# MOVING MESHES ON GENERAL SURFACES

by

Benjamin Crestel

M.A.Sc., Ecole Polytechnique de Montréal, 2007

B.A.Sc., Ecole Nationale des Ponts et Chaussées, 2005

THESIS SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

IN THE

DEPARTMENT OF MATHEMATICS

FACULTY OF SCIENCE

# APPROVAL

| | |
|---|---|
| **Name:** | Benjamin Crestel |
| **Degree:** | Master of Science |
| **Title of Thesis:** | Moving Meshes on General Surfaces |
| **Examining Committee:** | Dr. Razvan C. Fetecau (Chair)<br>Assistant Professor, Department of Mathematics<br>Simon Fraser University |
| | Dr. Robert D. Russell (Senior Supervisor)<br>Professor, Department of Mathematics<br>Simon Fraser University |
| | Dr. Steven J. Ruuth (co-Supervisor)<br>Professor, Department of Mathematics<br>Simon Fraser University |
| | Dr. Manfred R. Trummer (Internal Examiner)<br>Professor and Department Chair, Department of Mathematics<br>Simon Fraser University |
| **Date Approved:** | 16 August 2011 |

# Abstract

Many phenomena in the applied and natural sciences occur on surfaces. To solve accurately the corresponding partial differential equations (PDEs), it is often necessary to adapt the mesh, based upon the geometry of the surface, or based upon the behaviour of the PDE solution. Moving mesh methods are particularly efficient strategies in many situations. PDEs explicitly involving the mesh speed, called moving mesh PDEs (MMPDEs), offer a robust technique to adapt the mesh. In this work, we implement, with the C++ finite-element library deal.II, a mesh adaptation based on the variable diffusion method. We generalize the moving mesh problem to curved surfaces by deriving appropriate mathematical and finite-element formulations. Furthermore, a simple method using surface parameterization is developed and implemented with deal.II. The results, for both fixed and dynamically adapting meshes, demonstrate the effectiveness of our method.

*To my wife Marianne*

*and*

*our daughter Anémone.*

# Acknowledgments

I thank my supervisors, Prof. Bob Russell and Prof. Steve Ruuth, for their technical guidance and moral support throughout my master thesis, and for providing me with such an exciting research project.

I also thank the members of my committee, Prof. Manfred Trummer and Prof. Razvan Fetecau, for the time they spent on my thesis and for kindly coping with my erratic schedule.

For answering my numerous questions about MMPDEs and kindly sharing some of his MATLAB codes, I express my sincere gratitude to Prof. Weizhang Huang from the University of Kansas.

I would like to give credit to Prof. Weiming Cao from the University of Texas at San Antonio for suggesting the use of the recovery technique in the finite-element implementation of MMPDE. For his crucial help, I thank him very much.

I am very thankful to Prof. Nilima Nigam for helping me out with the theory of my finite-element formulation, for introducing me to deal.II and for offering a receptive ear in so many occasions.

I am grateful to Todd Keeler for sharing his knowledge in C++ when I needed and Kevin Mitchell for maintaining an almost permanent technical support for Linux and so many other powerful open-source applications.

This research was possible thanks to the financial support of Simon Fraser University Targeted Special Graduate Entrance Scholarship, two Simon Fraser University Graduate Fellowships and NSERC grants.

Finally, I owe an infinite debt of gratitude to my wife Marianne for constantly supporting my decision to go back to school, patiently accepting the demands of my new life style and eventually engaging with me in the tumultuous journey of parenthood.

# Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

The solutions to partial differential equations (PDEs) defined on surfaces are of central interest for many researchers in the applied and natural sciences. Applications emerge from fields as diverse as biology, material science, image processing and fluid dynamics, to name but a few ([RM08], [Mac08]). But while PDEs in $\mathbb{R}^n$ have been covered by a vast and thorough research, much less attention has been devoted to the numerical solution of PDEs on surfaces. Furthermore, almost no analytical solutions exist on general surfaces. The development of efficient numerical methods is therefore critical.

The most popular methods for the solution of PDEs on surfaces (parameterization, triangulation and embedding methods) all require the definition of a mesh. To solve accurately these PDEs, it is often necessary to adapt the mesh to the specifics of the problem. Adaptive techniques are traditionally sorted into three categories:

1. We can adapt the mesh by adding or removing grid points in certain parts of the domain; this is called h-adaptivity.

2. Another technique, when using the finite-element method, is to vary the degree of the polynomial approximations; this is called p-adaptivity. Sometimes, h- and p-adaptivity are used together, forming a hybrid category called hp-adaptivity.

3. Lastly, the technique we focus on in this work consists of moving the initial grid points in order to reposition them in an optimal way; one can adapt based upon the geometry of the surface, or based upon the behaviour of the PDE solution.

These moving mesh methods are particularly efficient strategies in many situations. A PDE explicitly involving the mesh speed, called a moving mesh PDE (MMPDE), offers a robust technique to compute the mesh transformation.

There has been very limited research carried out to apply moving mesh methods to the solution of PDEs on surfaces. Indeed, there has to our knowledge been no published research to date solving MMPDEs on surfaces. The theory of MMPDEs was developed over the last decade by Huang, Russell and collaborators ([HR10], [CHR99], [HR01]), and it focused on the adaptive solution to PDEs in $\mathbb{R}^n$. The computation typically requires two steps, one for the solution of the mesh and another for the solution of the physical PDE itself. Both steps can in theory be carried out simultaneously, but in practice, they are more often done successively. In order to investigate the adaptive solution to PDEs on surfaces, we concentrate in this thesis on the computation of moving meshes on surfaces which are graphs. The objective of this work is to study the feasibility of that idea and to propose methods to compute moving meshes on surfaces.

This thesis is organized into two parts. In Chapter 2, we introduce the mathematical background of MMPDEs and derive the equations and finite-element formulation for a specific example in the plane. In the conclusion of this chapter, we implement our method and solve that example with the C++ finite-element library deal.II. In the following chapter, we generalize the moving mesh problem to curved surfaces by deriving appropriate mathematical and finite-element formulations. Subsequently, we develop a simple method, using surface parameterization, that is implemented in deal.II. Lastly, Chapter 4 presents the conclusions of this work.

# Chapter 2

# Moving Mesh Partial Differential Equations

## 2.1 Theory of Moving Mesh Partial Differential Equation

Prior to solving an example of a Moving Mesh Partial Differential Equation (MMPDE), we present in this section the theory behind MMPDEs. The content of this section is taken from Huang and Russell [HR10], except where mentioned.

### 2.1.1 Basic concepts of MMPDEs – 1D case

The one dimensional case is used to introduce the main ideas of adaptive moving mesh methods.

#### Equidistribution principle

The concept of adaptive moving meshes in one dimension is directly based on the equidistribution principle. Working on a bounded interval $[a,b]$ and given a function $\rho(x) > 0$, an equidistributed mesh $\{x_i\}$: $x_1 (= a), x_2, \ldots, x_N (= b)$ satisfies

$$\int_{x_1}^{x_2} \rho(x)dx = \ldots = \int_{x_{N-1}}^{x_N} \rho(x)dx.$$

The function $\rho$ is called the monitor function; it controls the density of the mesh points. We generalize that concept with the definition of a continuous equidistribution principle.

Let's introduce the coordinate transformation $x(\xi)$:

$$x = x(\xi) : [0,1] \longrightarrow [a,b]. \tag{2.1}$$

The coordinate transformation (2.1) is called an equidistributing coordinate transformation for $\rho(x)$ if it satisfies

$$\int_a^{x(\xi)} \rho(x)dx = \sigma\xi, \quad 0 \leq \xi \leq 1. \tag{2.2}$$

We can differentiate (2.2) with respect to $\xi$ to obtain the differential equation

$$\rho(x)\frac{dx}{d\xi} = \sigma. \tag{2.3}$$

Alternatively, we can derive a differential equation in terms of the inverse coordinate transformation $\xi(x)$, an approach which is sometimes preferred for its robustness:

$$\frac{1}{\rho(x)}\frac{d\xi}{dx} = \frac{1}{\sigma}.$$

**Computation of an equidistributed mesh**

The two main techniques to compute an equidistributed mesh are de Boor's algorithm and a boundary value problem (BVP) method.

