerve head surfaces. That is, specialized numerical schemes to handle structures similar to sulcal curves are not needed since the surface is not highly convoluted.

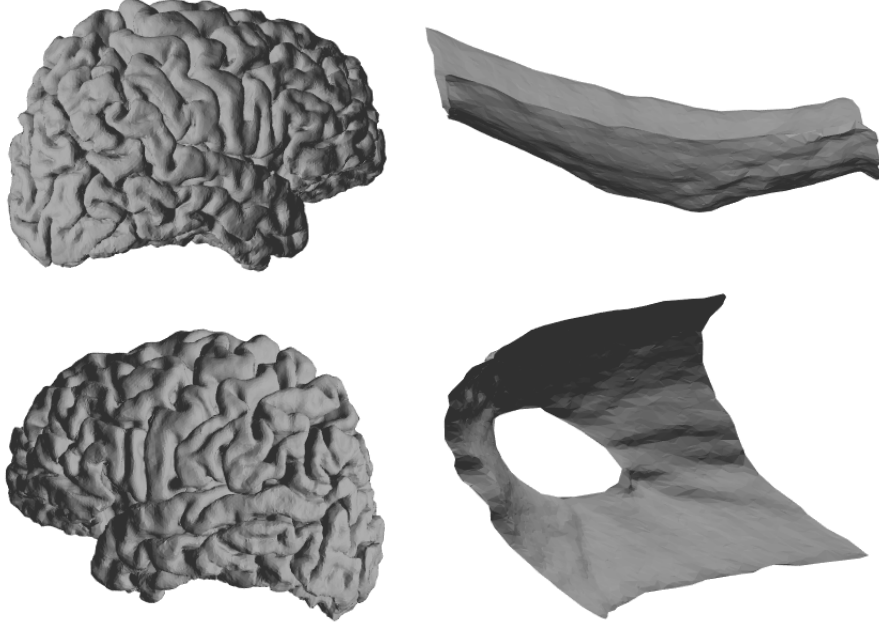


Figure 3.1: Comparison of anatomical surfaces of the brain (left) and optic nerve head (right). Notice the convoluted nature of the brain compared to the optic nerve head, which is much smoother. Anatomical surface data courtesy of the Medical Image Analysis Lab at Simon Fraser University.

It is also possible to include data fidelity terms with the harmonic energy to help enforce anatomical features. Consider feature functions $f_1 : \mathcal{M} \rightarrow \mathbb{R}$ on the source manifold and $f_2 : \mathcal{N} \rightarrow \mathbb{R}$ on the target manifold. A weight function $w : \mathcal{M} \rightarrow \mathbb{R}$ can also be included for further generality. The new energy we wish to minimize is now defined as

$$E[\mathbf{u}] = \frac{1}{2} \int_{\mathcal{M}} \|\mathbf{J}_{\mathbf{u}}^{\mathcal{M}}\|_{\mathcal{F}}^2 dv_{\mathcal{M}} + \frac{\lambda}{2} \int_{\mathcal{M}} w(f_1 - f_2(\mathbf{u}))^2 dv_{\mathcal{M}}, \quad (3.20)$$

where λ is a positive regularization parameter. The gradient descent flow corresponding to (3.20) is given in [68] as

$$\frac{\partial \mathbf{u}}{\partial t} = \Pi_{T_{\mathbf{u}}\mathcal{N}}(\Delta_{\mathcal{M}}\mathbf{u}) + \lambda w(f_1 - f_2(\mathbf{u}))\nabla_{\mathcal{N}}f_2(\mathbf{u}). \quad (3.21)$$

The numerical solution of (3.21) can be computed using a time splitting scheme (a.k.a. fractional step method) [49]. Time splitting is a simple idea similar to the evolution splitting used in Section 3.2. Consider the PDE

$$\frac{\partial \mathbf{v}}{\partial t} = \mathcal{A}(\mathbf{v}) + \mathcal{B}(\mathbf{v}). \quad (3.22)$$

Instead of solving (3.22) directly, one can split the PDE into subproblems $\partial \mathbf{v} / \partial t = \mathcal{A}(\mathbf{v})$ and $\partial \mathbf{v} / \partial t = \mathcal{B}(\mathbf{v})$ and then combine their solutions. A first order time splitting scheme

for (3.22) is

$$\begin{cases} \mathbf{v}^* = \Phi_{\mathcal{A}}(\mathbf{v}(\mathbf{x}, t), \varepsilon), \\ \mathbf{v}(\mathbf{x}, t + \varepsilon) = \Phi_{\mathcal{B}}(\mathbf{v}^*, \varepsilon), \end{cases} \quad (3.23)$$

where $\Phi_{\mathcal{A}}$ and $\Phi_{\mathcal{B}}$ are one step numerical methods, with step size ε , for subproblems \mathcal{A} and \mathcal{B} , respectively.

To solve (3.21) take $\mathcal{A}(\mathbf{u}) = \Pi_{T_{\mathbf{u}}\mathcal{N}}(\Delta_{\mathcal{M}}\mathbf{u})$ and $\mathcal{B}(\mathbf{u}) = \lambda w(f_1 - f_2(\mathbf{u}))\nabla_{\mathcal{N}}f_2(\mathbf{u})$. Algorithm 3.1 can then be used as the one step method $\Phi_{\mathcal{A}}$. To construct $\Phi_{\mathcal{B}}$ the closest point method for a single manifold (see Chapter 2) can be used. Note that general theory for higher order time splitting methods, e.g., second order Strang splitting, is presented in [44]. The first order method (3.23) is however sufficient for the numerical solution of (3.21) since Algorithm 3.1 is only first order accurate (see Section 3.3.2).

3.3 Justification of the approach

Here we justify the two steps in Algorithms 3.1 and 3.2 used to solve (3.15) or (3.17), respectively. The first result is that the solution from Algorithm 3.1 decreases the harmonic energy (3.5) in the limit as $\varepsilon \rightarrow 0$. The two steps in Algorithm 3.2 are also shown to be equivalent to the original PDE (3.17) as $\varepsilon \rightarrow 0$. Taylor's theorem will be needed for both of these proofs. Both functions from \mathbb{R} to \mathbb{R}^n and functions from \mathbb{R}^m to \mathbb{R}^n will be expanded using Taylor's theorem.

It is necessary to introduce some notation for Taylor's theorem. For two functions \mathbf{r} and \mathbf{s} one has that $\mathbf{r}(\mathbf{y}) = \mathcal{O}(\mathbf{s}(\mathbf{y}))$ as $\mathbf{y} \rightarrow \mathbf{z}$ if there exists positive constants $M, \delta \in \mathbb{R}$ such that

$$\|\mathbf{r}\| \leq M\|\mathbf{s}\|, \quad \text{for } \|\mathbf{y} - \mathbf{z}\| < \delta.$$

Another piece of notation is used to generalize derivatives and the vector-matrix multiplication needed for functions $\mathbf{f} : \mathbb{R}^m \rightarrow \mathbb{R}^n$. For vectors

$$\mathbf{h}_1 = \begin{bmatrix} h_{1,1} \\ h_{1,2} \\ \vdots \\ h_{1,m} \end{bmatrix}, \quad \mathbf{h}_2 = \begin{bmatrix} h_{2,1} \\ h_{2,2} \\ \vdots \\ h_{2,m} \end{bmatrix}, \quad \dots, \quad \mathbf{h}_k = \begin{bmatrix} h_{k,1} \\ h_{k,2} \\ \vdots \\ h_{k,m} \end{bmatrix},$$

we write

$$\mathbf{f}^{(k)}[\mathbf{h}_1, \mathbf{h}_2, \dots, \mathbf{h}_k] = \sum_{i_1, i_2, \dots, i_k=1}^m \frac{\partial^k \mathbf{f}}{\partial x^1 \partial x^2 \dots \partial x^k} h_{1, i_1} h_{2, i_2} \dots h_{k, i_k},$$

where the sum is over all sequences (i_1, i_2, \dots, i_k) , with $i_j \in \{1, 2, \dots, m\}$.

Taylor's theorem for the more general case, $\mathbf{f} : \mathbb{R}^m \rightarrow \mathbb{R}^n$, is stated as [26]:

Theorem 1 *Let A and B be normed vector spaces and O an open subset of A . Suppose that $\mathbf{x} \in O$ and $\mathbf{h} \in A$ such that $[\mathbf{x}, \mathbf{x} + \mathbf{h}] \subseteq O$. If $\mathbf{f} : O \rightarrow B$ is a C^{k+1} mapping, then*

$$\mathbf{f}(\mathbf{x} + \mathbf{h}) = \mathbf{f}(\mathbf{x}) + \mathbf{f}^{(1)}(\mathbf{x})[\mathbf{h}] + \frac{1}{2!}\mathbf{f}^{(2)}(\mathbf{x})[\mathbf{h}^2] + \dots + \frac{1}{k!}\mathbf{f}^{(k)}(\mathbf{x})[\mathbf{h}^k] + \mathcal{O}(\|\mathbf{h}\|^{k+1}), \quad (3.24)$$

where $[\mathbf{h}^k]$ denotes $[\mathbf{h}, \mathbf{h}, \dots, \mathbf{h}]$ with \mathbf{h} repeated k times. We call (3.24) the Taylor expansion of $\mathbf{f}(\mathbf{x} + \mathbf{h})$ about \mathbf{x} .

Consider $A = \mathbb{R}$ and $B = \mathbb{R}^n$, then (3.24) simplifies to

$$\mathbf{f}(x + h) = \mathbf{f}(x) + h \frac{d\mathbf{f}}{dx}(x) + \frac{h^2}{2!} \frac{d^2\mathbf{f}}{dx^2}(x) + \dots + \frac{h^k}{k!} \frac{d^k\mathbf{f}}{dx^k}(x) + \mathcal{O}(|h|^{k+1}). \quad (3.25)$$

There is a final notational simplification that one should realize. When $A = \mathbb{R}^m$ and $B = \mathbb{R}^n$, the first derivative and second derivative terms in (3.24) are the Jacobian and Hessian of \mathbf{f} , respectively. One therefore has that

$$\mathbf{f}^{(1)}(\mathbf{x})[\mathbf{h}] = \mathbf{J}_{\mathbf{f}(\mathbf{x})} \mathbf{h} \quad \text{and} \quad \mathbf{f}^{(2)}(\mathbf{x})[\mathbf{h}^2] = \mathbf{H}_{\mathbf{f}(\mathbf{x})}[\mathbf{h}, \mathbf{h}]. \quad (3.26)$$

With Taylor's theorem introduced we are now ready to prove the justifications of Algorithms 3.1 and 3.2.

3.3.1 Decrease of harmonic energy

We want to show that the solution computed by Algorithm 3.1, introduced in Section 3.2, decreases the harmonic energy (3.5) as $\varepsilon \rightarrow 0$. This can be accomplished by showing that

$$\frac{d}{dt} E[\mathbf{u}(\mathbf{x}, t)] = \lim_{\varepsilon \rightarrow 0} \frac{1}{2\varepsilon} \int_{\mathcal{M}} \left(\|\mathbf{J}_{\mathbf{u}(\mathbf{x}, t+\varepsilon)}^{\mathbf{d}_{\mathcal{M}}}\|_{\mathcal{F}}^2 - \|\mathbf{J}_{\mathbf{u}(\mathbf{x}, t)}^{\mathbf{d}_{\mathcal{M}}}\|_{\mathcal{F}}^2 \right) dv_{\mathcal{M}} \leq 0. \quad (3.27)$$

First Taylor expand $\tilde{\mathbf{u}}(\mathbf{x}, t + \varepsilon)$ in t and substitute $\partial_t \tilde{\mathbf{u}} = \Delta_{\mathcal{M}} \tilde{\mathbf{u}}$ from Step 1 of Algorithm 3.1 to give

$$\tilde{\mathbf{u}}(\mathbf{x}, t + \varepsilon) = \tilde{\mathbf{u}}(\mathbf{x}, t) + \varepsilon \Delta_{\mathcal{M}} \tilde{\mathbf{u}}(\mathbf{x}, t) + \mathcal{O}(\varepsilon^2).$$

Note that the latter Taylor expansion is the special case (3.25) when $A = \mathbb{R}$. Consequently, the map $\mathbf{u}(\mathbf{x}, t)$ must be assumed to be a C^2 map. Also recall the closest point function

can be expressed as in (3.9). The solution $\mathbf{u}(\mathbf{x}, t + \varepsilon)$ can then be written as

$$\begin{aligned}\mathbf{u}(\mathbf{x}, t + \varepsilon) &= \text{cp}_{\mathcal{N}}(\tilde{\mathbf{u}}(\mathbf{x}, t + \varepsilon)), \\ &= \tilde{\mathbf{u}}(\mathbf{x}, t + \varepsilon) - \text{d}_{\mathcal{N}}\left(\tilde{\mathbf{u}}(\mathbf{x}, t + \varepsilon)\right) \nabla \text{d}_{\mathcal{N}}\left(\tilde{\mathbf{u}}(\mathbf{x}, t + \varepsilon)\right), \\ &= \mathbf{u}(\mathbf{x}, t) + \varepsilon \Delta_{\mathcal{M}} \mathbf{u}(\mathbf{x}, t) + \mathcal{O}(\varepsilon^2) - \text{d}_{\mathcal{N}}\left(\mathbf{u}(\mathbf{x}, t) + \varepsilon \Delta_{\mathcal{M}} \mathbf{u}(\mathbf{x}, t) + \mathcal{O}(\varepsilon^2)\right) \\ &\quad \times \nabla \text{d}_{\mathcal{N}}\left(\mathbf{u}(\mathbf{x}, t) + \varepsilon \Delta_{\mathcal{M}} \mathbf{u}(\mathbf{x}, t) + \mathcal{O}(\varepsilon^2)\right),\end{aligned}$$

where we have also used $\tilde{\mathbf{u}}(\mathbf{x}, t) = \mathbf{u}(\mathbf{x}, t)$ from Step 1 of Algorithm 3.1.