De Boor's algorithm is an iterative procedure that uses piecewise constant approximation of the monitor function to compute the equidistributed mesh. Assuming that we have an existing mesh (the initial mesh or a mesh coming from a previous approximation) $\{y_i\}$: $y_1(=a), y_2, \ldots, y_N(=b)$, we build the piecewise constant approximation $p$ of the monitor function $\rho$:

$$p(x) = \begin{cases} \frac{1}{2}\left(\rho(y_1) + \rho(y_2)\right), & \text{for } x \in [y_1, y_2], \\ \frac{1}{2}\left(\rho(y_2) + \rho(y_3)\right), & \text{for } x \in (y_2, y_3], \\ \vdots & \\ \frac{1}{2}\left(\rho(y_{N-1}) + \rho(y_N)\right), & \text{for } x \in (y_{N-1}, y_N]. \end{cases}$$

With analogy to equation (2.2), we introduce $P(x)$:

$$P(x) = \int_a^x p(t)dt. \tag{2.4}$$

Since the function $p$ is piecewise constant, $P$ at a mesh point $y_j$ takes a simple form:

$$P(y_j) = \sum_{i=2}^{j}(y_i - y_{i-1})p(y_i) = \sum_{i=2}^{j}(y_i - y_{i-1})\frac{\rho(y_i)+\rho(y_{i-1})}{2}. \tag{2.5}$$

An equidistributed mesh $\{x_j\}$ must satisfy equation (2.2). This allows us to evaluate $P$ at a mesh point $x_j$:

$$P(x_j) = \xi_j P(y_N) = \xi_j P(b), \quad \forall j = 1,\dots,N-1,$$

with

$$\xi_j = \frac{j-1}{N-1}. \tag{2.6}$$

Because the monitor function is strictly positive, we see by equation (2.5) that the function $P$ is strictly increasing on $[a,b]$. Therefore, we can bound the unknown mesh point $x_j$ by bounding its image $P(x_j)$. Let $y_{k-1}$ and $y_k$ be the consecutive mesh points for which the images $P(y_{k-1})$ and $P(y_k)$ surround $P(x_j)$:

$$P(y_{k-1}) \le \xi_j P(b) = P(x_j) < P(y_k). \tag{2.7}$$

Equation (2.7) implies directly that $y_{k-1} \le x_j < y_k$. From the definition of $P$ (2.4), we can now evaluate the difference $\xi_j P(b) - P(y_{k-1})$:

$$\begin{aligned} & P(x_j) - P(y_{k-1}) = \int_{y_{k-1}}^{x_j} p(t)dt, \\ \Leftrightarrow \quad & \xi_j P(b) - P(y_{k-1}) = (x_j - y_{k-1})\frac{\rho(y_k)+\rho(y_{k-1})}{2}. \end{aligned} \tag{2.8}$$

From equation (2.8), we solve for the mesh point $x_j$:

$$x_j = y_{k-1} + 2\frac{\xi_j P(b) - P(y_{k-1})}{\rho(y_k)+\rho(y_{k-1})}.$$

As mentioned earlier, it is necessary to iterate several times to obtain a good approximation. The equidistribution quality measure $Q_{\text{eq}}$ provides a convenient stopping criterion. It is defined by

$$Q_{\text{eq}}(x) = \frac{\rho}{\sigma}\frac{dx}{d\xi}.$$

The maximum of the equidistribution quality measure satisfies

$$\max_x Q_{\text{eq}}(x) \geq 1,$$

with equality if and only if the mesh is equidistributed. It has the discrete analog

$$Q_{\text{eq},j} = \frac{\rho(x_j)+\rho(x_{j-1})}{2\sigma_h}\cdot\frac{x_j - x_{j-1}}{\xi_j - \xi_{j-1}}, \quad \forall j = 1,\dots,N-1,$$

with

$$\sigma_h = \sum_{j=1}^{N-1}\frac{\rho(x_j)+\rho(x_{j-1})}{2}\cdot(x_j - x_{j-1})$$

and $\xi_j$ as defined in equation (2.6).

Unfortunately, de Boor's algorithm does not extend well to higher dimensions. The BVP method is based on a boundary value problem formulation of equidistribution. Differentiating equation (2.3) with respect to $\xi$ gives a boundary value problem (2.9) formulated in terms of the unknown coordinate transformation $x(\xi)$ defined in equation (2.1), viz.,

$$\begin{cases} \dfrac{d}{d\xi}\left(\rho(x)\dfrac{dx}{d\xi}\right) = 0, \\ x(0) = a,\ x(1) = b. \end{cases} \tag{2.9}$$

**Theorem 2.1.1** (Euler-Lagrange equation [GF64]). *If $y(x)$ is a curve in $C[a,b]$ which minimizes the functional*

$$F[y(x)] = \int_a^b f\left(x, y(x), y'(x)\right)dx,$$

*then the following differential equation must be satisfied:*

$$\frac{\partial f}{\partial y} - \frac{d}{dx}\left(\frac{\partial f}{\partial y'}\right) = 0.$$

*This equation is called the Euler-Lagrange Equation.*

Equation (2.9) is the Euler-Lagrange equation for the functional

$$I[x] = \frac{1}{2}\int_0^1 \left(\rho(x)\frac{dx}{d\xi}\right)^2 d\xi. \tag{2.10}$$

Finding the coordinate transformation $x = x(\xi)$ is equivalent to finding a minimizer $x = x(\xi)$ for the functional $I[x]$. It is also possible to formulate the problem in terms of the inverse coordinate transformation,

$$\begin{cases} \dfrac{d}{dx}\left(\dfrac{1}{\rho(x)}\dfrac{d\xi}{dx}\right) = 0, \\ \xi(a) = 0,\ \xi(b) = 1. \end{cases} \tag{2.11}$$

Equation (2.11) is the Euler-Lagrange equation corresponding to the functional

$$I[\xi] = \frac{1}{2}\int_a^b \frac{1}{\rho(x)}\left(\frac{d\xi}{dx}\right)^2 dx. \tag{2.12}$$

The formulation in terms of the inverse coordinate transformation (2.11) is often preferred as the functional is then quadratic and its Euler-Lagrange equation is linear. In addition, well-posedness of the solution is easier to ensure.

**Moving Mesh Partial Differential Equations**

We now consider time-dependent problems, for which the monitor function and the coordinate transformation vary in time. The two main approaches to solve this problem are the following:

1. Solve the time-dependent equivalent of formulation (2.9),

$$\begin{cases} \dfrac{\partial}{\partial\xi}\left(\rho(x,t)\dfrac{\partial x}{\partial\xi}\right) = 0, \\ x(0,t) = a, \quad x(1,t) = b. \end{cases}$$

2. Solve an MMPDE, that is, a time-dependent PDE involving explicitly the mesh speed. This approach has two clear advantages over the first method.

   - This technique leads to a system of ODEs, which is "often easier to integrate than a system of differential-algebraic equations" (which arises in the first case) [HR10].

- The presence of the mesh speed in the PDE "provides a degree of temporal smoothing for mesh movement, which is necessary for accurate integration of many physical PDEs" [HR10].

Among the different techniques used to derive an MMPDE, the gradient flow equation (definition 2.1.1) has the advantage that it can be easily extended to higher dimensions. We saw that the equidistributed mesh will be a minimizer for some functional (e.g, functional (2.10) or (2.12)). If we look for the inverse coordinate transformation $\xi = \xi(x,t)$, the time-dependent functional is given by

$$I[\xi] = \frac{1}{2}\int_a^b \frac{1}{\rho(x,t)}\left(\frac{\partial \xi}{\partial x}\right)^2 dx. \tag{2.13}$$

**Definition 2.1.1** (Gradient Flow [Cow05])**.** A gradient flow can be defined as the following abstract ODE over a Hilbert space $H$:

$$\begin{cases} u'(t) = -K\, grad\left(F(u(t))\right), & t > 0, \\ u(0) = u_0 \in H, \end{cases}$$

where $F : H \to \mathbb{R}$ is some functional and $K > 0$.

The gradient flow lets the quantity $u$ evolve in a direction opposite to the gradient of some functional $F$, often representing an energy, which corresponds to letting the quantity $u$ evolve in a way that decreases the functional $F$ [Cow05]. The gradient flow equation for the functional (2.13) is given by

$$\frac{\partial \xi}{\partial t} = -\frac{P}{\tau}\frac{\delta I}{\delta \xi}, \tag{2.14}$$

where $\frac{\delta I}{\delta \xi}$ is the functional derivative of $I[\xi]$ (definition 2.1.2), $\tau > 0$ is a user-specified parameter for adjusting the response time of the mesh movement to changes in $\rho(x,t)$, and $P$ is a positive-definite differential operator [HR10].

**Definition 2.1.2** (Functional (or variational) derivative [GF64])**.** Let's consider the functional

$$J[y] = \int_a^b F(x,y,y')dx, \quad y(a) = A,\ y(b) = B.$$

Then, the functional (or variational) derivative is defined by

$$\frac{\delta J}{\delta y} = F_y(x,y,y') - \frac{d}{dx}F_{y'}(x,y,y').$$

Following definition 2.1.2, the functional derivative of $I[\xi]$ is given by

$$\frac{\delta I}{\delta \xi} = -\frac{\partial}{\partial x}\left(\frac{1}{\rho}\frac{\partial \xi}{\partial x}\right).$$

Using this result in equation (2.14) leads to the gradient flow equation

$$\frac{\partial \xi}{\partial t} = \frac{P}{\tau}\frac{\partial}{\partial x}\left(\frac{1}{\rho}\frac{\partial \xi}{\partial x}\right). \tag{2.15}$$

We solve here for the inverse coordinate transformation $\xi = \xi(x,t)$. It is a more desirable approach, as this technique is less likely to be singular than when solving for the direct coordinate transformation (a property which holds in higher dimensions as well). However, the quantity of interest remains the direct coordinate transformation $x = x(\xi,t)$. Therefore, we need to interchange the coordinates in equation (2.15). We use the relations

$$\begin{aligned} (t \text{ fixed})\ \frac{d\xi}{d\xi} = 1 = \frac{\partial \xi}{\partial x}\frac{\partial x}{\partial \xi} &\Rightarrow \frac{\partial x}{\partial \xi} = \left(\frac{\partial \xi}{\partial x}\right)^{-1}, \\ (\xi \text{ fixed})\ \frac{d\xi}{dt} = 0 = \frac{\partial \xi}{\partial t} + \frac{\partial \xi}{\partial x}\frac{\partial x}{\partial t} \Rightarrow \frac{\partial x}{\partial t} = -\left(\frac{\partial \xi}{\partial x}\right)^{-1}\frac{\partial \xi}{\partial t} &\Rightarrow \frac{\partial x}{\partial t} = -\frac{\partial x}{\partial \xi}\frac{\partial \xi}{\partial t}, \end{aligned} \tag{2.16}$$

which come from the differentiation of the identity $\xi = \xi(x(\xi,t),t)$, while holding $t$ or $\xi$ fixed. Replacing $\partial\xi/\partial t$ and $\partial\xi/\partial x$ in equation (2.15) with their expressions (2.16), we obtain the gradient flow equation (2.17) for the direct coordinate transformation, albeit based on the functional for the inverse coordinate transformation:

$$\begin{aligned} & \frac{\partial x}{\partial t} = -\frac{\partial x}{\partial \xi}\frac{P}{\tau}\frac{\partial}{\partial x}\left(\frac{1}{\rho}\frac{\partial \xi}{\partial x}\right), \\ \Leftrightarrow\quad & \frac{\partial x}{\partial t} = -\frac{P}{\tau}\frac{\partial}{\partial \xi}\left(\rho\frac{\partial x}{\partial \xi}\right)^{-1}, \\ \Leftrightarrow\quad & \frac{\partial x}{\partial t} = \frac{P}{\tau}\left(\rho\frac{\partial x}{\partial \xi}\right)^{-2}\frac{\partial}{\partial \xi}\left(\rho\frac{\partial x}{\partial \xi}\right). \end{aligned} \tag{2.17}$$

Different choices of $P$ lead to different MMPDEs. We only present two examples:

- $P = \left(\rho x_\xi\right)^2$ gives MMPDE5

$$\frac{\partial x}{\partial t} = \frac{1}{\tau}\frac{\partial}{\partial \xi}\left(\rho\frac{\partial x}{\partial \xi}\right).$$

- $P = \left(\rho x_\xi\right)^2/\rho$ gives the modified MMPDE5

$$\frac{\partial x}{\partial t} = \frac{1}{\tau\rho}\frac{\partial}{\partial \xi}\left(\rho\frac{\partial x}{\partial \xi}\right).$$

### 2.1.2 Generalization to higher dimensions

Some of the concepts introduced in Section 2.1.1 which intuitively generalize to higher dimensions are the following:

- The monitor function $M(\vec{x})$ is now a matrix, required to be symmetric and positive-definite. It contains the information on the mesh transformation.
- The mesh functional defines the properties of the adaptivity.
- The MMPDE is derived from a gradient flow equation.

We present below the main modifications from the one-dimensional case.

#### Equidistribution principle and mesh alignment conditions

In higher dimensions, the equidistribution principle will no longer uniquely determine the transformation of the mesh. Restricting ourselves to the case of a triangular mesh, each cell will be uniquely determined by its size, shape and orientation; equivalently, a cell can be described by its size and circumscribed ellipsoid. The equidistribution principle only addresses the mesh size. Consequently, in [HR10] a mesh alignment condition is introduced into the mesh generation process.
Consider a domain $\Omega \subset \mathbb{R}^d$, discretized by a mesh $T_h$. Mathematically speaking, the size and circumscribed ellipsoid can be controlled as follows:

1. All elements have a constant volume in the metric M, i.e,
$$\int_K \rho(\vec{x})\, d\vec{x} = \frac{\sigma}{N}, \quad \forall K \in T_h,$$
where $N$ is the number of the elements of $T_h$ and $\sigma = \int_\Omega \rho(\vec{x})\, d\vec{x}$.

2. All elements are equilateral in the metric M. Letting $\gamma_1, \dots, \gamma_{d(d+1)/2}$ be the edges of an element $K$, this condition requires
$$|\gamma_1|_{\mathrm{M}} = \ldots = |\gamma_{d(d+1)/2}|_{\mathrm{M}}, \quad \forall K \in T_h,$$
where $|\gamma_i|_M$ denotes the length of the edge $\gamma_i$ in the metric M.

A uniform mesh in the metric M, i.e, a mesh with cells of constant volume and equilateral in the metric M, is called M-uniform.

We simplify the problem by working with an averaged value of the monitor function $M_K$ on each cell $K$ of the mesh $T_h$, where

$$M_K = \frac{1}{|K|}\int_K M(\vec{x})\,d\vec{x}.$$

Working with this averaged value of the monitor function, the two previous conditions are re-formulated as follows:

$$\int_K \rho_K\,d\vec{x} = \frac{\sigma_h}{N}, \quad \forall K \in T_h,$$
$$|\gamma_1|_{M_K} = \ldots = |\gamma_{d(d+1)/2}|_{M_K}, \quad \forall K \in T_h,$$

with the notations

$$\rho_K = \sqrt{\det(M_K)}, \quad \sigma_h = \sum_{K\in T_h}\int_K \rho_K\,d\vec{x}.$$

Since $M_K$, and consequently $\rho_K$, is constant, the equidistribution condition simplifies to

$$\rho_K\,|K| = \frac{\sigma_h}{N}. \tag{2.18}$$

After more lengthy calculations [HR10], we obtain the equivalent formulation for the mesh alignment condition

$$\frac{1}{d}\mathrm{tr}\left((F_K')^T M_K F_K'\right) = \det\left((F_K')^T M_K F_K'\right)^{\frac{1}{d}}, \quad \forall K \in T_h, \tag{2.19}$$

where $F_K'$ denotes the Jacobian matrix of mapping $F_K : \hat{K} \to K$, with $\hat{K}$ the reference cell for the mesh $T_h$.