The columns of $\mathbf{J}_{\mathbf{u}(\mathbf{x}, t + \varepsilon)}$ are

$$\begin{aligned}\frac{\partial \mathbf{u}(\mathbf{x}, t + \varepsilon)}{\partial x_j} &= \left(\frac{\partial \mathbf{u}}{\partial x_j} + \varepsilon \frac{\partial(\Delta_{\mathcal{M}} \mathbf{u})}{\partial x_j} + \mathcal{O}(\varepsilon^2) \right) \\ &\quad - \left(\nabla \text{d}_{\mathcal{N}}\left(\mathbf{u} + \varepsilon \Delta_{\mathcal{M}} \mathbf{u} + \mathcal{O}(\varepsilon^2)\right) \cdot \left[\frac{\partial \mathbf{u}}{\partial x_j} + \varepsilon \frac{\partial(\Delta_{\mathcal{M}} \mathbf{u})}{\partial x_j} + \mathcal{O}(\varepsilon^2) \right] \right) \\ &\quad \times \nabla \text{d}_{\mathcal{N}}\left(\mathbf{u} + \varepsilon \Delta_{\mathcal{M}} \mathbf{u} + \mathcal{O}(\varepsilon^2)\right) - \text{d}_{\mathcal{N}}\left(\mathbf{u} + \varepsilon \Delta_{\mathcal{M}} \mathbf{u} + \mathcal{O}(\varepsilon^2)\right) \\ &\quad \times \mathbf{H}_{\text{d}_{\mathcal{N}}(\mathbf{u} + \varepsilon \Delta_{\mathcal{M}} \mathbf{u} + \mathcal{O}(\varepsilon^2))} \left(\frac{\partial \mathbf{u}}{\partial x_j} + \varepsilon \frac{\partial(\Delta_{\mathcal{M}} \mathbf{u})}{\partial x_j} + \mathcal{O}(\varepsilon^2) \right).\end{aligned}\tag{3.28}$$

Note that we have dropped the dependence of \mathbf{u} since all terms on the right hand side are evaluated at (\mathbf{x}, t) . Ultimately we will take $\varepsilon \rightarrow 0$, so the final term in (3.28) will vanish since $\text{d}_{\mathcal{N}}(\mathbf{u}(\mathbf{x}, t)) = 0$. Note however that other terms involving ε can not be taken to zero yet since we divide by ε in (3.27). Equation (3.28) is simplified further using the identity (3.12), $\nabla \text{d}_{\mathcal{N}}(\mathbf{u}(\mathbf{x}, t)) \cdot \partial \mathbf{u}(\mathbf{x}, t) / \partial x_j = 0$. With these two simplifications the columns of $\mathbf{J}_{\mathbf{u}(\mathbf{x}, t + \varepsilon)}$ become

$$\frac{\partial \mathbf{u}(\mathbf{x}, t + \varepsilon)}{\partial x_j} = \frac{\partial \mathbf{u}}{\partial x_j} + \varepsilon \frac{\partial(\Delta_{\mathcal{M}} \mathbf{u})}{\partial x_j} + \mathcal{O}(\varepsilon^2) - \varepsilon \mu_j(\varepsilon) \nabla \text{d}_{\mathcal{N}}\left(\mathbf{u} + \varepsilon \Delta_{\mathcal{M}} \mathbf{u} + \mathcal{O}(\varepsilon^2)\right),\tag{3.29}$$

where we denote

$$\mu_j(\varepsilon) \equiv \nabla \text{d}_{\mathcal{N}}\left(\mathbf{u} + \varepsilon \Delta_{\mathcal{M}} \mathbf{u} + \mathcal{O}(\varepsilon^2)\right) \cdot \frac{\partial(\Delta_{\mathcal{M}} \mathbf{u})}{\partial x_j},$$

to help simplify calculations below.

Now that we know $\mathbf{J}_{\mathbf{u}(\mathbf{x}, t + \varepsilon)}$ we can compute $\mathbf{J}_{\mathbf{u}(\mathbf{x}, t + \varepsilon)}^{\text{d}_{\mathcal{M}}}$ since $\mathbf{J}_{\mathbf{u}(\mathbf{x}, t + \varepsilon)}^{\text{d}_{\mathcal{M}}} = \mathbf{J}_{\mathbf{u}(\mathbf{x}, t + \varepsilon)} - \mathbf{J}_{\mathbf{u}(\mathbf{x}, t + \varepsilon)} \nabla \text{d}_{\mathcal{M}} \nabla \text{d}_{\mathcal{M}}^T$. The components of $\mathbf{J}_{\mathbf{u}(\mathbf{x}, t + \varepsilon)} \nabla \text{d}_{\mathcal{M}} \nabla \text{d}_{\mathcal{M}}^T$ are

$$\{\mathbf{J}_{\mathbf{u}(\mathbf{x}, t + \varepsilon)} \nabla \text{d}_{\mathcal{M}} \nabla \text{d}_{\mathcal{M}}^T\}_{ij} = \sum_{k=1}^m \{\mathbf{J}_{\mathbf{u}(\mathbf{x}, t + \varepsilon)}\}_{ik} \frac{\partial \text{d}_{\mathcal{M}}}{\partial x_k} \frac{\partial \text{d}_{\mathcal{M}}}{\partial x_j},$$

since $\{\nabla d_{\mathcal{M}} \nabla d_{\mathcal{M}}^T\}_{ij} = (\partial d_{\mathcal{M}} / \partial x_i)(\partial d_{\mathcal{M}} / \partial x_j)$ and matrix multiplication with $\mathbf{J}_{\mathbf{u}(\mathbf{x}, t+\varepsilon)}$ is written out in summation form. Hence the components of $\mathbf{J}_{\mathbf{u}(\mathbf{x}, t+\varepsilon)}^{\text{d}\mathcal{M}}$ are

$$\begin{aligned} \{\mathbf{J}_{\mathbf{u}(\mathbf{x}, t+\varepsilon)}^{\text{d}\mathcal{M}}\}_{ij} &= \{\mathbf{J}_{\mathbf{u}(\mathbf{x}, t+\varepsilon)}\}_{ij} - \sum_{k=1}^m \{\mathbf{J}_{\mathbf{u}(\mathbf{x}, t+\varepsilon)}\}_{ik} \frac{\partial d_{\mathcal{M}}}{\partial x_k} \frac{\partial d_{\mathcal{M}}}{\partial x_j}, \\ &= \left(\frac{\partial u^i}{\partial x_j} + \varepsilon \frac{\partial(\Delta_{\mathcal{M}} u^i)}{\partial x_j} + \mathcal{O}(\varepsilon^2) - \varepsilon \mu_j(\varepsilon) \frac{\partial}{\partial x_i} d_{\mathcal{N}}(\mathbf{u} + \varepsilon \Delta_{\mathcal{M}} \mathbf{u} + \mathcal{O}(\varepsilon^2)) \right) \\ &\quad - \sum_{k=1}^m \left(\frac{\partial u^i}{\partial x_k} + \varepsilon \frac{\partial(\Delta_{\mathcal{M}} u^i)}{\partial x_k} + \mathcal{O}(\varepsilon^2) \right. \\ &\quad \left. - \varepsilon \mu_k(\varepsilon) \frac{\partial}{\partial x_i} d_{\mathcal{N}}(\mathbf{u} + \varepsilon \Delta_{\mathcal{M}} \mathbf{u} + \mathcal{O}(\varepsilon^2)) \right) \frac{\partial d_{\mathcal{M}}}{\partial x_k} \frac{\partial d_{\mathcal{M}}}{\partial x_j}, \end{aligned}$$

where we have used (3.29) to substitute for $\{\mathbf{J}_{\mathbf{u}(\mathbf{x}, t+\varepsilon)}\}_{ij}$. Analogously for $\mathbf{J}_{\mathbf{u}(\mathbf{x}, t)}^{\text{d}\mathcal{M}}$ we have components

$$\begin{aligned} \{\mathbf{J}_{\mathbf{u}(\mathbf{x}, t)}^{\text{d}\mathcal{M}}\}_{ij} &= \{\mathbf{J}_{\mathbf{u}(\mathbf{x}, t)}\}_{ij} - \sum_{k=1}^m \{\mathbf{J}_{\mathbf{u}(\mathbf{x}, t)}\}_{ik} \frac{\partial d_{\mathcal{M}}}{\partial x_k} \frac{\partial d_{\mathcal{M}}}{\partial x_j}, \\ &= \frac{\partial u^i}{\partial x_j} - \sum_{k=1}^m \frac{\partial u^i}{\partial x_k} \frac{\partial d_{\mathcal{M}}}{\partial x_k} \frac{\partial d_{\mathcal{M}}}{\partial x_j}, \end{aligned}$$

since the components of $\mathbf{J}_{\mathbf{u}(\mathbf{x}, t)}$ are simply $\{\mathbf{J}_{\mathbf{u}(\mathbf{x}, t)}\}_{ij} = \partial u^i / \partial x_j$.

Now we need to compute both $\|\mathbf{J}_{\mathbf{u}(\mathbf{x}, t+\varepsilon)}^{\text{d}\mathcal{M}}\|_{\mathcal{F}}^2$ and $\|\mathbf{J}_{\mathbf{u}(\mathbf{x}, t)}^{\text{d}\mathcal{M}}\|_{\mathcal{F}}^2$. The rest of this derivation is quite tedious because the Frobenius norms of $\mathbf{J}_{\mathbf{u}(\mathbf{x}, t+\varepsilon)}^{\text{d}\mathcal{M}}$ and $\mathbf{J}_{\mathbf{u}(\mathbf{x}, t)}^{\text{d}\mathcal{M}}$ involve squaring the entries of the latter two Jacobians. Starting with the more difficult Jacobian, we have

$$\begin{aligned} \|\mathbf{J}_{\mathbf{u}(\mathbf{x}, t+\varepsilon)}^{\text{d}\mathcal{M}}\|_{\mathcal{F}}^2 &= \sum_{i=1}^m \sum_{j=1}^m |\{\mathbf{J}_{\mathbf{u}(\mathbf{x}, t+\varepsilon)}^{\text{d}\mathcal{M}}\}_{ij}|^2, \\ &= \sum_{i=1}^m \sum_{j=1}^m \left\{ \left(\frac{\partial u^i}{\partial x_j} + \varepsilon \frac{\partial(\Delta_{\mathcal{M}} u^i)}{\partial x_j} + \mathcal{O}(\varepsilon^2) \right. \right. \\ &\quad \left. \left. - \varepsilon \mu_j(\varepsilon) \frac{\partial}{\partial x_i} d_{\mathcal{N}}(\mathbf{u} + \varepsilon \Delta_{\mathcal{M}} \mathbf{u} + \mathcal{O}(\varepsilon^2)) \right)^2 \right. \\ &\quad + \left(\sum_{k=1}^m \left[\frac{\partial u^i}{\partial x_k} + \varepsilon \frac{\partial(\Delta_{\mathcal{M}} u^i)}{\partial x_k} + \mathcal{O}(\varepsilon^2) \right. \right. \\ &\quad \left. \left. - \varepsilon \mu_k(\varepsilon) \frac{\partial}{\partial x_i} d_{\mathcal{N}}(\mathbf{u} + \varepsilon \Delta_{\mathcal{M}} \mathbf{u} + \mathcal{O}(\varepsilon^2)) \right] \frac{\partial d_{\mathcal{M}}}{\partial x_k} \frac{\partial d_{\mathcal{M}}}{\partial x_j} \right)^2 \\ &\quad - 2 \left(\frac{\partial u^i}{\partial x_j} + \varepsilon \frac{\partial(\Delta_{\mathcal{M}} u^i)}{\partial x_j} + \mathcal{O}(\varepsilon^2) - \varepsilon \mu_j(\varepsilon) \frac{\partial}{\partial x_i} d_{\mathcal{N}}(\mathbf{u} + \varepsilon \Delta_{\mathcal{M}} \mathbf{u} + \mathcal{O}(\varepsilon^2)) \right) \\ &\quad \times \sum_{k=1}^m \left(\frac{\partial u^i}{\partial x_k} + \varepsilon \frac{\partial(\Delta_{\mathcal{M}} u^i)}{\partial x_k} + \mathcal{O}(\varepsilon^2) \right. \\ &\quad \left. - \varepsilon \mu_k(\varepsilon) \frac{\partial}{\partial x_i} d_{\mathcal{N}}(\mathbf{u} + \varepsilon \Delta_{\mathcal{M}} \mathbf{u} + \mathcal{O}(\varepsilon^2)) \right) \frac{\partial d_{\mathcal{M}}}{\partial x_k} \frac{\partial d_{\mathcal{M}}}{\partial x_j} \left. \right\}. \end{aligned}$$

To expand the remaining squares we use the identity $(a+b-c)^2 = a^2 + 2ab - 2ac + b^2 - 2bc + c^2$ and drop terms that are $\mathcal{O}(\varepsilon^2)$, i.e., drop $b^2 - 2bc + c^2$. Further multiplying the last term out and also dropping $\mathcal{O}(\varepsilon^2)$ terms gives

$$\begin{aligned}
\|\mathbf{J}_{\mathbf{u}(\mathbf{x},t+\varepsilon)}^{\mathbf{d}_{\mathcal{M}}}\|_{\mathcal{F}}^2 &\approx \sum_{i=1}^m \sum_{j=1}^m \left\{ \left(\frac{\partial u^i}{\partial x_j} \right)^2 + 2\varepsilon \frac{\partial u^i}{\partial x_j} \frac{\partial(\Delta_{\mathcal{M}} u^i)}{\partial x_j} - 2\varepsilon \mu_j(\varepsilon) \frac{\partial u^i}{\partial x_j} \frac{\partial}{\partial x_i} \mathbf{d}_{\mathcal{N}}(\mathbf{u} + \varepsilon \Delta_{\mathcal{M}} \mathbf{u}) \right. \\
&\quad + \left(\sum_{k=1}^m \frac{\partial u^i}{\partial x_k} \frac{\partial \mathbf{d}_{\mathcal{M}}}{\partial x_k} \frac{\partial \mathbf{d}_{\mathcal{M}}}{\partial x_j} \right)^2 + 2\varepsilon \left(\sum_{k=1}^m \frac{\partial u^i}{\partial x_k} \frac{\partial \mathbf{d}_{\mathcal{M}}}{\partial x_k} \frac{\partial \mathbf{d}_{\mathcal{M}}}{\partial x_j} \right) \\
&\quad \times \left(\sum_{l=1}^m \frac{\partial(\Delta_{\mathcal{M}} u^i)}{\partial x_l} \frac{\partial \mathbf{d}_{\mathcal{M}}}{\partial x_l} \frac{\partial \mathbf{d}_{\mathcal{M}}}{\partial x_j} \right) - 2\varepsilon \left(\sum_{k=1}^m \frac{\partial u^i}{\partial x_k} \frac{\partial \mathbf{d}_{\mathcal{M}}}{\partial x_k} \frac{\partial \mathbf{d}_{\mathcal{M}}}{\partial x_j} \right) \\
&\quad \times \left(\sum_{l=1}^m \mu_l(\varepsilon) \frac{\partial}{\partial x_i} \mathbf{d}_{\mathcal{N}}(\mathbf{u} + \varepsilon \Delta_{\mathcal{M}} \mathbf{u}) \frac{\partial \mathbf{d}_{\mathcal{M}}}{\partial x_l} \frac{\partial \mathbf{d}_{\mathcal{M}}}{\partial x_j} \right) \\
&\quad - 2 \frac{\partial u^i}{\partial x_j} \sum_{k=1}^m \left(\frac{\partial u^i}{\partial x_k} + \varepsilon \frac{\partial(\Delta_{\mathcal{M}} u^i)}{\partial x_k} - \varepsilon \mu_k(\varepsilon) \frac{\partial}{\partial x_i} \mathbf{d}_{\mathcal{N}}(\mathbf{u} + \varepsilon \Delta_{\mathcal{M}} \mathbf{u}) \right) \frac{\partial \mathbf{d}_{\mathcal{M}}}{\partial x_k} \frac{\partial \mathbf{d}_{\mathcal{M}}}{\partial x_j} \\
&\quad - 2\varepsilon \frac{\partial(\Delta_{\mathcal{M}} u^i)}{\partial x_j} \left(\sum_{k=1}^m \frac{\partial u^i}{\partial x_k} \frac{\partial \mathbf{d}_{\mathcal{M}}}{\partial x_k} \frac{\partial \mathbf{d}_{\mathcal{M}}}{\partial x_j} \right) \\
&\quad \left. + 2\varepsilon \mu_j(\varepsilon) \frac{\partial}{\partial x_i} \mathbf{d}_{\mathcal{N}}(\mathbf{u} + \varepsilon \Delta_{\mathcal{M}} \mathbf{u}) \left(\sum_{k=1}^m \frac{\partial u^i}{\partial x_k} \frac{\partial \mathbf{d}_{\mathcal{M}}}{\partial x_k} \frac{\partial \mathbf{d}_{\mathcal{M}}}{\partial x_j} \right) \right\}.
\end{aligned}$$