The geometry of the mesh, through the size, shape and orientation of each cell, is controlled by the monitor function. Specifically, we have

**Theorem 2.1.2** ([HR10])**.** *The size and circumscribed ellipsoids of the elements of an M-uniform mesh $T_h$ satisfying (2.18) and (2.19) are completely determined from the monitor function M. More specifically, the size of mesh elements is inversely proportional to*

$\rho_K = \sqrt{\det(M_k)}$. *Also, the principal axes of the circumscribed ellipsoid of any element $K \in T_h$ are formed by the eigenvectors of $M_K$ and their semi-lengths are determined by*

$$a_i = \frac{\hat{h}}{2}\left(\frac{\sigma_h}{N}\right)^{\frac{1}{d}} \frac{1}{\sqrt{\lambda_i}}, \quad i = 1, \ldots, d.$$

*Furthermore,*

$$h_K \leq \hat{h}\left(\frac{\sigma_h}{N}\right)^{\frac{1}{d}} \frac{1}{\sqrt{\lambda_{min}}}, \quad \forall K \in T_h,$$

*where $\hat{h}$ is given by*

$$\hat{h} = 2\sqrt{\frac{d}{d+1}}\left(\frac{d!}{\sqrt{d+1}}\right)^{\frac{1}{d}},$$

*and $\lambda_{\min}$ is the minimum eigenvalue of $M_K$.*

In the perspective of controlling the equidistribution and mesh alignment conditions through a mesh functional, we extend the definitions (2.18) and (2.19) to a continuous setting ((2.20) and (2.21) respectively) for the coordinate transformation $\vec{x} = \vec{x}(\vec{\xi}) : \Omega_c \to \Omega$.

$$J\rho = \frac{\sigma}{|\Omega_c|}, \tag{2.20}$$

$$\frac{1}{d}\operatorname{tr}\left(\mathbf{J}^{-1}M^{-1}\mathbf{J}^{-T}\right) = \det\left(\mathbf{J}^{-1}M^{-1}\mathbf{J}^{-T}\right)^{\frac{1}{d}}, \tag{2.21}$$

where

$$\sigma = \int_\Omega \rho(\vec{x})\, d\vec{x}, \quad \rho(\vec{x}) = \sqrt{\det(M(\vec{x}))}.$$

## MMPDEs in higher dimensions

Similar to the one-dimensional case, various MMPDEs can be derived in higher dimensions. We now proceed to write general MMPDEs whose form depends upon the functional chosen.

As discussed in Section 2.1.1, MMPDEs should generally be based on the inverse coordinate transformation, which not only guarantees existence and uniqueness of the solution, but also prevents the eventual folding of the mesh. Let $I[\vec{\xi}]$ be a general functional

$$I[\vec{\xi}] = \int_\Omega F(\nabla\vec{\xi}, \vec{\xi}, \vec{x})\, d\vec{x}.$$

Equivalently, the functional can be expressed in terms of the contravariant base vectors $\vec{a}^i = \nabla \xi_i$ and the Jacobian $J$ by

$$I[\vec{\xi}] = \int_\Omega F(\vec{a}^1, \vec{a}^2, \vec{a}^3, J, \vec{x})\, d\vec{x}. \tag{2.22}$$

Because the coordinate transformation solution is a minimizer of that functional, it cancels the first variation of $I[\vec{\xi}]$. Skipping intermediate steps that are provided in Huang and Russell [HR10], the first variation of functional (2.22) is given by

$$\delta I[\vec{\xi}] = -\int_\Omega \sum_{i=1}^{3} \nabla \cdot \left[ \frac{\partial F}{\partial \vec{a}^i} - J \frac{\partial F}{\partial J} \vec{a}_i \right] d\xi_i\, d\vec{x} + \int_{\partial\Omega} \sum_{i=1}^{3} \vec{n} \cdot \left[ \frac{\partial}{\partial \vec{a}^i} - J \frac{\partial F}{\partial J} \vec{a}_i \right] d\xi_i\, dS.$$

Arguing that the first variation must cancel for all admissible $\delta\xi_i$, we obtain the Euler-Lagrange equation

$$-\nabla \cdot \left[ \frac{\partial F}{\partial \vec{a}^i} - J \frac{\partial F}{\partial J} \vec{a}_i \right] = 0, \quad i = 1,2,3. \tag{2.23}$$

For practicality, we want to formulate that equation in terms of the direct coordinate transformation. After lengthy calculations [HR10], we obtain the elliptic PDE

$$\sum_{i,j} A_{i,j} \frac{\partial^2 \vec{x}}{\partial \xi_i \partial \xi_j} + \sum_i B_i \frac{\partial \vec{x}}{\partial \xi_i} = 0, \tag{2.24}$$

where

$$\begin{aligned} A_{i,j} =& \sum_{k,s} \left( (\vec{a}^i)^T \cdot \frac{\partial^2 F}{\partial \vec{a}^s \partial \vec{a}^k} \cdot \vec{a}^j \right) \vec{a}_s \cdot (\vec{a}^k)^T - J \sum_s \left( (\vec{a}^i)^T \cdot \frac{\partial^2 F}{\partial \vec{a}^s \partial J} \right) \vec{a}_s \cdot (\vec{a}^j)^T \qquad (2.25) \\ & - J \sum_k \left( (\vec{a}^i)^T \cdot \frac{\partial^2 F}{\partial J \partial \vec{a}^k} \right) \vec{a}_i \cdot (\vec{a}^k)^T + J^2 \left( \frac{2}{J} \frac{\partial F}{\partial J} + \frac{\partial^2 F}{\partial J^2} \right) \vec{a}_i \cdot (\vec{a}^j)^T, \\ B_i =& -\sum_k \left( (\vec{a}^k)^T \cdot \frac{\partial^2 F}{\partial \vec{a}^i \partial \vec{x}} \cdot \vec{a}_k \right) + J \vec{a}_i \cdot \left( \frac{\partial^2 F}{\partial J \partial \vec{x}} \right)^T. \qquad (2.26) \end{aligned}$$

We turn the elliptic PDE (2.24) into an MMPDE using the gradient flow equation

$$\frac{\partial \vec{\xi}}{\partial t} = -\frac{1}{\tau p(\vec{x},t)} \frac{\delta I}{\delta \vec{\xi}},$$

where $p(\vec{x},t)$ is a balancing function, $\tau > 0$ is a parameter to modify the time-scale of the mesh movement and $\frac{\delta I}{\delta \vec{\xi}}$ is the functional derivative of $I$. The functional derivative is directly given from equation (2.23). This leads to the MMPDE

$$\frac{\partial \xi_i}{\partial t} = \frac{1}{\tau p(\vec{x},t)} \nabla \cdot \left[ \frac{\partial F}{\partial \vec{a}^i} - J \frac{\partial F}{\partial J} \vec{a}_i \right].$$

Interchanging the coordinates to derive an MMPDE in terms of the direct coordinate transformation $\vec{x}$, we obtain MMPDE5:

$$\dot{\vec{x}} = \frac{1}{\tau p(\vec{x},t)} \left[ \sum_{i,j} A_{i,j} \frac{\partial^2 \vec{x}}{\partial \xi_i \partial \xi_j} + \sum_i B_i \frac{\partial \vec{x}}{\partial \xi_i} \right], \tag{2.27}$$

with $A_{i,j}$ and $B_i$ defined respectively by equations (2.25) and (2.26).

**Choice of the functional**

An ideal functional for mesh movement will control both the equidistribution and mesh alignment. A clear and intelligible way to achieve this is by combining the functionals coming from each principle. From condition (2.20) one can derive the functional

$$I_{\text{eq}}[\vec{\xi}] = \int_\Omega \frac{\rho}{(J\rho)^\gamma} d\vec{x} \tag{2.28}$$

imposing the equidistribution principle only. Condition (2.21) leads to the functional

$$I_{\text{ali}}[\vec{\xi}] = \frac{1}{2} \int_\Omega \rho \left( \sum_i (\nabla \xi_i)^T M^{-1} \nabla \xi_i \right)^{\frac{d}{2}} d\vec{x} \tag{2.29}$$

for the mesh alignment. The dimensionally balanced functional imposing both principles is then a weighted average of both functionals (2.28) and (2.29),

$$I_{\text{Hua}}[\vec{\xi}, \theta, \gamma] = \theta \int_\Omega \rho \left( \sum_i (\nabla \xi_i)^T M^{-1} \nabla \xi_i \right)^{\frac{d\gamma}{2}} + (1 - 2\theta) d^{\frac{d\gamma}{2}} \int_\Omega \frac{\rho}{(J\rho)^\gamma} d\vec{x}.$$

A simpler and commonly used functional in practice is

$$I_{\text{Win}}[\vec{\xi}] = \frac{1}{2} \int_\Omega \frac{1}{\omega} \sum_i (\nabla \xi_i)^T \nabla \xi_i \, d\vec{x} = \frac{1}{2} \int_\Omega \frac{1}{\omega} \sum_i \|\nabla \xi_i\|^2 d\vec{x}. \tag{2.30}$$

The functional (2.30) is a weighted version of the Dirichlet energy used in harmonic maps. By looking for a minimizer of that functional, we are looking for a minimizer of the deformations in the mapping $\vec{\xi}$ under the constraints of the monitor function $\omega$, though we have no indications about how those deformations are minimized. Functional (2.30) is also very similar to the mesh alignment functional (2.29) with a diagonal monitor function $M = \omega I$. However the missing factor $\rho$ produces a completely different behaviour, incorporating both equidistribution and mesh alignment conditions (for details, see [HR10]). The Euler-Lagrange equation corresponding to the function (2.30) is

$$-\nabla\cdot\left(\frac{1}{\omega}\nabla\xi_i\right) = 0, \quad i = 1,2,\ldots,d. \tag{2.31}$$

Equation (2.31) corresponds to a diffusion process with a variable coefficient, so methods based on functional (2.30) are called variable diffusion methods.

## 2.2 Finite-Element Formulation of an MMPDE

In this thesis, we solve MMPDE5 (2.27) using Winslow's monitor function, in two dimensions, on a square domain $[-1,1]\times[-1,1]$. The computational domain is referenced by the coordinates $(\xi,\eta)$ while the physical domain is expressed in terms of the $(x,y)$ coordinates. Winslow's monitor function is an isotropic diagonal monitor function [CHR99]. It is given in two dimensions by

$$G_1 = G_2 = \begin{bmatrix} \omega & 0 \\ 0 & \omega \end{bmatrix} = \omega(\vec{x})I_2,$$

where $I_2$ is the $2\times 2$ identity matrix and $\omega$ is a strictly positive function on the domain. For now, the monitor function is assumed to be time-independent.

### 2.2.1 MMPDE5 with Winslow's monitor function

In two dimensions, MMPDE5 (2.27) reduces to

$$\tau p(\vec{x},t)\dot{\vec{x}} = A_{1,1}\frac{\partial^2\vec{x}}{\partial\xi^2} + (A_{1,2}+A_{2,1})\frac{\partial^2\vec{x}}{\partial\xi\partial\eta} + A_{2,2}\frac{\partial^2\vec{x}}{\partial\eta^2} + B_1\frac{\partial\vec{x}}{\partial\xi} + B_2\frac{\partial\vec{x}}{\partial\eta}. \tag{2.32}$$

We use the functional

$$I[\vec{\xi}] = \frac{1}{2}\int_\Omega \left\{(\nabla\xi)^T \cdot G_1^{-1} \cdot \nabla\xi + (\nabla\eta)^T \cdot G_2^{-1} \cdot \nabla\eta\right\} d\vec{x} \tag{2.33}$$

taken from Cao et al. [CHR99]. It is a popular choice that greatly simplifies the resulting MMPDE while imposing both equidistribution and mesh alignment conditions. Combined with Winslow's monitor function, functional (2.33) leads to

$$I[\vec{\xi}] = \frac{1}{2}\int_\Omega \frac{1}{\omega(\vec{x})}\left(\|\nabla\xi\|^2 + \|\nabla\eta\|^2\right) d\vec{x}. \tag{2.34}$$

Notice that this is precisely the functional (2.30) corresponding to the variable diffusion method. The integrand $F(\nabla\vec{\xi}, \vec{\xi}, \vec{x})$ corresponding to the functional (2.34) can be expressed as a function of the contravariant vectors $\vec{a}^i$ and the coordinate transformation $\vec{x}(\vec{\xi})$. Indeed, the first two contravariant vectors, $\vec{a}^1$ and $\vec{a}^2$, correspond to the gradients of the inverse coordinate transformation:

$$\vec{a}^1 = \nabla\xi, \quad \vec{a}^2 = \nabla\eta.$$

Using these notations, we re-write the integrand

$$F(\nabla\vec{\xi}, \vec{\xi}, \vec{x}) = \frac{1}{2\omega(\vec{x})}\left(\|\nabla\xi\|^2 + \|\nabla\eta\|^2\right)$$

as

$$F(\vec{a}^1, \vec{a}^2, \vec{x}) = \frac{1}{2\omega(\vec{x})}\left(\|\vec{a}^1\|^2 + \|\vec{a}^2\|^2\right). \tag{2.35}$$

Coefficients $A_{i,j}$ (2.25) and $B_i$ (2.26) depend on the functional through partial derivatives of the integrand $F$ (2.35). Obviously, partial derivatives involving the Jacobian are null. Following the convention on vector differentiation as defined in Appendix A, the other partial derivatives are given by

$$\begin{aligned}
\frac{\partial^2 F}{(\partial\vec{a}^1)^2} &= \begin{bmatrix} 1/\omega & 0 \\ 0 & 1/\omega \end{bmatrix} = \frac{\partial^2 F}{(\partial\vec{a}^2)^2}, \\
\frac{\partial^2 F}{\partial\vec{a}^1\partial\vec{a}^2} &= \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} = \frac{\partial^2 F}{\partial\vec{a}^2\partial\vec{a}^1}, \\
\frac{\partial^2 F}{\partial\vec{a}^i\partial\vec{x}} &= \begin{bmatrix} \frac{\partial}{\partial x}\left(\frac{1}{\omega}\right)\cdot\frac{\partial\xi_i}{\partial x} & \frac{\partial}{\partial y}\left(\frac{1}{\omega}\right)\cdot\frac{\partial\xi_i}{\partial x} \\ \frac{\partial}{\partial x}\left(\frac{1}{\omega}\right)\cdot\frac{\partial\xi_i}{\partial y} & \frac{\partial}{\partial y}\left(\frac{1}{\omega}\right)\cdot\frac{\partial\xi_i}{\partial y} \end{bmatrix} = \vec{a}^i\cdot\left(\nabla_{\vec{x}}\left(\frac{1}{\omega}\right)\right)^T.
\end{aligned} \tag{2.36}$$