The expansion of $\|\mathbf{J}_{\mathbf{u}(\mathbf{x},t)}^{\mathbf{d}_{\mathcal{M}}}\|_{\mathcal{F}}^2$ is much simpler, viz.,

$$\begin{aligned}
\|\mathbf{J}_{\mathbf{u}(\mathbf{x},t)}^{\mathbf{d}_{\mathcal{M}}}\|_{\mathcal{F}}^2 &= \sum_{i=1}^m \sum_{j=1}^m |\{\mathbf{J}_{\mathbf{u}(\mathbf{x},t)}^{\mathbf{d}_{\mathcal{M}}}\}_{ij}|^2, \\
&= \sum_{i=1}^m \sum_{j=1}^m \left\{ \left(\frac{\partial u^i}{\partial x_j} \right)^2 + \left(\sum_{k=1}^m \frac{\partial u^i}{\partial x_k} \frac{\partial \mathbf{d}_{\mathcal{M}}}{\partial x_k} \frac{\partial \mathbf{d}_{\mathcal{M}}}{\partial x_j} \right)^2 - 2 \frac{\partial u^i}{\partial x_j} \sum_{k=1}^m \frac{\partial u^i}{\partial x_k} \frac{\partial \mathbf{d}_{\mathcal{M}}}{\partial x_k} \frac{\partial \mathbf{d}_{\mathcal{M}}}{\partial x_j} \right\}.
\end{aligned}$$

Subtracting these two results for $\|\mathbf{J}_{\mathbf{u}(\mathbf{x},t+\varepsilon)}^{\mathbf{d}_{\mathcal{M}}}\|_{\mathcal{F}}^2$ and $\|\mathbf{J}_{\mathbf{u}(\mathbf{x},t)}^{\mathbf{d}_{\mathcal{M}}}\|_{\mathcal{F}}^2$, dividing by 2ε and taking $\varepsilon \rightarrow 0$ gives

$$\begin{aligned}
\lim_{\varepsilon \rightarrow 0} \frac{1}{2\varepsilon} (\|\mathbf{J}_{\mathbf{u}(\mathbf{x},t+\varepsilon)}^{\mathbf{d}_{\mathcal{M}}}\|_{\mathcal{F}}^2 - \|\mathbf{J}_{\mathbf{u}(\mathbf{x},t)}^{\mathbf{d}_{\mathcal{M}}}\|_{\mathcal{F}}^2) &= \\
&\sum_{i=1}^m \sum_{j=1}^m \left\{ \frac{\partial u^i}{\partial x_j} \frac{\partial(\Delta_{\mathcal{M}} u^i)}{\partial x_j} - \mu_j(0) \frac{\partial u^i}{\partial x_j} \frac{\partial}{\partial x_i} \mathbf{d}_{\mathcal{N}}(\mathbf{u}) \right. \\
&\quad + \left(\sum_{k=1}^m \frac{\partial u^i}{\partial x_k} \frac{\partial \mathbf{d}_{\mathcal{M}}}{\partial x_k} \frac{\partial \mathbf{d}_{\mathcal{M}}}{\partial x_j} \right) \left(\sum_{l=1}^m \left[\frac{\partial(\Delta_{\mathcal{M}} u^i)}{\partial x_l} - \mu_l(0) \frac{\partial}{\partial x_i} \mathbf{d}_{\mathcal{N}}(\mathbf{u}) \right] \frac{\partial \mathbf{d}_{\mathcal{M}}}{\partial x_l} \frac{\partial \mathbf{d}_{\mathcal{M}}}{\partial x_j} \right) \\
&\quad - \frac{\partial u^i}{\partial x_j} \sum_{k=1}^m \left(\frac{\partial(\Delta_{\mathcal{M}} u^i)}{\partial x_k} - \mu_k(0) \frac{\partial}{\partial x_i} \mathbf{d}_{\mathcal{N}}(\mathbf{u}) \right) \frac{\partial \mathbf{d}_{\mathcal{M}}}{\partial x_k} \frac{\partial \mathbf{d}_{\mathcal{M}}}{\partial x_j} \\
&\quad \left. - \left(\frac{\partial(\Delta_{\mathcal{M}} u^i)}{\partial x_j} - \mu_j(0) \frac{\partial}{\partial x_i} \mathbf{d}_{\mathcal{N}}(\mathbf{u}) \right) \left(\sum_{k=1}^m \frac{\partial u^i}{\partial x_k} \frac{\partial \mathbf{d}_{\mathcal{M}}}{\partial x_k} \frac{\partial \mathbf{d}_{\mathcal{M}}}{\partial x_j} \right) \right\} \quad (3.30)
\end{aligned}$$

One recognizes that

$$\sum_{i=1}^m \frac{\partial u^i(\mathbf{x}, t)}{\partial x_j} \frac{\partial}{\partial x_i} d_{\mathcal{N}}(\mathbf{u}(\mathbf{x}, t)) = \frac{\partial \mathbf{u}(\mathbf{x}, t)}{\partial x_j} \cdot \nabla d_{\mathcal{N}}(\mathbf{u}(\mathbf{x}, t)) = 0,$$

which simplifies numerous terms in (3.30) yielding

$$\begin{aligned} \lim_{\varepsilon \rightarrow 0} \frac{1}{2\varepsilon} (\|\mathbf{J}_{\mathbf{u}(\mathbf{x}, t+\varepsilon)}^{\text{d}_{\mathcal{M}}}\|_{\mathcal{F}}^2 - \|\mathbf{J}_{\mathbf{u}(\mathbf{x}, t)}^{\text{d}_{\mathcal{M}}}\|_{\mathcal{F}}^2) = \\ \sum_{i=1}^m \sum_{j=1}^m \frac{\partial u^i}{\partial x_j} \frac{\partial(\Delta_{\mathcal{M}} u^i)}{\partial x_j} + \sum_{i=1}^m \sum_{j=1}^m \left(\sum_{k=1}^m \frac{\partial u^i}{\partial x_k} \frac{\partial d_{\mathcal{M}}}{\partial x_k} \frac{\partial d_{\mathcal{M}}}{\partial x_j} \right) \left(\sum_{l=1}^m \frac{\partial(\Delta_{\mathcal{M}} u^i)}{\partial x_l} \frac{\partial d_{\mathcal{M}}}{\partial x_l} \frac{\partial d_{\mathcal{M}}}{\partial x_j} \right) \\ - \sum_{i=1}^m \sum_{j=1}^m \frac{\partial u^i}{\partial x_j} \sum_{k=1}^m \frac{\partial(\Delta_{\mathcal{M}} u^i)}{\partial x_k} \frac{\partial d_{\mathcal{M}}}{\partial x_k} \frac{\partial d_{\mathcal{M}}}{\partial x_j} - \sum_{i=1}^m \sum_{j=1}^m \frac{\partial(\Delta_{\mathcal{M}} u^i)}{\partial x_j} \sum_{k=1}^m \frac{\partial u^i}{\partial x_k} \frac{\partial d_{\mathcal{M}}}{\partial x_k} \frac{\partial d_{\mathcal{M}}}{\partial x_j}. \end{aligned}$$

Swapping dummy indices j and k in the last term gives the second to last term so we have

$$\begin{aligned} \lim_{\varepsilon \rightarrow 0} \frac{1}{2\varepsilon} (\|\mathbf{J}_{\mathbf{u}(\mathbf{x}, t+\varepsilon)}^{\text{d}_{\mathcal{M}}}\|_{\mathcal{F}}^2 - \|\mathbf{J}_{\mathbf{u}(\mathbf{x}, t)}^{\text{d}_{\mathcal{M}}}\|_{\mathcal{F}}^2) = \\ \sum_{i=1}^m \sum_{j=1}^m \frac{\partial u^i}{\partial x_j} \frac{\partial(\Delta_{\mathcal{M}} u^i)}{\partial x_j} + \sum_{i=1}^m \sum_{j=1}^m \left(\sum_{k=1}^m \frac{\partial u^i}{\partial x_k} \frac{\partial d_{\mathcal{M}}}{\partial x_k} \frac{\partial d_{\mathcal{M}}}{\partial x_j} \right) \left(\sum_{l=1}^m \frac{\partial(\Delta_{\mathcal{M}} u^i)}{\partial x_l} \frac{\partial d_{\mathcal{M}}}{\partial x_l} \frac{\partial d_{\mathcal{M}}}{\partial x_j} \right) \quad (3.31) \\ - 2 \sum_{i=1}^m \sum_{j=1}^m \sum_{k=1}^m \frac{\partial u^i}{\partial x_j} \frac{\partial(\Delta_{\mathcal{M}} u^i)}{\partial x_k} \frac{\partial d_{\mathcal{M}}}{\partial x_j} \frac{\partial d_{\mathcal{M}}}{\partial x_k}. \end{aligned}$$

Notice now that

$$\begin{aligned} \sum_{i=1}^m \sum_{j=1}^m \left(\sum_{k=1}^m \frac{\partial u^i}{\partial x_k} \frac{\partial d_{\mathcal{M}}}{\partial x_k} \frac{\partial d_{\mathcal{M}}}{\partial x_j} \right) \left(\sum_{l=1}^m \frac{\partial}{\partial x_l} (\Delta_{\mathcal{M}} u^i) \frac{\partial d_{\mathcal{M}}}{\partial x_l} \frac{\partial d_{\mathcal{M}}}{\partial x_j} \right) \\ = \sum_{i=1}^m \left(\sum_{k=1}^m \frac{\partial u^i}{\partial x_k} \frac{\partial d_{\mathcal{M}}}{\partial x_k} \right) \nabla d_{\mathcal{M}} \cdot \left(\sum_{l=1}^m \frac{\partial(\Delta_{\mathcal{M}} u^i)}{\partial x_l} \frac{\partial d_{\mathcal{M}}}{\partial x_l} \right) \nabla d_{\mathcal{M}}, \\ = \sum_{i=1}^m \sum_{k=1}^m \sum_{l=1}^m \frac{\partial u^i}{\partial x_k} \frac{\partial(\Delta_{\mathcal{M}} u^i)}{\partial x_l} \frac{\partial d_{\mathcal{M}}}{\partial x_k} \frac{\partial d_{\mathcal{M}}}{\partial x_l} \|\nabla d_{\mathcal{M}}\|_2^2, \end{aligned}$$

which is the same as the last term in (3.31) because $d_{\mathcal{M}}$ is a signed distance function so $\|\nabla d_{\mathcal{M}}\|_2^2 = 1$.

After much simplification we have obtained

$$\begin{aligned} \lim_{\varepsilon \rightarrow 0} \frac{1}{2\varepsilon} (\|\mathbf{J}_{\mathbf{u}(\mathbf{x}, t+\varepsilon)}^{\text{d}_{\mathcal{M}}}\|_{\mathcal{F}}^2 - \|\mathbf{J}_{\mathbf{u}(\mathbf{x}, t)}^{\text{d}_{\mathcal{M}}}\|_{\mathcal{F}}^2) = \\ \sum_{i=1}^m \sum_{j=1}^m \left(\frac{\partial u^i}{\partial x_j} \frac{\partial(\Delta_{\mathcal{M}} u^i)}{\partial x_j} - \sum_{k=1}^m \frac{\partial u^i}{\partial x_k} \frac{\partial(\Delta_{\mathcal{M}} u^i)}{\partial x_j} \frac{\partial d_{\mathcal{M}}}{\partial x_k} \frac{\partial d_{\mathcal{M}}}{\partial x_j} \right), \quad (3.32) \\ = \sum_{i=1}^m \nabla(\Delta_{\mathcal{M}} u^i) \cdot \nabla_{\mathcal{M}} u^i. \end{aligned}$$

The dot product form is realized by working in the opposite direction. Rewriting $\nabla_{\mathcal{M}}u^i = \Pi_{T_{\mathbf{u}}\mathcal{N}}\nabla u^i$ one has

$$\begin{aligned}\nabla_{\mathcal{M}}u^i &= \nabla u^i - \nabla d_{\mathcal{M}}\nabla d_{\mathcal{M}}^T\nabla u^i, \\ \Rightarrow \{\nabla_{\mathcal{M}}u^i\}_j &= \frac{\partial u^i}{\partial x_j} - \sum_{k=1}^m \frac{\partial u^i}{\partial x_k} \frac{\partial d_{\mathcal{M}}}{\partial x_k} \frac{\partial d_{\mathcal{M}}}{\partial x_j}.\end{aligned}$$

Hence

$$\begin{aligned}\sum_{i=1}^m \nabla(\Delta_{\mathcal{M}}u^i) \cdot \nabla_{\mathcal{M}}u^i &= \sum_{i=1}^m \nabla(\Delta_{\mathcal{M}}u^i) \cdot \left(\nabla u^i - \nabla d_{\mathcal{M}}\nabla d_{\mathcal{M}}^T\nabla u^i\right) \\ &= \sum_{i=1}^m \sum_{j=1}^m \frac{\partial(\Delta_{\mathcal{M}}u^i)}{\partial x_j} \left(\frac{\partial u^i}{\partial x_j} - \sum_{k=1}^m \frac{\partial u^i}{\partial x_k} \frac{\partial d_{\mathcal{M}}}{\partial x_k} \frac{\partial d_{\mathcal{M}}}{\partial x_j}\right), \\ &= \sum_{i=1}^m \sum_{j=1}^m \left(\frac{\partial u^i}{\partial x_j} \frac{\partial(\Delta_{\mathcal{M}}u^i)}{\partial x_j} - \sum_{k=1}^m \frac{\partial u^i}{\partial x_k} \frac{\partial(\Delta_{\mathcal{M}}u^i)}{\partial x_j} \frac{\partial d_{\mathcal{M}}}{\partial x_k} \frac{\partial d_{\mathcal{M}}}{\partial x_j}\right),\end{aligned}$$

which matches (3.32).

A final identity is needed before we integrate and show that $dE/dt \leq 0$. From an application of the product rule, with scalar functions ψ and ϕ ,

$$\nabla \cdot (\phi \nabla_{\mathcal{M}}\psi) = \nabla\phi \cdot \nabla_{\mathcal{M}}\psi + \phi(\nabla \cdot \nabla_{\mathcal{M}}\psi).$$

Rearranging and recognizing that $\nabla \cdot \nabla_{\mathcal{M}}\psi = \nabla \cdot (\Pi_{T_{\mathcal{N}}}\nabla\psi)$ is an implicit representation of $\Delta_{\mathcal{M}}\psi$, we have

$$\nabla\phi \cdot \nabla_{\mathcal{M}}\psi = \nabla \cdot (\phi \nabla_{\mathcal{M}}\psi) - \phi(\Delta_{\mathcal{M}}\psi).$$

Substituting $\phi = \Delta_{\mathcal{M}}u^i$ and $\psi = u^i$ gives

$$\nabla(\Delta_{\mathcal{M}}u^i) \cdot \nabla_{\mathcal{M}}u^i = \nabla \cdot ((\Delta_{\mathcal{M}}u^i)\nabla_{\mathcal{M}}u^i) - (\Delta_{\mathcal{M}}u^i)^2. \quad (3.33)$$

Using (3.33) and integrating

$$\begin{aligned}\frac{d}{dt}E[\mathbf{u}(\mathbf{x}, t)] &= \lim_{\varepsilon \rightarrow 0} \frac{1}{2\varepsilon} \int_{\mathcal{M}} (\|\mathbf{J}_{\mathbf{u}(\mathbf{x}, t+\varepsilon)}^{\mathcal{M}}\|_{\mathcal{F}}^2 - \|\mathbf{J}_{\mathbf{u}(\mathbf{x}, t)}^{\mathcal{M}}\|_{\mathcal{F}}^2) dv_{\mathcal{M}}, \\ &= \int_{\mathcal{M}} \sum_{i=1}^n \left(\nabla(\Delta_{\mathcal{M}}u^i) \cdot \nabla_{\mathcal{M}}u^i\right) dv_{\mathcal{M}}, \\ &= \sum_{i=1}^n \int_{\mathcal{M}} \left(\nabla \cdot ((\Delta_{\mathcal{M}}u^i)\nabla_{\mathcal{M}}u^i) - (\Delta_{\mathcal{M}}u^i)^2\right) dv_{\mathcal{M}}, \\ &= \sum_{i=1}^n \int_{\partial\mathcal{M}} ((\Delta_{\mathcal{M}}u^i)\nabla_{\mathcal{M}}u^i) \cdot \mathbf{n} dS_{\mathcal{M}} - \sum_{i=1}^n \int_{\mathcal{M}} (\Delta_{\mathcal{M}}u^i)^2 dv_{\mathcal{M}}.\end{aligned}$$

The last step uses the divergence theorem [41] stated in Theorem 2 below, with \mathbf{n} denoting the outward normal to the surface \mathcal{M} and $dS_{\mathcal{M}}$ the area element on \mathcal{M} .

Theorem 2 *Let \mathcal{M} be a compact oriented m -dimensional manifold with boundary $\partial\mathcal{M}$. If \mathbf{G} is a differentiable vector field defined in a neighbourhood of \mathcal{M} then,*

$$\int_{\mathcal{M}} \nabla \cdot \mathbf{G} dv_{\mathcal{M}} = \int_{\partial\mathcal{M}} \mathbf{G} \cdot \mathbf{n} dS_{\mathcal{M}}.$$

The divergence theorem is a special case of Stokes' theorem [69].

Applying the Neumann boundary condition $\mathbf{J}_{\mathbf{u}}^{\text{d}\mathcal{M}} \mathbf{n}|_{\partial\mathcal{M}} = \mathbf{0}$ the boundary integral vanishes, since components of $\mathbf{J}_{\mathbf{u}}^{\text{d}\mathcal{M}} \mathbf{n}$ are $\nabla_{\mathcal{M}} u^i \cdot \mathbf{n}$, for $i = 1, 2, \dots, n$. We are therefore left with the desired result,

$$\begin{aligned} \frac{d}{dt} E[\mathbf{u}(\mathbf{x}, t)] &= - \sum_{i=1}^n \int_{\mathcal{M}} (\Delta_{\mathcal{M}} u^i)^2, \\ &= - \int_{\mathcal{M}} \|\Delta_{\mathcal{M}} \mathbf{u}\|_2^2 dv_{\mathcal{M}} \leq 0. \end{aligned}$$

3.3.2 Equivalence to projection onto target manifold \mathcal{N}

We now show that the solution computed by Algorithm 3.2 (or 3.1) is equivalent to the PDE (3.17) (or (3.15)) as $\varepsilon \rightarrow 0$. The equivalence is shown for the more general case, Algorithm 3.2, since Algorithm 3.1 is just the specific case when $\mathbf{F} = \Delta_{\mathcal{M}} \mathbf{u}$ and $\eta = 1$.

Similar to the start of Section 3.3.1, first Taylor expand $\tilde{\mathbf{u}}(\mathbf{x}, t + \varepsilon)$ and substitute $\eta \mathbf{F}(\mathbf{x}, \tilde{\mathbf{u}}, \mathbf{J}_{\tilde{\mathbf{u}}}^{\text{d}\mathcal{M}}, \dots)$ for $\partial \tilde{\mathbf{u}} / \partial t$ to obtain

$$\tilde{\mathbf{u}}(\mathbf{x}, t + \varepsilon) = \tilde{\mathbf{u}}(\mathbf{x}, t) + \varepsilon \eta \mathbf{F}(\mathbf{x}, \tilde{\mathbf{u}}(\mathbf{x}, t), \mathbf{J}_{\tilde{\mathbf{u}}(\mathbf{x}, t)}^{\text{d}\mathcal{M}}, \dots) + \mathcal{O}(\varepsilon^2).$$

Taylor expanding above assumes that $\mathbf{u}(\mathbf{x}, t)$ is a C^2 map. The solution at time $t + \varepsilon$ can therefore be expressed as

$$\mathbf{u}(\mathbf{x}, t + \varepsilon) = \text{cp}_{\mathcal{N}}(\tilde{\mathbf{u}}(\mathbf{x}, t) + \varepsilon \eta \mathbf{F}(\mathbf{x}, \tilde{\mathbf{u}}(\mathbf{x}, t), \mathbf{J}_{\tilde{\mathbf{u}}(\mathbf{x}, t)}^{\text{d}\mathcal{M}}, \dots) + \mathcal{O}(\varepsilon^2)).$$

Now since $\text{cp}_{\mathcal{N}} : \mathbb{R}^n \rightarrow \mathbb{R}^n$ the general form of Theorem 1 must be used for its Taylor expansion. Note that the simplified notation in (3.26) is used here. Taylor expanding $\text{cp}_{\mathcal{N}}$ we have

$$\begin{aligned} \mathbf{u}(\mathbf{x}, t + \varepsilon) &= \text{cp}_{\mathcal{N}}(\tilde{\mathbf{u}}(\mathbf{x}, t)) + \mathbf{J}_{\text{cp}_{\mathcal{N}}(\tilde{\mathbf{u}}(\mathbf{x}, t))} \left(\varepsilon \eta \mathbf{F}(\mathbf{x}, \tilde{\mathbf{u}}(\mathbf{x}, t), \mathbf{J}_{\tilde{\mathbf{u}}(\mathbf{x}, t)}^{\text{d}\mathcal{M}}, \dots) + \mathcal{O}(\varepsilon^2) \right) + \mathcal{O}(\varepsilon^2), \\ &= \mathbf{u}(\mathbf{x}, t) + \varepsilon \eta \mathbf{J}_{\text{cp}_{\mathcal{N}}(\mathbf{u}(\mathbf{x}, t))} \mathbf{F}(\mathbf{x}, \mathbf{u}(\mathbf{x}, t), \mathbf{J}_{\mathbf{u}(\mathbf{x}, t)}^{\text{d}\mathcal{M}}, \dots) + \mathcal{O}(\varepsilon^2), \end{aligned}$$

since $\text{cp}_{\mathcal{N}}(\mathbf{u}(\mathbf{x}, t)) = \mathbf{u}(\mathbf{x}, t)$ and $\tilde{\mathbf{u}}(\mathbf{x}, t) = \mathbf{u}(\mathbf{x}, t)$. The assumption made when Taylor expanding is that $\text{cp}_{\mathcal{N}}$ is a C^2 map.

It is shown by März et al. [52] that for any $\mathbf{y} \in \mathcal{S}$

$$\mathbf{J}_{\text{cp}_{\mathcal{S}}(\mathbf{y})} = \mathbf{I} - \nabla d_{\mathcal{S}}(\mathbf{y}) \nabla d_{\mathcal{S}}(\mathbf{y})^T = \Pi_{T_{\mathbf{y}}\mathcal{S}}. \quad (3.34)$$

Using (3.34) in our previous expansion yields

$$\mathbf{u}(\mathbf{x}, t + \varepsilon) = \mathbf{u}(\mathbf{x}, t) + \varepsilon \eta \Pi_{T_{\mathbf{u}(\mathbf{x}, t)}\mathcal{N}}(\mathbf{F}(\mathbf{x}, \mathbf{u}(\mathbf{x}, t), \mathbf{J}_{\mathbf{u}(\mathbf{x}, t)}^{\text{d}\mathcal{M}}, \dots)) + \mathcal{O}(\varepsilon^2).$$

Rearranging to obtain an approximation to the time derivative gives

$$\frac{\mathbf{u}(\mathbf{x}, t + \varepsilon) - \mathbf{u}(\mathbf{x}, t)}{\varepsilon} = \eta \Pi_{T_{\mathbf{u}(\mathbf{x}, t)}\mathcal{N}}(\mathbf{F}(\mathbf{x}, \mathbf{u}(\mathbf{x}, t), \mathbf{J}_{\mathbf{u}(\mathbf{x}, t)}^{\text{d}\mathcal{M}}, \dots)) + \mathcal{O}(\varepsilon). \quad (3.35)$$

Taking the limit as $\varepsilon \rightarrow 0$ completes the proof,

$$\frac{\partial \mathbf{u}}{\partial t} = \eta \Pi_{T_{\mathbf{u}}\mathcal{N}}(\mathbf{F}(\mathbf{x}, \mathbf{u}, \mathbf{J}_{\mathbf{u}}^{\text{d}\mathcal{M}}, \mathbf{H}_{\mathbf{u}}^{\text{d}\mathcal{M}}, \dots)).$$

The truncation error involved when splitting the evolution (3.17) in Algorithm 3.2 is first order. To see this define the truncation error as

$$\tau = \frac{\partial \hat{\mathbf{u}}}{\partial t} - \eta \Pi_{T_{\mathbf{u}(\mathbf{x}, t)}\mathcal{N}}(\mathbf{F}(\mathbf{x}, \mathbf{u}(\mathbf{x}, t), \mathbf{J}_{\mathbf{u}(\mathbf{x}, t)}^{\text{d}\mathcal{M}}, \dots)), \quad (3.36)$$

where $\partial \hat{\mathbf{u}}/\partial t$ is an approximation to $\partial \mathbf{u}/\partial t$. Here the approximation to $\partial \mathbf{u}/\partial t$ is given in (3.35) as

$$\partial \hat{\mathbf{u}}/\partial t = \eta \Pi_{T_{\mathbf{u}(\mathbf{x}, t)}\mathcal{N}}(\mathbf{F}(\mathbf{x}, \mathbf{u}(\mathbf{x}, t), \mathbf{J}_{\mathbf{u}(\mathbf{x}, t)}^{\text{d}\mathcal{M}}, \dots)) + \mathcal{O}(\varepsilon).$$

Substituting $\partial \hat{\mathbf{u}}/\partial t$ into (3.36) gives a first order truncation error $\tau = \mathcal{O}(\varepsilon)$. This is consistent with the numerical convergence studies presented in Chapter 4.

Chapter 4

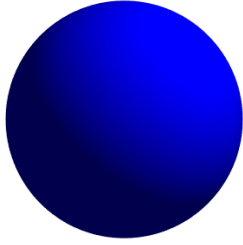
Numerical Results

There are numerous areas of application of harmonic maps and general manifold mappings. Certain applications, such as direct cortical mapping [68], are primarily interested in the values of the map \mathbf{u} . Other applications are more visual. In this section we highlight the behaviour and performance of our method with the computation of identity maps and two other visual applications. First, convergence studies of identity maps for the unit sphere, an ellipsoid and a torus are given. In Section 4.2 we denoise texture maps following the idea from Mémoli, Sapiro and Osher [54]. We also perform colour image denoising via chromaticity diffusion [74] in Section 4.3.