Using (2.36) in equations (2.25) and (2.26), we get

$$\begin{aligned}
A_{i,j} &= \left((\vec{a}^i)^T \cdot \frac{\partial^2 F}{(\partial \vec{a}^1)^2} \cdot \vec{a}^j\right) \vec{a}_1 \cdot (\vec{a}^1)^T + \left((\vec{a}^i)^T \cdot \frac{\partial^2 F}{(\partial \vec{a}^2)^2} \cdot \vec{a}^j\right) \vec{a}_2 \cdot (\vec{a}^2)^T, \\
&= \frac{1}{\omega(\vec{x})} \left((\vec{a}^i)^T \cdot \vec{a}^j\right) \left(\vec{a}_1 \cdot (\vec{a}^1)^T + \vec{a}_2 \cdot (\vec{a}^2)^T\right), \qquad (2.37) \\
B_i &= -(\vec{a}^1)^T \cdot \frac{\partial^2 F}{\partial \vec{a}^i \partial \vec{x}} \cdot \vec{a}_1 - (\vec{a}^2)^T \cdot \frac{\partial^2 F}{\partial \vec{a}^i \partial \vec{x}} \cdot \vec{a}_2, \\
&= -(\vec{a}^1)^T \cdot \vec{a}^i \cdot \left(\nabla_{\vec{x}} \left(\frac{1}{\omega}\right)\right)^T \cdot \vec{a}_1 - (\vec{a}^2)^T \cdot \vec{a}^i \cdot \left(\nabla_{\vec{x}} \left(\frac{1}{\omega}\right)\right)^T \cdot \vec{a}_2. \qquad (2.38)
\end{aligned}$$

To obtain the unknowns in the coefficients $A_{i,j}$ and $B_i$ as the physical coordinates, we use the relation

$$\vec{a}^i = \frac{1}{J}(\vec{a}_j \times \vec{a}_k), \quad i,j,k \text{ cyclic} \qquad (2.39)$$

to interchange the coordinates, with $J$ denoting the Jacobian. In two dimensions, the Jacobian is given by

$$J = \det(\mathbf{J}) = \det\left(\frac{\partial \vec{x}}{\partial \vec{\xi}}\right) = \frac{\partial x}{\partial \xi}\frac{\partial y}{\partial \eta} - \frac{\partial y}{\partial \xi}\frac{\partial x}{\partial \eta}.$$

Equation (2.39) in two dimensions requires the substitution $\vec{a}_3 = \vec{a}^3 = (0,0,1)^T$ and adding a zero to the third entry of the other vectors. It gives

$$\begin{aligned}
\vec{a}^1 &= \frac{1}{J}\vec{a}_2 \times \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} = \frac{1}{J}\begin{pmatrix} x_\eta \\ y_\eta \\ (0) \end{pmatrix} \times \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} = \frac{1}{J}\begin{pmatrix} y_\eta \\ -x_\eta \\ (0) \end{pmatrix}, \\
\vec{a}^2 &= \frac{1}{J}\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \times \vec{a}_1 = \frac{1}{J}\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \times \begin{pmatrix} x_\xi \\ y_\xi \\ (0) \end{pmatrix} = \frac{1}{J}\begin{pmatrix} -y_\xi \\ x_\xi \\ (0) \end{pmatrix}.
\end{aligned} \qquad (2.40)$$

**Coefficients $B_i$**

The gradient in the physical coordinates that appears in (2.38) can similarly be modified to give

$$\begin{aligned}\left(\nabla_{\vec{x}}\left(\frac{1}{\omega}\right)\right)^T\cdot\vec{a}_i &= \frac{\partial}{\partial x}\left(\frac{1}{\omega}\right)\frac{\partial x}{\partial\xi_i}+\frac{\partial}{\partial y}\left(\frac{1}{\omega}\right)\frac{\partial y}{\partial\xi_i}=\frac{d}{d\xi_i}\left(\frac{1}{\omega}\right),\\ &= -\frac{1}{\omega^2}\frac{d\omega}{d\xi_i}.\end{aligned}\tag{2.41}$$

For ease of notation, we use the convention

$$\begin{aligned}\frac{d\omega}{d\xi} &= \frac{d}{d\xi}\omega\left(\vec{x}(\vec{\xi})\right)=\omega_\xi,\\ \frac{d\omega}{d\eta} &= \frac{d}{d\eta}\omega\left(\vec{x}(\vec{\xi})\right)=\omega_\eta.\end{aligned}$$

Using equations (2.41) and (2.38), we re-write $B_1$ and $B_2$ as

$$\begin{aligned}B_1 &= \frac{1}{J^2\omega^2}\left(x_\eta^2+y_\eta^2\right)\omega_\xi-\frac{1}{J^2\omega^2}\left(x_\xi x_\eta+y_\xi y_\eta\right)\omega_\eta,\\ B_2 &= -\frac{1}{J^2\omega^2}\left(x_\xi x_\eta+y_\xi y_\eta\right)\omega_\xi+\frac{1}{J^2\omega^2}\left(x_\xi^2+y_\xi^2\right)\omega_\eta.\end{aligned}$$

The expressions are factored to obtain

$$\begin{aligned}B_1 &= \frac{1}{J^2\omega^2}\left[\left(x_\eta^2+y_\eta^2\right)\omega_\xi-\left(x_\xi x_\eta+y_\xi y_\eta\right)\omega_\eta\right],\\ B_2 &= \frac{1}{J^2\omega^2}\left[-\left(x_\xi x_\eta+y_\xi y_\eta\right)\omega_\xi+\left(x_\xi^2+y_\xi^2\right)\omega_\eta\right].\end{aligned}\tag{2.42}$$

**Coefficients $A_{i,j}$**

The terms $\vec{a}_i\cdot(\vec{a}^i)^T$ can be formulated in terms of the physical coordinates as

$$\begin{aligned}\vec{a}_1\cdot(\vec{a}^1)^T &= \begin{pmatrix}x_\xi\\ y_\xi\end{pmatrix}\cdot\frac{1}{J}\begin{pmatrix}y_\eta\\ -x_\eta\end{pmatrix}^T=\frac{1}{J}\begin{bmatrix}+x_\xi y_\eta & -x_\xi x_\eta\\ +y_\xi y_\eta & -y_\xi x_\eta\end{bmatrix},\\ \vec{a}_2\cdot(\vec{a}^2)^T &= \begin{pmatrix}x_\eta\\ y_\eta\end{pmatrix}\cdot\frac{1}{J}\begin{pmatrix}-y_\xi\\ x_\xi\end{pmatrix}^T=\frac{1}{J}\cdot\begin{bmatrix}-x_\eta y_\xi & +x_\eta x_\xi\\ -y_\eta y_\xi & +y_\eta x_\xi\end{bmatrix}.\end{aligned}$$

and their sum results in the identity matrix

$$\vec{a}_1 \cdot (\vec{a}^1)^T + \vec{a}_2 \cdot (\vec{a}^2)^T = \frac{1}{J} \begin{bmatrix} x_\xi y_\eta - x_\eta y_\xi & 0 \\ 0 & -y_\xi x_\eta + y_\eta x_\xi \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = I_2.$$

We can thus simplify equation (2.37) for $A_{i,j}$ as

$$A_{i,j} = \frac{1}{\omega(\vec{x})} \left( (\vec{a}^i)^T \cdot \vec{a}^j \right) \cdot I_2, \tag{2.43}$$

implying

$$\begin{aligned} A_{1,1} &= \frac{1}{J^2 \omega} \left( x_\eta^2 + y_\eta^2 \right) I_2, \\ A_{1,2} &= -\frac{1}{J^2 \omega} \left( x_\xi x_\eta + y_\xi y_\eta \right) I_2 = A_{2,1}, \\ A_{2,2} &= \frac{1}{J^2 \omega} \left( x_\xi^2 + y_\xi^2 \right) I_2. \end{aligned} \tag{2.44}$$