4.1 Identity maps

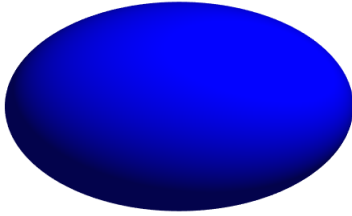
As mentioned in Chapter 3 the identity map $\mathbf{u}(\mathbf{x}) = \mathbf{x}$ is a harmonic map. Identity maps are obviously not the most interesting application since $\mathcal{M} = \mathcal{N}$; however, convergence studies can be conducted since the exact harmonic map is known. Here we perform convergence studies of identity maps for the unit sphere, an ellipsoid and a torus. Gaussian noise is applied in each direction of \mathbb{R}^3 to points on \mathcal{N} . The points are then projected back onto \mathcal{N} using $\text{cp}_{\mathcal{N}}$. In this manner an initial noisy map from \mathcal{M} to \mathcal{N} is created. Algorithm 3.1 is used with this initial noisy map as \mathbf{u}_0 to compute the harmonic map from \mathcal{M} to \mathcal{N} .

The addition of random noise produces slightly different convergence rates each time Algorithm 3.1 is used. Tables 4.1-4.3 therefore show convergence rates averaged over 10 trials of the algorithm. The forward Euler method was used to discretize in time and



Δx	Error	Conv. rate
0.1	1.89e - 02	
0.05	9.90e - 03	0.9285
0.025	5.07e - 03	0.9658
0.0125	2.49e - 03	1.0252
0.00625	1.24e - 03	1.0070

Table 4.1: Convergence study for the computation of a unit sphere identity map with Algorithm 3.1.



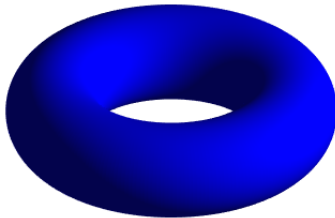
Δx	Error	Conv. rate
0.1	1.84e - 02	
0.05	9.77e - 03	0.9139
0.025	4.92e - 03	0.9910
0.0125	2.52e - 03	0.9674
0.00625	1.23e - 03	1.0268

Table 4.2: Convergence study for the computation of an ellipsoid identity map using Algorithm 3.1. The ellipsoid was constructed with minor axis of length 0.75 and major axis of length 1.25.

second order centred finite differences were used for the Laplacian operator. The time step size Δt was set as $\Delta t = 0.1\Delta x^2$ to ensure stability. The error was taken to be the average absolute error in the position of each point. That is, if the initial identity map was comprised of N points

$$\text{Error} = \frac{1}{N} \sum_{i=1}^N \|\mathbf{u}(\mathbf{x}_i) - \mathbf{x}_i\|_2.$$

All identity maps were computed using Algorithm 3.1 until time $t_f = 0.005$. Tables 4.1-4.3 confirm the first order convergence of Algorithm 3.1.



Δx	Error	Conv. rate
0.1	1.93e - 02	
0.05	9.90e - 03	0.9655
0.025	5.04e - 03	0.9744
0.0125	2.52e - 03	1.0019
0.00625	1.25e - 03	1.0101

Table 4.3: Convergence study of a torus identity map computation using Algorithm 3.1. The torus was constructed with minor radius 0.4 and major radius 1.

4.2 Diffusion of noisy texture maps

We now show the behaviour of Algorithm 3.1 by denoising a noisy texture map. The texture map gives a means for visualization of the map \mathbf{u} . To begin, a texture map \mathbf{T} is created using the ideas of Zigelman et al. [79]. The map \mathbf{T} is inverted to yield a map $\mathbf{w}(\mathbf{x}) : D \rightarrow \mathcal{N}$ from the planar image domain D to the surface \mathcal{N} .

A noisy map is created by adding a normally distributed random map $\mathbf{r}(\mathbf{x}) : D \rightarrow \mathbb{R}^3$ to \mathbf{w} . The sum of \mathbf{r} and \mathbf{w} is generally not on \mathcal{N} , so this summation is followed by a projection step onto \mathcal{N} . This gives a noisy map $\mathbf{u}_0 : D \rightarrow \mathcal{N}$ from the image domain to the surface \mathcal{N} defined by

$$\mathbf{u}_0(\mathbf{x}) = \text{cp}_{\mathcal{N}}(\mathbf{w}(\mathbf{x}) + \mathbf{r}(\mathbf{x})). \quad (4.1)$$

The construction of \mathbf{u}_0 here is the same as in Section 4.1 but with $\mathbf{w}(\mathbf{x})$ instead of $\mathbf{u}(\mathbf{x})$. The gradient descent equations (3.15) are expected to diffuse \mathbf{u}_0 analogous to H^1 -regularization of a planar image. It is important to recognize, however, that it is the initial map \mathbf{u}_0 that is diffused not the colour values of the image. The image is only placed on \mathcal{N} to enable visualization of the initial map and the harmonic map computed by Algorithm 3.1.

4.2.1 Texture mapping by multidimensional scaling

For completeness, an explanation of the texture mapping method by Zigelmann et al. [79] via multidimensional scaling (MDS) is needed. Note that there are many ways of obtaining \mathbf{T} . Computation of \mathbf{T} can rely on dimensionality reduction like Zigelman et al. (see [36] for a survey) or other mapping techniques such as conformal mappings [42]. Here a very simple method is chosen since the construction of \mathbf{T} is not the primary concern of our work; the map \mathbf{T} is only needed for visualization of our method. There are two key aspects to create \mathbf{T} : the computation of geodesic distances on \mathcal{N} and the dimensionality reduction of \mathcal{N} based on geodesic distances.

There are a number of possible ways to compute geodesic distance between points on the manifold \mathcal{N} . For the example of $\mathcal{N} = S^2$ in Section 4.2.2, there is an exact expression for geodesic distance between points \mathbf{u}_i and \mathbf{u}_j . The geodesic distance between points \mathbf{u}_i and \mathbf{u}_j on the unit sphere is simply

$$d_{ij} = \cos^{-1}(\mathbf{u}_i \cdot \mathbf{u}_j).$$

For more arbitrary objects there have been many different methods developed to compute geodesic distance between \mathbf{u}_i and \mathbf{u}_j . Some examples for polyhedral surfaces are [56, 79]. A method for use with a level set implicit representation of the surface is given in [53]. A recent approach applicable to many different surface representations, including point clouds, is introduced in [28]. For the example in Section 4.2.2 with \mathcal{N} as a triangulated pig surface we use the publicly available MATLAB software [1], which implements the work of Mitchell et al. [56] with some minor improvements.

Once geodesic distances can be computed classical MDS is used to construct a map \mathbf{T} from \mathcal{N} to the plane $D \subset \mathbb{R}^2$. Choose q points on \mathcal{N} and let the geodesic distance between points \mathbf{u}_i and \mathbf{u}_j be denoted d_{ij} . A $q \times q$ real symmetric matrix, \mathbf{M} , of squared geodesic distances is formed, where $\{\mathbf{M}\}_{ij} = d_{ij}^2$. MDS is then used to “flatten” the n -dimensional coordinates of \mathcal{N} to 2-dimensional coordinates based on \mathbf{M} . Hence, the goal is to compute a $q \times 2$ matrix \mathbf{X} that contains the 2-dimensional coordinates corresponding to each n -dimensional point on \mathcal{N} . Obviously for manifolds with effective Gaussian curvature, one cannot flatten \mathcal{N} such that the Euclidean distance between points in \mathbf{X} equals the geodesic distance between points on \mathcal{N} (see [31]). Instead MDS aims to compute coordinates in \mathbf{X} such that the error between the Euclidean distances and corresponding geodesic distances is small.

To compute the 2-dimensional coordinates in \mathbf{X} one first applies a double centering and normalization to \mathbf{M} . This is accomplished by multiplying both sides of \mathbf{M} by the matrix $\mathbf{C} = \mathbf{I} - \frac{1}{q}\mathbf{1}\mathbf{1}^T$, where $\mathbf{1}$ is a vector of q ones. Then the matrix

$$\mathbf{B} = -\frac{1}{2}\mathbf{C}\mathbf{M}\mathbf{C}$$

is approximated (in a least squares sense) by a matrix of rank 2. From the properties of eigenvalue decompositions, we know that \mathbf{B} can be approximated using the two largest eigenvalues of \mathbf{B} and their corresponding eigenvectors. Let Λ_+ denote the 2×2 diagonal matrix of largest positive eigenvalues. Let \mathbf{Q}_+ be the $q \times 2$ matrix of eigenvectors corresponding to eigenvalues in Λ_+ . The approximate 2-dimensional coordinates of the q points on \mathcal{N} are

$$\mathbf{X} = \mathbf{Q}_+\Lambda_+.$$

Solving the eigenvalue problem to compute \mathbf{X} is accomplished in MATLAB as shown in Algorithm 4.1. The matrix \mathbf{X} can be shown to be the minimizer of the Strain loss function

Algorithm 4.1 Classical Multidimensional Scaling

```
C = eye(size(M,1)) - ones(size(M,1))./(size(M,1));  
B = -0.5 * C * M * C; % center and normalize M  
[Q, Λ, ~] = eigs(B,2,'LA'); % solve for eigenvalues and eigenvectors  
X = [sqrt(Λ(1,1)).*Q(:,1), sqrt(Λ(2,2)).*Q(:,2)];
```

$$L(\mathbf{X}) = \|\mathbf{X}\mathbf{X}^T - \mathbf{B}\|,$$

which is one of many alternative loss functions in the theory of multidimensional scaling. For more information see [79] and references within.

The correspondence between the q selected points on \mathcal{N} and 2-dimensional points in \mathbf{X} defines the texture map \mathbf{T} from \mathcal{N} to D . The map is inverted to obtain the map $\mathbf{w} : D \rightarrow \mathcal{N}$. Since colour values of the image I may be defined at different locations \mathbf{x} than locations in D , an interpolation scheme must be used to determine the colour value at each point in \mathbf{X} . This interpolation was accomplished using the `griddata` command in MATLAB. After interpolation we have constructed the map $\mathbf{w}(\mathbf{x}) : I(\mathbf{x}) \rightarrow \mathcal{N}$, which allows placement of I onto \mathcal{N} . To visualize the image on \mathcal{N} the surface is triangulated and then displayed using `patch` in MATLAB.

Note that one must take q large enough such that when the surface is triangulated the triangles are not visible, i.e., high enough resolution of \mathcal{N} . One also needs to distribute the q points around \mathcal{N} somewhat uniformly; there is no benefit from increasing q if all the points cluster in one portion of \mathcal{N} .

Define the closest point set, $\Omega_{\text{cp}_{\mathcal{N}}}$, as the points on \mathcal{N} obtained from evaluating $\text{cp}_{\mathcal{N}}(\mathbf{u})$ for each $\mathbf{u} \in \Omega_c$. Remember that the computational domain Ω_c consists of a uniform grid in \mathbb{R}^3 . For $\mathcal{N} = S^2$ the closest point set, $\Omega_{\text{cp}_{\mathcal{N}}}$, clusters in portions of the sphere, which leads to unsatisfactory triangulations. Instead a points set Ω_{S^2} obtained from the MATLAB command `sphere` and barycentric Lagrange interpolation, with degree 3 polynomials, is used to obtain the values of the map at Ω_{S^2} from $\Omega_{\text{cp}_{\mathcal{N}}}$. This also means that the map \mathbf{w} only needs to be constructed for Ω_{S^2} , which is more efficient when solving the eigenvalue problem in practice. In Section 4.2.2 we used $q = 640,000$ for $\mathcal{N} = S^2$ to create \mathbf{w} . Solving the eigenvalue problem in MATLAB is memory intensive, requiring more than 200Gb of RAM on Compute Canada's WestGrid server.

For the pig target manifold example in Section 4.2.2 we used $q \approx 18,000$ at points defined by a refinement of the original triangulation. The refinement was accomplished using the

publicly available MATLAB software [2]. The refinement gave an adequate distribution of the points for creating the map \mathbf{w} . Linear interpolation was once again used in each dimension to obtain values of the map at $\Omega_{\mathcal{N}=\text{pig}}$ from $\Omega_{\text{cp}_{\mathcal{N}}}$.

4.2.2 Harmonic maps from the plane

The initial noisy map \mathbf{u}_0 from \mathcal{M} , the plane, to \mathcal{N} can now be constructed as described in equation (4.1). Algorithm 3.1 is then applied with \mathbf{u}_0 as an initial condition. Numerical implementation of Algorithm 3.1 is nearly identical to the closest point method for PDEs on a single manifold [61]. The sole difference is the need to evaluate the closest points of $\tilde{\mathbf{u}}(\mathbf{x}, t + \varepsilon)$ on \mathcal{N} via $\text{cp}_{\mathcal{N}}(\tilde{\mathbf{u}}(\mathbf{x}, t + \varepsilon))$. Note that the computational domain Ω_c was banded with respect to the target manifold \mathcal{N} .

Our codes are all straightforward modifications of existing MATLAB closest point method software [4]. In particular, note that if $\text{cp}_{\mathcal{N}}$ has an explicit formula, then we add the corresponding formula to the code for heat flow on \mathcal{M} . If instead $\text{cp}_{\mathcal{N}}$ is defined on a discrete set of points, we introduce an interpolation step since $\tilde{\mathbf{u}}$ will not necessarily lie on the grid defining $\text{cp}_{\mathcal{N}}$. This interpolation was accomplished using `scatteredInterpolant` in MATLAB with a linear interpolant.

In both examples below, the heat equation is discretized by second order centred differences in space and forward Euler time-stepping. The discretization step size in space is taken to be $\Delta x = 0.05$ for the plane to unit sphere maps and $\Delta x = 0.003125$ for the plane to pig maps discussed below. The time step size is set as $\Delta t = 0.1\Delta x^2$ to ensure numerical stability. Further note that the surfaces are only triangulated afterwards for visualization purposes.

In our first example, we compute from a plane to $\mathcal{N} = S^2$. Figure 4.1 shows the original map \mathbf{w} (first column), the noisy map $\mathbf{u}_0(\mathbf{x})$ (second column) and our computed harmonic map $\mathbf{u}(\mathbf{x})$ (third column). Two different colour images are mapped onto $\mathcal{N} = S^2$ to exhibit different behaviour of the algorithm. Two viewing angles are shown for each of the different planar images. The blue, white, black and yellow checkerboard image [3] shows that angles are preserved with the harmonic map. The image of two parrots [60] shows that the overall image does not become distorted.

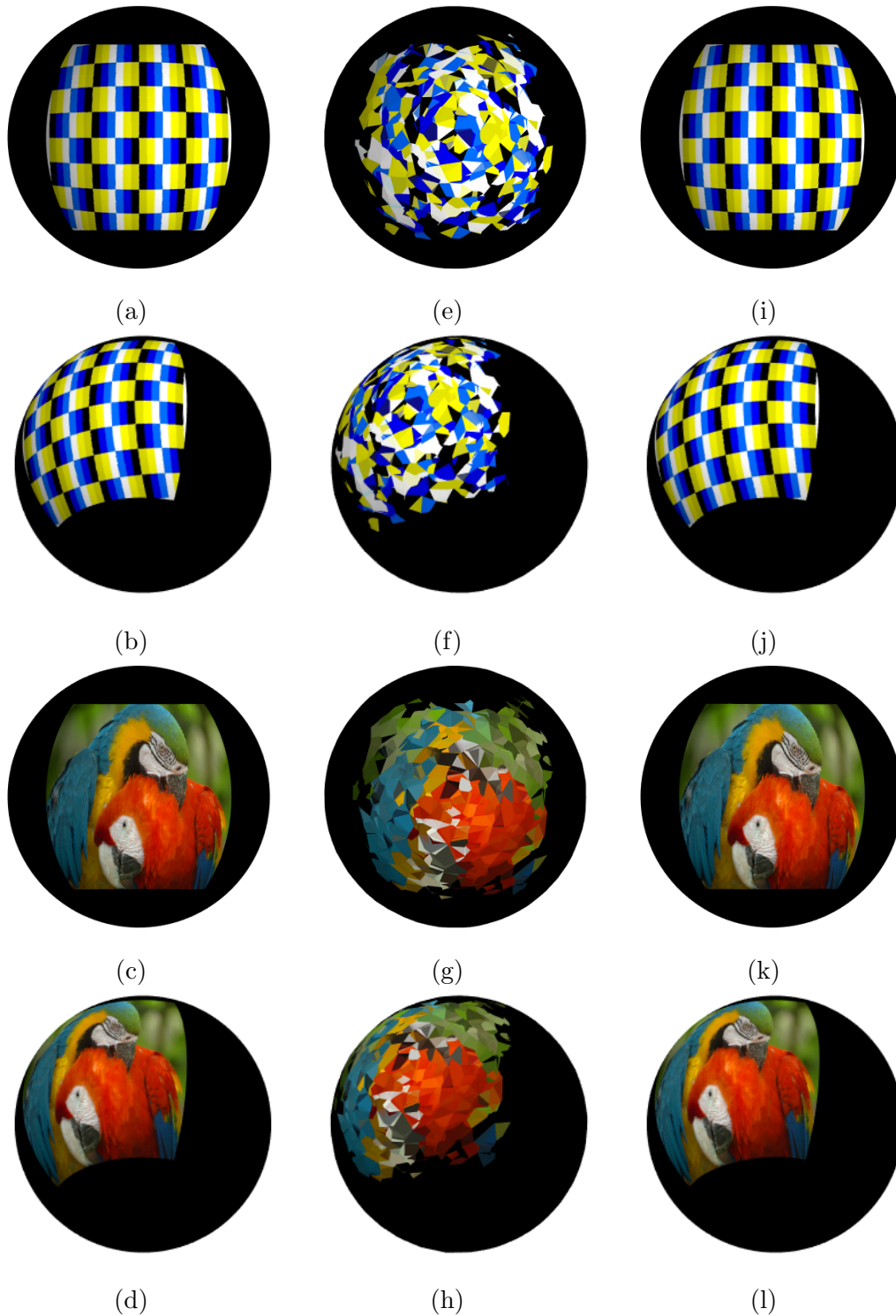


Figure 4.1: The original mapping \mathbf{w} of the image is given in (a)-(d) for two viewing angles with the two different planar images. The noisy initial map $\mathbf{u}_0(\mathbf{x})$ from a plane to the unit sphere is shown in (e)-(h). The harmonic map $\mathbf{u}(\mathbf{x})$ computed with Algorithm 3.1, after 60 time steps, is displayed on the unit sphere in (i)-(l).

Notice that the closest point function for the unit sphere is known analytically, $\text{cp}_{S^2} = \mathbf{x}/\|\mathbf{x}\|_2$, so no interpolation is needed. The harmonic map $\mathbf{u}(\mathbf{x})$ in Fig. 4.1 took approximately 1 second to compute on a Macbook Air (1.4 GHz Intel Core i5). Computation time does not include the preprocessing computation of \mathbf{u}_0 , which is different for every application.

In our second example, a harmonic map between a plane source manifold and a pig target manifold [46] is constructed. The first column of Figure 4.2 shows the initial map \mathbf{w} using two different planar images and at two viewing angles of the pig. The second and third columns of Figure 4.2 give two viewing angles of the initial noisy map \mathbf{u}_0 and computed harmonic map \mathbf{u} , respectively. The two planar images once again exhibit how harmonic maps preserve angles and the overall fidelity of the original image. The colourful horizontal checker planar image was found at [5].

The closest point function $\text{cp}_{\mathcal{N}}$ for the pig is evaluated on a grid near the pig surface using an algorithm described in [61]. The closest point function $\text{cp}_{\mathcal{N}}$ is then evaluated as needed using linear interpolation based on these grid values. This interpolation combined with a smaller Δx (to capture fine features on the pig) cause much more computational work. The harmonic map in Fig. 4.2 took roughly 2.4 hours to compute on a Macbook Air (1.4 GHz Intel Core i5). Computation time does not include the preprocessing steps of computing \mathbf{u}_0 and $\text{cp}_{\mathcal{N}}$.

We conclude this section with a convergence study of \mathcal{M} as a plane and $\mathcal{N} = S^2$. There is no analytical solution for this example so we compute an error estimate with a reference solution, \mathbf{u}_{ref} , using $\Delta x = 0.003125$. The difference between \mathbf{u}_{ref} and \mathbf{u} is compared with different Δx at the final time $t_f = 0.015$. The error estimate used is the average 2-norm of the difference between \mathbf{u}_{ref} and \mathbf{u} . Specifically, the error estimate is taken to be

$$(\text{Error Est.}) = \frac{1}{N} \sum_{i=1}^N \|\mathbf{u}(\mathbf{x}_i) - \mathbf{u}_{\text{ref}}(\mathbf{x}_i)\|_2,$$

where $N = 800^2$ is the number of points in the initial texture map. In Table 4.4 first order convergence is observed when averaging the errors and convergence rates over 100 trials.

4.2.3 Harmonic maps from a cylinder to a submanifold of S^2

In Section 4.1 examples of harmonic maps between two curved surfaces, with $\mathcal{M} = \mathcal{N}$, were shown. In the last section harmonic maps were computed between the plane and a

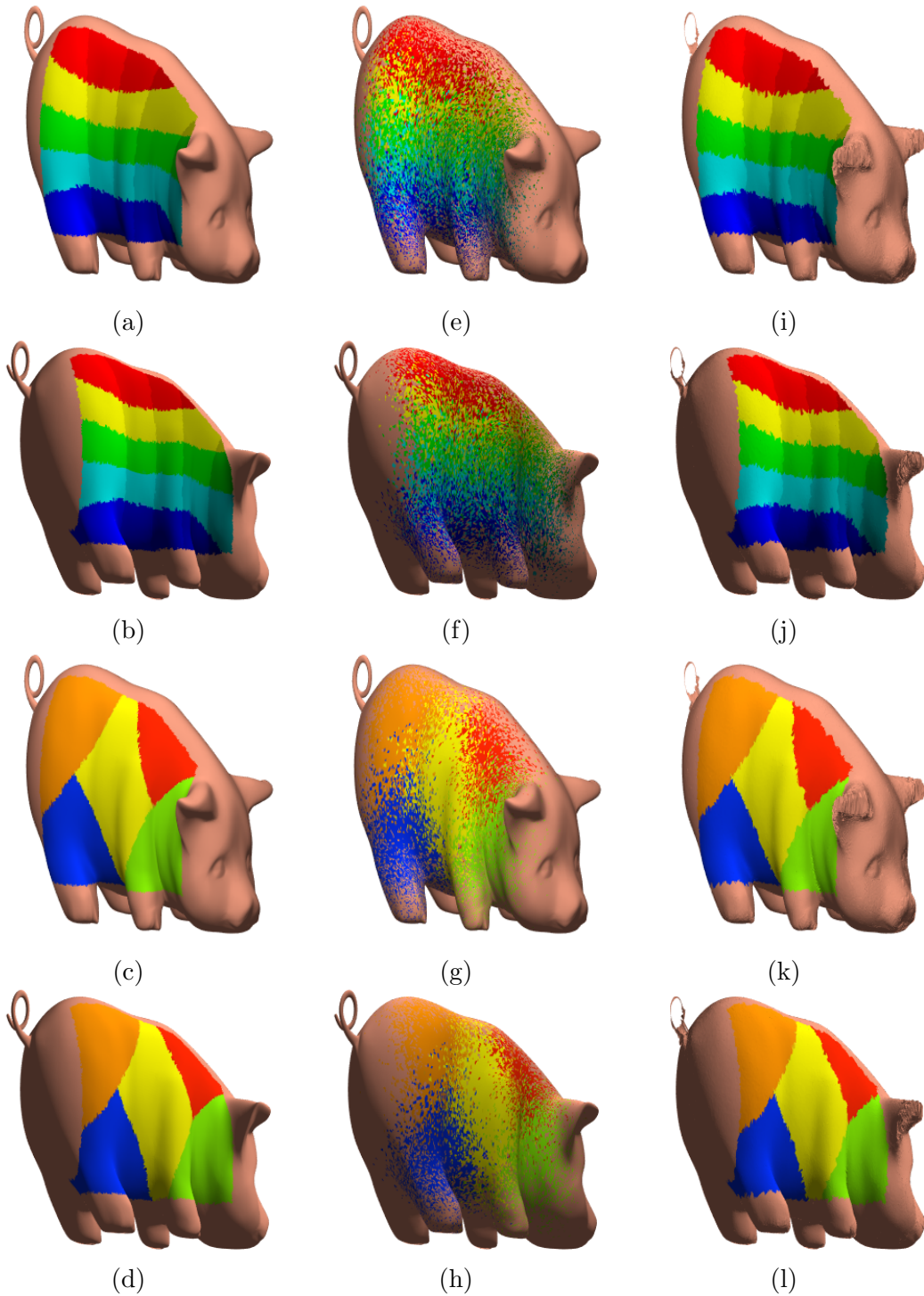


Figure 4.2: The original mapping \mathbf{w} of the image is given in (a)-(d) for two viewing angles with the two different planar images. The noisy initial map $\mathbf{u}_0(\mathbf{x})$ from a plane to a pig is shown in (e)-(h). The harmonic map $\mathbf{u}(\mathbf{x})$ computed using Algorithm 3.1 with 15 time steps is displayed on the pig in (i)-(l).

Δx	Error Est.	Conv. rate
0.1	6.59e - 03	
0.05	2.93e - 03	1.1684
0.025	1.43e - 03	1.0353
0.0125	7.32e - 04	0.9655
0.00625	3.71e - 04	0.9789

Table 4.4: Convergence study using errors between a reference solution \mathbf{u}_{ref} and harmonic map \mathbf{u} from a plane to S^2 .

curved surface \mathcal{N} . An example of harmonic maps between two curved surfaces with $\mathcal{M} \neq \mathcal{N}$ is given in this section. The harmonic maps are computed from a cylinder to a portion of the unit sphere. We take a unit radius cylinder with $z \in [-2, 2]$ and no top or bottom.

The construction of the initial map for this example is slightly different; the texture mapping method by Zigelman et al. [79] is not used. Instead an image is placed on the surface of the cylinder by a simple change of variables. The image is sized so that $x \in [-\pi, \pi]$ and $y \in [-2, 2]$. The x coordinate of the pixel is taken to represent the angle θ in cylindrical coordinates. The y coordinate of the pixel is taken as the height z in cylindrical coordinates. Intuitively, the planar image is simply rolled into a cylinder.

A texture map onto the sphere is then computed by finding points on the cylinder in the radial direction of points on the sphere. This is accomplished using the inverse of the closest point function for the sphere cp_{S^2} . For each point $\mathbf{p}_{S^2} = (p_1, p_2, p_3)^T$ of interest on the sphere, the point \mathbf{p}_{cyl} on the cylinder in the radial directions from \mathbf{p}_{S^2} is

$$\mathbf{p}_{\text{cyl}} = \begin{bmatrix} p_1 \sqrt{1 + p_3^2} \\ p_2 \sqrt{1 + p_3^2} \\ \frac{p_3}{\sqrt{1 - p_3^2}} \end{bmatrix}.$$

The point \mathbf{p}_{cyl} is generally not a pixel location on the cylinder, so linear interpolation is used to obtain the colour value at \mathbf{p}_{cyl} . The same colour is assigned to the corresponding point \mathbf{p}_{S^2} to place the image on the sphere. Note that since the cylinder is restricted to $z \in [-2, 2]$ the map only consists of points on a sphere with open north and south poles.

The addition of noise to the map between the cylinder and sphere is performed as before in (4.1). This noisy map from the cylinder to the sphere is taken as \mathbf{u}_0 . The noise in the map is removed by computing the harmonic map between the cylinder and sphere with Algorithm 3.1.

The computational domain is constructed as a band around the surface of the cylinder. Once again forward Euler time-stepping and second order centred finite differences in space are used. A spatial discretization of $\Delta x = 0.00625$ is taken for 300 time steps of $\Delta t = 0.1\Delta x^2$. The original, noisy and computed harmonic maps from the cylinder to a portion of the unit sphere are depicted in Figure 4.3. Two different planar images are placed on the sphere. For each planar image the sphere is viewed at two angles. The colour checker planar image source is [6].

4.3 Denoising colour images

Colour image denoising is another application that validates our numerical framework for harmonic maps and p -harmonic maps. The first approach one might use to denoise a colour image would be to denoise the intensity values, e.g., RGB colour intensity vector $I(\mathbf{x}) = (I_R(\mathbf{x}), I_G(\mathbf{x}), I_B(\mathbf{x}))^T$ with \mathbf{x} denoting the pixel location. Many authors have however applied a different approach to denoise colour images, since colour artifacts are frequently observed when denoising just intensity values. These artifacts are thought to occur when the direction (chromaticity) of I is not well preserved. The idea is therefore to denoise both the intensity I and the chromaticity $\mathbf{u} = I/\|I\|_2$, separately (see [74, 76] and numerous references within).