**Assembling the MMPDE**

Using the above formulas derived for $A_{i,j}$ (2.44) and $B_i$ (2.42), we re-write MMPDE5 as

$$\begin{aligned} \tau p(\vec{x},t)\dot{\vec{x}} =& \frac{1}{J^2\omega} \left\{ \left( x_\eta^2 + y_\eta^2 \right) \frac{\partial^2 \vec{x}}{\partial \xi^2} - 2 \left( x_\xi x_\eta + y_\xi y_\eta \right) \frac{\partial^2 \vec{x}}{\partial \xi \partial \eta} + \left( x_\xi^2 + y_\xi^2 \right) \frac{\partial^2 \vec{x}}{\partial \eta^2} \right\} \\ &+ \frac{1}{J^2 \omega^2} \left\{ \left[ \omega_\xi \left( x_\eta^2 + y_\eta^2 \right) - \omega_\eta \left( x_\xi x_\eta + y_\xi y_\eta \right) \right] \frac{\partial \vec{x}}{\partial \xi} \right. \\ &\qquad \left. + \left[ \omega_\eta \left( x_\xi^2 + y_\xi^2 \right) - \omega_\xi \left( x_\xi x_\eta + y_\xi y_\eta \right) \right] \frac{\partial \vec{x}}{\partial \eta} \right\}, \\ =& \frac{1}{J^2 \omega^2} \left\{ \left( x_\eta^2 + y_\eta^2 \right) \left( \omega \frac{\partial^2 \vec{x}}{\partial \xi^2} + \omega_\xi \frac{\partial \vec{x}}{\partial \xi} \right) - \left( x_\xi x_\eta + y_\xi y_\eta \right) \left( 2\omega \frac{\partial^2 \vec{x}}{\partial \xi \partial \eta} \right. \right. \\ &\qquad \left. \left. + \omega_\eta \frac{\partial \vec{x}}{\partial \xi} + \omega_\xi \frac{\partial \vec{x}}{\partial \eta} \right) + \left( x_\xi^2 + y_\xi^2 \right) \left( \omega \frac{\partial^2 \vec{x}}{\partial \eta^2} + \omega_\eta \frac{\partial \vec{x}}{\partial \eta} \right) \right\}. \end{aligned}$$

This simplifies to give the final expression

$$\begin{aligned} \dot{\vec{x}} =& \frac{1}{J^2 \omega^2 \tau p(\vec{x},t)} \left\{ \left( x_\eta^2 + y_\eta^2 \right) \frac{\partial}{\partial \xi} \left( \omega \frac{\partial \vec{x}}{\partial \xi} \right) - \left( x_\xi x_\eta + y_\xi y_\eta \right) \frac{\partial}{\partial \eta} \left( \omega \frac{\partial \vec{x}}{\partial \xi} \right) \right. \\ &\qquad \left. - \left( x_\xi x_\eta + y_\xi y_\eta \right) \frac{\partial}{\partial \xi} \left( \omega \frac{\partial \vec{x}}{\partial \eta} \right) + \left( x_\xi^2 + y_\xi^2 \right) \frac{\partial}{\partial \eta} \left( \omega \frac{\partial \vec{x}}{\partial \eta} \right) \right\}. \end{aligned} \tag{2.45}$$

Since we are only interested in the final solution, for simplicity we start the computations with initial conditions corresponding to a uniform grid:

$$\begin{cases} x(\xi,\eta,0)=\xi, \\ y(\xi,\eta,0)=\eta. \end{cases}$$

**Boundary conditions**

Grid points along the boundaries of the domain are also moved adaptively [CHR99]. Their movement involves the solution to the one-dimensional MMPDE (2.3), where the one-dimensional monitor function $\rho(x)$ is given by

$$\rho(x)=\sqrt{\det G_i(x)}=\omega(x),$$

with $G_i=G_1=G_2$ the monitor functions for the two-dimensional case.
One can verify that the adaptivity along the boundaries and on the interior of the domain are consistent with each other. For instance, along a horizontal boundary, the differentiation of equation (2.3) with respect to $\xi$ gives

$$\frac{d}{d\xi}\left(\rho(x)\frac{dx}{d\xi}\right)=0,$$

while MMPDE5's steady-state solution along the same boundary will satisfy

$$\begin{aligned} \frac{1}{J^2\omega^2\tau p(\vec{x},t)}\left\{\left(x_\eta^2+y_\eta^2\right)\frac{\partial}{\partial\xi}\left(\omega\frac{\partial x}{\partial\xi}\right)\right\}&=\dot{\vec{x}}=0, \\ \Rightarrow\quad \frac{\partial}{\partial\xi}\left(\omega\frac{\partial x}{\partial\xi}\right)&=0. \end{aligned}$$

### 2.2.2 Derivation of the weak formulation

The MMPDE (2.45) for the $x$ and $y$ coordinates are independent and of the same form, so we only derive the equations for the $x$-direction.

Define $\alpha_1$, $\alpha_2$, $\alpha_3$ and $A$ by

$$\begin{aligned} \alpha_1(\vec{x})&=x_\eta^2+y_\eta^2, \\ \alpha_2(\vec{x})&=x_\xi x_\eta+y_\xi y_\eta, \\ \alpha_3(\vec{x})&=x_\xi^2+y_\xi^2, \\ A(\vec{x})&=J^2\omega^2\tau p(\vec{x},t). \end{aligned} \tag{2.46}$$

The weak formulation is obtained via the following procedure: we multiply each side of equation (2.45) by a function $v \in H_0^1(\Omega_c)$, where $H_0^1(\Omega_c)$ is the Sobolev space of functions that are compactly supported on $\Omega_c$ and have a weak derivative on that domain. We integrate over the whole domain, then integrate by parts the right-hand side to obtain:

$$\begin{aligned}
\int_{\Omega_c} A(\vec{x})\,(\dot{x}\cdot v)\,d\vec{\xi} = \int_{\Omega_c} \Bigg\{ & \alpha_1(\vec{x})\frac{\partial}{\partial \xi}\left(\omega(\vec{x})\frac{\partial x}{\partial \xi}\right)\cdot v \\
& - \left[\alpha_2(\vec{x})\frac{\partial}{\partial \eta}\left(\omega(\vec{x})\frac{\partial x}{\partial \xi}\right)\cdot v + \alpha_2(\vec{x})\frac{\partial}{\partial \xi}\left(\omega(\vec{x})\frac{\partial x}{\partial \eta}\right)\cdot v\right] \\
& + \alpha_3(\vec{x})\frac{\partial}{\partial \eta}\left(\omega(\vec{x})\frac{\partial x}{\partial \eta}\right)\cdot v \Bigg\}\, d\vec{\xi}, \\
\Leftrightarrow \int_{\Omega_c} A(\vec{x})\,(\dot{x}\cdot v)\,d\vec{\xi} = \int_{\Omega_c} \omega(\vec{x}) \Bigg\{ & -\frac{\partial}{\partial \xi}(\alpha_1(\vec{x})v)\cdot\frac{\partial x}{\partial \xi} \\
& + \left[\frac{\partial}{\partial \eta}(\alpha_2(\vec{x})v)\cdot\frac{\partial x}{\partial \xi} + \frac{\partial}{\partial \xi}(\alpha_2(\vec{x})v)\cdot\frac{\partial x}{\partial \eta}\right] \\
& - \frac{\partial}{\partial \eta}(\alpha_3(\vec{x})v)\cdot\frac{\partial x}{\partial \eta} \Bigg\}\, d\vec{\xi}.
\end{aligned} \tag{2.47}$$

### 2.2.3 Derivation of the finite-element formulation

Before deriving the finite-element formulation, we define the function space in which we are going to work.