Originally the difficulty when denoising the chromaticity was because it is a map, $\mathbf{u}(\mathbf{x}) : \mathcal{M} \rightarrow S^2$, from a plane \mathcal{M} to the unit sphere S^2 . The chromaticity is however a map that can be easily denoised using our framework. Specifically, the algorithms for p -harmonic maps with $p = 2$ (isotropic diffusion) and $p = 1$ (anisotropic diffusion) are implemented. Methods for denoising the intensity of an image are well-established, e.g., H^1 -regularization, TV denoising [20], filtering approaches [9], but are not our focus here. Noise is only applied to the chromaticity $\mathbf{u} = I/\|I\|_2$ of the images here, so as to not confuse the denoising caused by chroma diffusion with intensity diffusion.

“Salt and pepper” like chromaticity noise is applied to the original image in the following manner. Choose some small subset of the image pixels, 5% in our examples, in a uniformly random manner. For each randomly selected pixel, choose red, green or blue in a uniformly random way and set $\mathbf{u} = (1, 0, 0)^T$ if red, $\mathbf{u} = (0, 1, 0)^T$ if green or $\mathbf{u} = (0, 0, 1)^T$ if blue. The

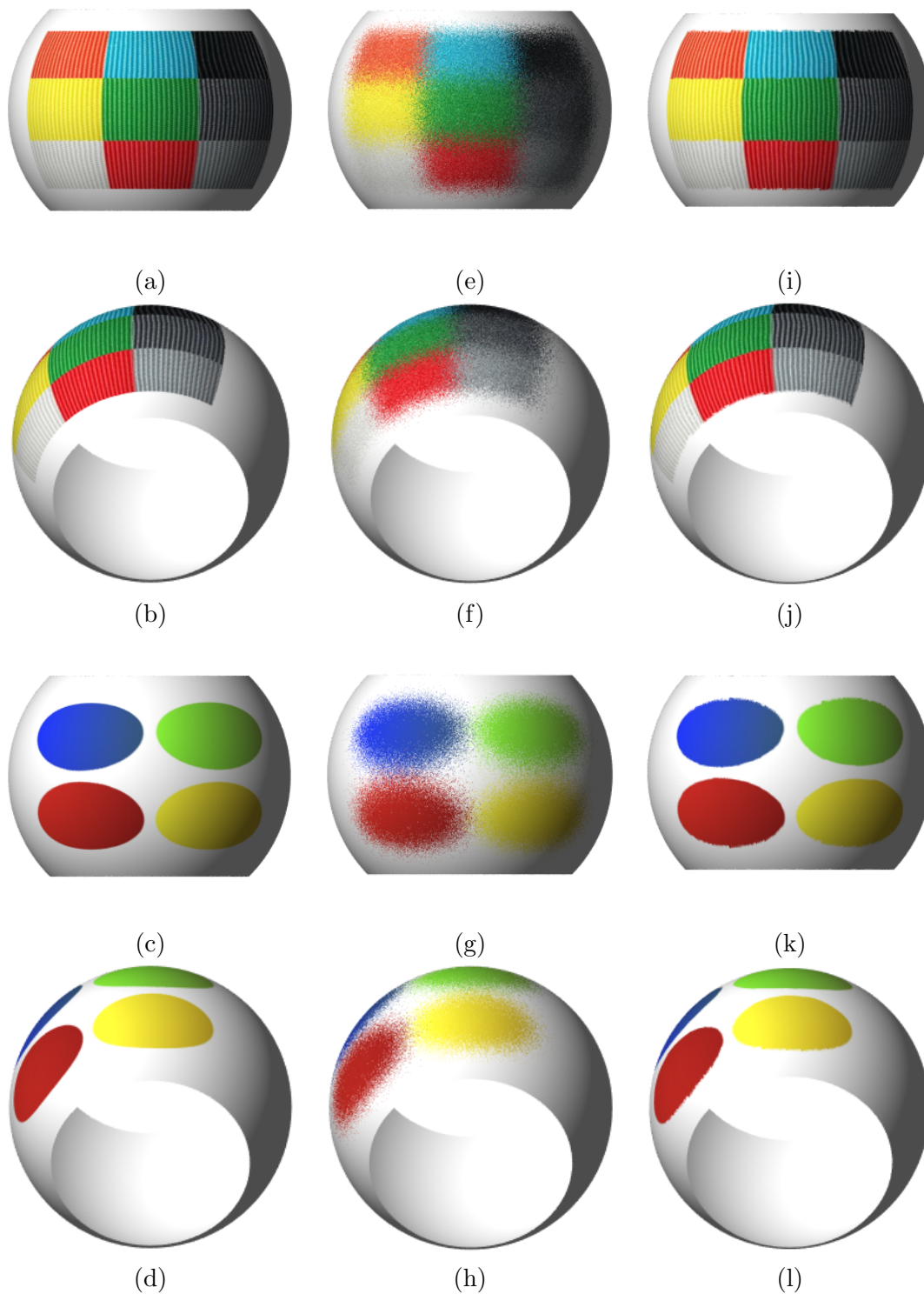


Figure 4.3: The original mapping (described in text) of the image is given in (a)-(d) with two planar images and viewed at two different angles for each. The corresponding noisy initial map $\mathbf{u}_0(\mathbf{x})$ from the cylinder to unit sphere is shown in (e)-(h). The harmonic map $\mathbf{u}(\mathbf{x})$ computed with Algorithm 3.1 after 300 time steps is displayed on the sphere in (i)-(l).



Figure 4.4: Isotropic diffusion of chromaticity noise for two 256×256 pixel images: cartoon of Newfoundland row houses [19] (first row) and natural scene of flowers (second row). The noisy image (b) was denoised to give (c) after 25 time steps. The noisy flower image in (e) took 30 time steps to give the denoised image in (f).

intensity values $I(\mathbf{x})$ of the original image remain unchanged. The latter noisy chromaticity map is taken as the initial map $\mathbf{u}_0(\mathbf{x})$.

To denoise the chromaticity with isotropic diffusion, Algorithm 3.1 is implemented. The diffusion is evolved for a number of time steps until the noise is considered sufficiently removed. Stopping criteria based on reaching steady state could also be implemented. To avoid interpolation of the initial map \mathbf{u}_0 , the uniform spatial discretization is taken at a resolution of one pixel, i.e., $\Delta x = 1$ pixel. The heat equation in Algorithm 3.1 is discretized using second order centred finite differences in space. A forward Euler discretization in time is taken with time step size $\Delta t = 0.1\Delta x^2$. Figure 4.4 shows the original images in the first column, chromaticity noise added in the second column and the isotropically denoised image in the third column.

Anisotropic chromaticity diffusion is slightly more involved numerically. The anisotropic diffusion of the initial noisy map is accomplished using (3.19) with $p = 1$, which simplifies



Figure 4.5: Anisotropic diffusion of 256×256 pixel images with chromaticity noise. A cartoon of Newfoundland row houses [19] (a) with noise added to the chromaticity in (b) was denoised to give (c) after 55 time steps. The flower image (d) with noise in (e) took 100 time steps to give the denoised image in (f).

to

$$\begin{cases} \frac{\partial \mathbf{u}}{\partial t} = \Pi_{T_{\mathbf{u}}\mathcal{N}} \left(\nabla \cdot \left(\frac{\mathbf{J}_{\mathbf{u}}}{\|\mathbf{J}_{\mathbf{u}}\|_{\mathcal{F}}} \right) \right), \\ \mathbf{u}(\mathbf{x}, 0) = \mathbf{u}_0(\mathbf{x}), \\ \mathbf{J}_{\mathbf{u}} \mathbf{n}|_{\partial \mathcal{M}} = 0. \end{cases} \quad (4.2)$$

Remember that the divergence of the matrix is defined to be the divergence of each row of the matrix. As mentioned in Chapter 3, the PDE (4.2) is in the class of PDEs that can be numerically solved by Algorithm 3.2. Each row of $\mathbf{J}_{\mathbf{u}}$ is computed using first order forward finite differences in space. The divergence of each row of $\mathbf{J}_{\mathbf{u}}/\|\mathbf{J}_{\mathbf{u}}\|_{\mathcal{F}}$ is then computed using first order backward finite differences. Forward Euler time-stepping is once again used but with time step size $\Delta t = 0.5\Delta x^2$. One obvious difficulty is if $\|\mathbf{J}_{\mathbf{u}}\|_{\mathcal{F}} = 0$, which causes a division by zero. In practice one divides by $\|\mathbf{J}_{\mathbf{u}}\|_{\mathcal{F}} + \delta$ where $\delta \in \mathbb{R}$ is some small positive constant. In the examples here δ is taken to be $\delta = 10^{-16}$.

Figure 4.5 shows the same original images as Figure 4.4 in the first column. The chro-

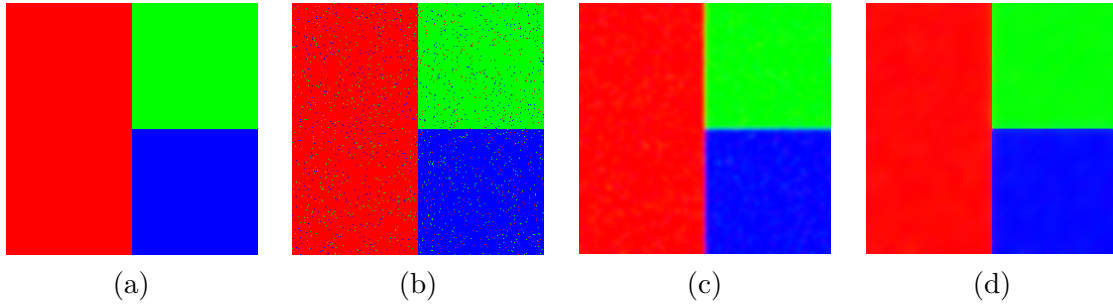


Figure 4.6: Comparison of isotropic and anisotropic diffusion of a simple 256×256 pixel red, green and blue image. Chromaticity noise is added in (b) and denoised with 30 time steps in (c) using isotropic diffusion. Anisotropic diffusion is applied for 125 time steps to give the denoised image (d). Notice the blur of colours in (c) and the preservation of the edges between colours in (d).

maticity noise is shown in the second column and the anisotropically denoised image is given in the third column. It is difficult to see the difference between the isotropic and anisotropic diffusion with the images in Figures 4.4 and 4.5. Figure 4.6 shows clearly how the anisotropic diffusion preserves edges between different colours better than isotropic diffusion.

Chapter 5

Conclusion

The main goal of this thesis was to establish a numerical framework for variational problems and PDEs that define maps between a source manifold \mathcal{M} and target manifold \mathcal{N} . A complete framework for all types of manifold mapping variational problems and PDEs is highly dependent on each type of problem. Some necessary ideas were however illustrated with the important example of p -harmonic maps. The inclusion of a data fidelity term with the harmonic mapping energy was also discussed.

To our knowledge, there has been no numerical framework developed for solving general variational problems and PDEs that define maps between manifolds. As mentioned at the start of Chapter 3, there has mostly been work on numerical methods for harmonic maps onto $\mathcal{N} = S^{n-1}$ using a finite element approach. The work progressed to a finite element method for harmonic maps onto more general target manifolds in [13]. Another approach, more closely related to the work here, is that of Mémoli et al. [54]. They embed the problem in the surrounding space using level set representations instead of the closest point representation. Their approach is detailed for the harmonic mapping problem; however, their ideas also extend to more general manifold mapping problems.

In our approach, the problem is embedded in the surrounding space using the closest point representations of \mathcal{M} and \mathcal{N} . This is accomplished by writing all geometric quantities intrinsic to \mathcal{M} and \mathcal{N} in terms of $\text{cp}_{\mathcal{M}}$ and $\text{cp}_{\mathcal{N}}$, respectively. By adopting the closest point formulation, geometric flexibility is gained (the surfaces can be open or closed and need not be oriented) as well as improved algorithm simplicity and efficiency. Moreover, the closest point approach does not have the same stability restrictions and convergence order degradation compared to the level set approach.

Algorithm 3.1 for harmonic maps between manifolds \mathcal{M} and \mathcal{N} is particularly straightforward; one alternates between a step of the closest point method for heat flow intrinsic to \mathcal{M} and a projection step onto \mathcal{N} (via the closest point mapping). Our novel splitting of the evolution into two steps reduces the problem of solving a PDE with quantities on both \mathcal{M} and \mathcal{N} to solving a PDE on just \mathcal{M} . Algorithm 3.1 was shown to decrease the harmonic energy as the time step $\varepsilon \rightarrow 0$. Algorithm 3.1 is also consistent with the original PDE up to first order. A more general PDE (3.17), involving intrinsic geometric quantities on \mathcal{M} that are projected onto the tangent plane of \mathcal{N} , can be handled similarly. Algorithm 3.2 details the two alternating steps to solve (3.17). The latter algorithm is shown to be first order accurate as well.

Chapter 4 gave numerical examples of identity maps, denoising noisy texture maps and denoising colour images to demonstrate the behaviour and performance of Algorithms 3.1 and 3.2. Algorithm 3.1 was confirmed numerically to be first order accurate for identity maps of the unit sphere, an ellipsoid and a torus. It was also shown to be first order accurate for a map from a plane to the unit sphere.

There are two obvious directions for future work. Exploring the use of the harmonic maps in other applications not presented here is one direction. Extending the framework to handle more general variational problems and PDEs is another possible avenue. Applying this framework to the texture mapping method of Dinh et al. [30] is a possible application. Another application is direct cortical mapping [68] discussed briefly in Section 3.2.1. The energy functional for direct cortical mapping involves the harmonic energy (3.5) and a data fidelity term. The difficulties for direct cortical mapping comes from the specialized numerical schemes needed to handle sulcal curves and the construction of an initial map \mathbf{u}_0 . A simpler medical imaging application is the computation of maps between optic nerve heads [39].

Somewhat less related to the work of this thesis, however important future work nonetheless, is the computation of geodesics. Geodesic computation arises in many problems in science and was needed for the texture mapping application of harmonic maps in Section 4.2. There are two promising approaches to geodesic computation with the closest point method. Similar to Mémoli et al. [53] one could solve the Euler-Lagrange equations (3.13) for harmonic maps from \mathbb{R} to the target manifold \mathcal{N} . The second approach would be to apply the ideas of Crane et al. [28] with the closest point method. Crane et al. [28] show that

their approach using heat flow is comparable in accuracy and much faster than the usual alternative of solving the eikonal equation.

Bibliography

- [1] Exact geodesic for triangular meshes. <http://www.mathworks.com/matlabcentral/fileexchange/18168-exact-geodesic-for-triangular-meshes>, March 2008. 37
- [2] Triangular mesh refinement. <http://www.mathworks.com/matlabcentral/fileexchange/16215-triangular-mesh-refinement>, February 2010. 39
- [3] Blue, yellow, black and white planar image. <http://imgbuddy.com/black-and-blue-checkerboard.asp>, March 2015. 39
- [4] Closest point method software. https://github.com/cbm755/cp_matrices, January 2015. 39
- [5] Colourful checkerboard planar image. <https://play.google.com/store/apps/details?id=idv.seventhmoon.colorboard>, March 2015. 41
- [6] Textured colour checkers planar image. http://www.strapworks.com/Tubular_Nylon_s/62.htm, March 2015. 44
- [7] François Alouges. A new algorithm for computing liquid crystal stable configurations: the harmonic mapping case. *SIAM Journal on Numerical Analysis*, 34(5):1708–1726, 1997. 15
- [8] François Alouges, Evaggelos Kritsikis, Jutta Steiner, and Jean-Christophe Toussaint. A convergent and precise finite element scheme for Landau–Lifschitz–Gilbert equation. *Numerische Mathematik*, pages 1–24, 2012. 15
- [9] Jaakko Astola, Petri Haavisto, and Yrjö Neuvo. Vector median filters. *Proceedings of the IEEE*, 78(4):678–689, 1990. 44
- [10] John W Barrett, Sören Bartels, Xiaobing Feng, and Andreas Prohl. A convergent and constraint-preserving finite element method for the p -harmonic flow into spheres. *SIAM Journal on Numerical Analysis*, 45(3):905–927, 2007. 15
- [11] Sören Bartels. Stability and convergence of finite-element approximation schemes for harmonic maps. *SIAM Journal on Numerical Analysis*, 43(1):220–238, 2005. 15
- [12] Sören Bartels. *Finite element approximation of harmonic maps between surfaces*. PhD thesis, Humboldt Universitt zu Berlin, 2009. 1
- [13] Sören Bartels. Numerical analysis of a finite element scheme for the approximation of harmonic maps into surfaces. *Mathematics of Computation*, 79(271):1263–1301, 2010. 15, 49

- [14] Leonid Berlyand and Evgenii Khruslov. Homogenization of harmonic maps and superconducting composites. *SIAM Journal on Applied Mathematics*, 59(5):1892–1916, 1999. 1
- [15] Jean-Paul Berrut and Lloyd N Trefethen. Barycentric lagrange interpolation. *SIAM Review*, 46(3):501–517, 2004. 7
- [16] Marcelo Bertalmío, Li-Tien Cheng, Stanley J Osher, and Guillermo Sapiro. Variational problems and partial differential equations on implicit surfaces. *Journal of Computational Physics*, 174(2):759–780, 2001. 2
- [17] Marcelo Bertalmío, Facundo Mévoli, Li-Tien Cheng, Guillermo Sapiro, and Stanley Osher. Variational problems and partial differential equations on implicit surfaces: Bye bye triangulated surfaces? In *Geometric Level Set Methods in Imaging, Vision, and Graphics*, pages 381–397. Springer, 2003. 2
- [18] Paul J Besl and Neil D McKay. Method for registration of 3-d shapes. In *Robotics-DL tentative*, pages 586–606. International Society for Optics and Photonics, 1992. 23
- [19] Doug Bird. *Fine fall day*. Holyrood Studios, 2015. xii, 46, 47
- [20] Peter Blomgren and Tony F Chan. Color TV: total variation methods for restoration of vector-valued images. *Image Processing, IEEE Transactions on*, 7(3):304–309, 1998. 44
- [21] Haïm Brezis, Jean-Michel Coron, and Elliott H Lieb. Harmonic maps with defects. *Communications in Mathematical Physics*, 107(4):649–705, 1986. 12
- [22] James A Carlson and Domingo Toledo. Harmonic mappings of Kähler manifolds to locally symmetric spaces. *Publications Mathématiques de l’IHÉS*, 69(1):173–201, 1989. 13
- [23] Yunmei Chen, Min-Chun Hong, and Norbert Hungerbühler. Heat flow of p -harmonic maps with values into spheres. *Mathematische Zeitschrift*, 215(1):25–35, 1994. 12, 22
- [24] Robert Cohen, Robert Hardt, David Kinderlehrer, San-Yin Lin, and Mitchell Luskin. *Minimum energy configurations for liquid crystals: Computational results*. Springer, 1987. 15
- [25] Robert Cohen, San-Yih Lin, and Mitchell Luskin. Relaxation and gradient methods for molecular orientation in liquid crystals. *Computer Physics Communications*, 53(1):455–465, 1989. 15
- [26] Rodney Coleman. *Calculus on Normed Vector Spaces*. Springer Science & Business Media, 2012. 26
- [27] Patrick Courilleau and Françoise Demengel. Heat flow for p -harmonic maps with values in the circle. *Nonlinear Analysis: Theory, Methods & Applications*, 41(5):689–700, 2000. 12
- [28] Keenan Crane, Clarisse Weischedel, and Max Wardetzky. Geodesics in heat: a new approach to computing distance based on heat flow. *ACM Transactions on Graphics (TOG)*, 32(5):152, 2013. 37, 50

- [29] E. Dean, R. Glowinski, and C.H. Li. Application of operator splitting methods to the numerical solution of nonlinear problems in continuum mechanics and physics. *Mathematics Applied to Science*, pages 13–64, 1988. 15
- [30] Huong Q Dinh, Anthony Yezzi, and Greg Turk. Texture transfer during shape transformation. *ACM Transactions on Graphics (TOG)*, 24(2):289–310, 2005. 1, 12, 50
- [31] Manfredo Perdigao Do Carmo. *Differential geometry of curves and surfaces*, volume 2. Prentice-Hall Englewood Cliffs, 1976. 37
- [32] James Eells and Luc Lemaire. A report on harmonic maps. *Bulletin of the London mathematical society*, 10(1):1–68, 1978. 12, 14
- [33] James Eells and Luc Lemaire. Another report on harmonic maps. *Bulletin of the London Mathematical Society*, 20(5):385–524, 1988. 12
- [34] James Eells and Joseph H Sampson. Harmonic mappings of Riemannian manifolds. *American Journal of Mathematics*, pages 109–160, 1964. 12
- [35] Michael S Floater and Kai Hormann. Surface parameterization: a tutorial and survey. In *Advances in multiresolution for geometric modelling*, pages 157–186. Springer, 2005. 1
- [36] Imola K Fodor. A survey of dimension reduction techniques, 2002. 36
- [37] Cătălin Gherghe. Harmonic maps and stability on locally conformal Kähler manifolds. *Journal of Geometry and Physics*, 70:48–53, 2013. 13
- [38] Mariano Giaquinta, Giuseppe Modica, and Jaroslav Souček. Variational problems for maps of bounded variation with values in S^1 . *Calculus of Variations and Partial Differential Equations*, 1(1):87–121, 1993. 12, 22
- [39] Eli Gibson, Mei Young, Marinko V Sarunic, and Mirza Faisal Beg. Optic nerve head registration via hemispherical surface and volume registration. *Biomedical Engineering, IEEE Transactions on*, 57(10):2592–2595, 2010. 23, 50
- [40] John B Greer. An improvement of a recent Eulerian method for solving PDEs on general geometries. *Journal of Scientific Computing*, 29(3):321–352, 2006. 2, 3
- [41] Helmut Grunsky. *The general Stokes’ theorem*, volume 9. Addison-Wesley Longman Ltd, 1983. 31
- [42] Xianfeng Gu, Yalin Wang, Tony F Chan, Paul M Thompson, and Shing-Tung Yau. Genus zero surface conformal mapping and its application to brain surface mapping. *Medical Imaging, IEEE Transactions on*, 23(8):949–958, 2004. 36
- [43] Robert Gulliver and J-M Coron. Minimizing p -harmonic maps into spheres. *Journal für die reine und angewandte Mathematik*, 401:82–100, 1989. 12
- [44] Ernst Hairer, Christian Lubich, and Gerhard Wanner. *Geometric numerical integration: structure-preserving algorithms for ordinary differential equations*, volume 31. Springer Science & Business Media, 2006. 25

- [45] Robert Hardt. Singularities of harmonic maps. *Bulletin of the American Mathematical Society*, 34(1):15–34, 1997. 12
- [46] Annie Hui. Pig in the AIM@SHAPE shape repository. <http://shapes.aimatshape.net>, 2008. 41
- [47] Jürgen Jost. *Riemannian geometry and geometric analysis*. Springer, 6 edition, 2011. 13, 14
- [48] Lev D Landau and Eugenii M Lifshitz. On the theory of the dispersion of magnetic permeability in ferromagnetic bodies. *Phys. Z. Sowjetunion*, 8(153):101–114, 1935. 1
- [49] Randall J LeVeque. *Finite difference methods for ordinary and partial differential equations: steady-state and time-dependent problems*, volume 98. SIAM, 2007. 2, 24
- [50] San-Yih Lin and Mitchell Luskin. Relaxation methods for liquid crystal problems. *SIAM Journal on Numerical Analysis*, 26(6):1310–1324, 1989. 15
- [51] Colin B Macdonald and Steven J Ruuth. The implicit closest point method for the numerical solution of partial differential equations on surfaces. *SIAM Journal on Scientific Computing*, 31(6):4330–4350, 2009. 5, 8
- [52] Thomas Marz and Colin B Macdonald. Calculus on surfaces with general closest point functions. *SIAM Journal on Numerical Analysis*, 50(6):3303–3328, 2012. 5, 6, 15, 20, 33
- [53] Facundo Mévoli and Guillermo Sapiro. Fast computation of weighted distance functions and geodesics on implicit hyper-surfaces. *Journal of computational Physics*, 173(2):730–764, 2001. 37, 50
- [54] Facundo Mévoli, Guillermo Sapiro, and Stanley J Osher. Solving variational problems and partial differential equations mapping into general target manifolds. *Journal of Computational Physics*, 195(1):263–292, 2004. 2, 3, 15, 16, 17, 19, 20, 21, 22, 23, 34, 49
- [55] Facundo Mévoli, Guillermo Sapiro, and Paul M Thompson. Implicit brain imaging. *NeuroImage*, 23:S179–S188, 2004. 1, 12, 16
- [56] Joseph SB Mitchell, David M Mount, and Christos H Papadimitriou. The discrete geodesic problem. *SIAM Journal on Computing*, 16(4):647–668, 1987. 37
- [57] Roger Moser. *Partial regularity for harmonic maps and related problems*. World Scientific, 2005. 5, 13, 16, 17, 19, 20
- [58] John Nash. C^1 isometric imbeddings. *Annals of Mathematics*, pages 383–396, 1954. 13
- [59] John Nash. The imbedding problem for Riemannian manifolds. *Annals of mathematics*, pages 20–63, 1956. 13
- [60] Guillermo Ossa. Macaws. <http://www.stockvault.net/photo/113075/macaws>, 2010. 39

- [61] Steven J Ruuth and Barry Merriman. A simple embedding method for solving partial differential equations on surfaces. *Journal of Computational Physics*, 227(3):1943–1961, 2008. 2, 5, 6, 9, 39, 41
- [62] Joseph H Sampson. Some properties and applications of harmonic mappings. *Ann. Sci. Ecole Norm. Sup*, 4(11):2, 1978. 12
- [63] Richard Schoen and Karen Uhlenbeck. A regularity theory for harmonic maps. *Journal of Differential Geometry*, 17(2):307–335, 1982. 12
- [64] Richard Schoen and Karen Uhlenbeck. Boundary regularity and the Dirichlet problem for harmonic maps. *J. diff. Geom*, 18(2):253–268, 1983. 12
- [65] Richard Schoen and Karen Uhlenbeck. Regularity of minimizing harmonic maps into the sphere. *Inventiones mathematicae*, 78(1):89–100, 1984. 12
- [66] Richard Schoen and Shing-Tung Yau. *Lectures on harmonic maps*. International Press Cambridge, 1997. 13, 14, 19
- [67] Yibing Shen and Yan Zhang. Second variation of harmonic maps between Finsler manifolds. *Science in China Series A: Mathematics*, 47(1):39–51, 2004. 13
- [68] Yonggang Shi, Paul M Thompson, Ivo Dinov, Stanley J Osher, and Arthur W Toga. Direct cortical mapping via solving partial differential equations on implicit surfaces. *Medical image analysis*, 11(3):207–223, 2007. 1, 23, 24, 34, 50
- [69] Michael Spivak. *Calculus on manifolds*, volume 1. WA Benjamin New York, 1965. 32
- [70] Michael Struwe. On the evolution of harmonic mappings of Riemannian surfaces. *Commentarii Mathematici Helvetici*, 60(1):558–581, 1985. 12
- [71] Michael Struwe. The evolution of harmonic maps: Existence, partial regularity, and singularities. In *Nonlinear Diffusion Equations and Their Equilibrium States, 3*, pages 485–491. Springer, 1992. 12
- [72] Michael Struwe. Uniqueness of harmonic maps with small energy. *manuscripta mathematica*, 96(4):463–486, 1998. 12
- [73] Bei Tang, Guillermo Sapiro, and Vicent Caselles. Diffusion of general data on non-flat manifolds via harmonic maps theory: The direction diffusion case. *International Journal of Computer Vision*, 36(2):149–161, 2000. 1
- [74] Bei Tang, Guillermo Sapiro, and Vicent Caselles. Color image enhancement via chromaticity diffusion. *Image Processing, IEEE Transactions on*, 10(5):701–707, 2001. 1, 12, 22, 34, 44
- [75] Arthur W Toga, Paul M Thompson, Michael S Mega, Katherine L Narr, and Rebecca E Blanton. Probabilistic approaches for atlasing normal and disease-specific brain variability. *Anatomy and embryology*, 204(4):267–282, 2001. 23
- [76] Luminita A Vese and Stanley J Osher. Numerical methods for p -harmonic flows and applications to image processing. *SIAM Journal on Numerical Analysis*, 40(6):2085–2104, 2002. 1, 15, 22, 44

- [77] Epifanio G Virga. *Variational theories for liquid crystals*, volume 8 of *Applied Mathematics and Mathematical Computation*. Chapman & Hall, London, 1994. 1, 18
- [78] Thomas J Willmore. *Riemannian geometry*, volume 33. Clarendon Press Oxford, 1993. 1
- [79] Gil Zigelman, Ron Kimmel, and Nahum Kiryati. Texture mapping using surface flattening via multidimensional scaling. *Visualization and Computer Graphics, IEEE Transactions on*, 8(2):198–207, 2002. 36, 37, 38, 43