**Theorem 2.2.1** ([Bra07]). *Let $k \geq 1$ and suppose $\Omega$ is bounded. Then a piecewise infinitely differentiable function $v : \bar{\Omega} \to \mathbb{R}$ belongs to $H^k(\Omega)$ if and only if $v \in C^{k-1}(\bar{\Omega})$.*

Finite elements that are $C^1$ on the whole domain are much harder to build and more expensive computationally than globally continuous finite elements. For this project, we use piecewise continuous polynomial finite elements that are continuous but not $C^1$ over the whole domain. These elements belong to $H^1(\Omega_c)$ (Theorem 2.2.1), and therefore do not necessarily have a weak second derivative. However the second derivatives of the solutions are required, in the weak formulation (2.47), to obtain the derivatives of the coefficients $\alpha_i$, $i = 1,2,3$ (2.46). In order to circumvent this difficulty, the derivatives of the solutions have to be computed using a recovery technique[1]. For this, we solve an alternate MMPDE in

[1]Cao, Weiming – University of Texas at San Antonio. Personal communication. March 31, 2011.

which the nonlinear terms $\alpha_i$, $i = 1,2,3$, and $A$ (2.46), are replaced with the approximations

$$\begin{aligned}\tilde{\alpha}_1(\vec{x}) &= \tilde{X}_\eta^2 + \tilde{Y}_\eta^2,\\ \tilde{\alpha}_2(\vec{x}) &= \tilde{X}_\xi \tilde{X}_\eta + \tilde{Y}_\xi \tilde{Y}_\eta,\\ \tilde{\alpha}_3(\vec{x}) &= \tilde{X}_\xi^2 + \tilde{Y}_\xi^2,\\ \tilde{A}(\vec{x}) &= \tilde{J}^2 \omega^2 \tau \tilde{p}(\vec{\tilde{X}}, t),\end{aligned}$$

where $\tilde{X}_\xi$, $\tilde{X}_\eta$, $\tilde{Y}_\xi$ and $\tilde{Y}_\eta$ are the approximations of $x_\xi$, $x_\eta$, $y_\xi$ and $y_\eta$ computed by the recovery technique.

## Recovery technique

Because we use piecewise continuous finite elements, their first derivatives are discontinuous at the cell edges. The recovery technique computes a continuous piecewise linear approximation of the first derivatives of the solutions. The approximations of the first derivatives connect the averaged values of those discontinuous derivatives at the vertices; in practice, the average is done on the shape functions by making their first derivatives continuous at the vertices. The approximations of the second derivatives are obtained by differentiation of the continuous approximations of the first derivatives.

## Time-stepping scheme

We use a time discretization

$$t_n = t_0 + n\Delta t, \quad n = 0,1,2,\ldots,$$

with $\Delta t$ the time step-size, and the solution at time $t_n$ is denoted by

$$\vec{x}_n = \vec{x}(t_n).$$

Two schemes are used for the time discretization; a forward Euler and a lagged backward Euler scheme. The lagged backward Euler discretization corresponds to a backward Euler discretization in which all nonlinear coefficients ($\tilde{\alpha}_i$'s and $\tilde{A}$) are evaluated at the current time step $t_n$ [CHR99]; the lagged backward Euler scheme applied to equation (2.47) gives

$$\int_{\Omega_c} \tilde{A}(\vec{x}_n)\left(\frac{x_{n+1}-x_n}{\Delta t}\cdot v\right) d\vec{\xi} = \int_{\Omega_c} \omega(\vec{x}_n)\Bigg\{ -\frac{\partial}{\partial \xi}\left(\tilde{\alpha}_1(\vec{x}_n)v\right)\cdot\frac{\partial x_{n+1}}{\partial \xi} + \left[\frac{\partial}{\partial \eta}\left(\tilde{\alpha}_2(\vec{x}_n)v\right)\cdot\frac{\partial x_{n+1}}{\partial \xi} + \frac{\partial}{\partial \xi}\left(\tilde{\alpha}_2(\vec{x}_n)v\right)\cdot\frac{\partial x_{n+1}}{\partial \eta}\right] - \frac{\partial}{\partial \eta}\left(\tilde{\alpha}_3(\vec{x}_n)v\right)\cdot\frac{\partial x_{n+1}}{\partial \eta}\Bigg\} d\vec{\xi}. \tag{2.48}$$

## Finite element definition

Given a mesh $\Omega_{c_h}$ over the domain $\Omega_c$, we denote by $V_h$ the set of piecewise linear functions on that mesh which are continuous over the whole domain $\Omega_c$. $V_h$ is a finite-dimensional space, for which we can find a basis $\left\{\varphi_j\right\}_j$. The problem is now to look for an approximation $x_h$ to $x$ in $V_h$. Decomposing $x_h$ in terms of the basis $\left\{\varphi_j\right\}_j$,

$$x_h(t_n) = \sum_j C_j(t_n)\varphi_j(\vec{\xi}), \tag{2.49}$$

we seek solutions to (2.47) in $V_h$. The equation must hold for any $v_h \in V_h$, or equivalently for any basis function $\varphi_i$ of $V_h$. It follows that we require

$$\sum_j \frac{C_j(t_{n+1})-C_j(t_n)}{\Delta t}\int_{\Omega_{c_h}} \tilde{A}(\vec{x}_h(t_n))\left(\varphi_j\cdot\varphi_i\right) d\vec{\xi} = \sum_j C_j(t_{n+1})\int_{\Omega_{c_h}} \omega(\vec{x}_h(t_n))\Bigg\{ -\frac{\partial}{\partial \xi}\left(\alpha_1(\vec{x}_h(t_n))\varphi_i\right)\cdot\frac{\partial \varphi_j}{\partial \xi} + \left[\frac{\partial}{\partial \eta}\left(\alpha_2(\vec{x}_h(t_n))\varphi_i\right)\cdot\frac{\partial \varphi_j}{\partial \xi} + \frac{\partial}{\partial \xi}\left(\alpha_2(\vec{x}_h(t_n))\varphi_i\right)\cdot\frac{\partial \varphi_j}{\partial \eta}\right] - \frac{\partial}{\partial \eta}\left(\alpha_3(\vec{x}_h(t_n))\varphi_i\right)\cdot\frac{\partial \varphi_j}{\partial \eta}\Bigg\} d\vec{\xi}, \tag{2.50}$$

for all $i$.

**Matrix form**

Let $\{C\}^n$ be the vector solution at time $t_n$, i.e,

$$\{C\}^n = \begin{pmatrix} C_1(t_n) \\ C_2(t_n) \\ \vdots \\ \vdots \end{pmatrix},$$

and let $[K_1]^n$ and $[K_2]^n$ be the two matrices arising from equation (2.50),

$$[K_1]^n = \left( \int_{\Omega_{c_h}} \tilde{A}(\vec{x}_h(t_n)) \left(\varphi_j \cdot \varphi_i\right) d\vec{\xi} \right)_{i,j},$$

$$\begin{aligned}[K_2]^n = \Bigg( \int_{\Omega_{c_h}} \omega(\vec{x}_h(t_n)) \Bigg\{ & -\frac{\partial}{\partial \xi}\left(\tilde{\alpha}_1(\vec{x}_h(t_n)) \cdot \varphi_i\right) \cdot \frac{\partial \varphi_j}{\partial \xi} \\ & + \left[ \frac{\partial}{\partial \eta}\left(\tilde{\alpha}_2(\vec{x}_h(t_n)) \cdot \varphi_i\right) \cdot \frac{\partial \varphi_j}{\partial \xi} + \frac{\partial}{\partial \xi}\left(\tilde{\alpha}_2(\vec{x}_h(t_n)) \cdot \varphi_i\right) \cdot \frac{\partial \varphi_j}{\partial \eta} \right] \\ & - \frac{\partial}{\partial \eta}\left(\tilde{\alpha}_3(\vec{x}_h(t_n)) \cdot \varphi_i\right) \cdot \frac{\partial \varphi_j}{\partial \eta} \Bigg\} d\vec{\xi} \Bigg)_{i,j}.\end{aligned}$$

Note that $[K_1]^n$ and $[K_2]^n$ have coefficients that depend on the solutions $x$ and $y$ at time $t_n$. Equation (2.50) can now be written in the matrix form

$$\begin{aligned} & [K_1]^n \cdot \frac{\{C\}^{n+1} - \{C\}^n}{\Delta t} = [K_2]^n \cdot \{C\}^{n+1}, \\ \Leftrightarrow \quad & \left([K_1]^n - \Delta t [K_2]^n\right) \cdot \{C\}^{n+1} = [K_1]^n \cdot \{C\}^n. \end{aligned} \tag{2.51}$$

At each time step $t_n$, we solve for $\{C\}^{n+1}$. The matrix $[K_1]^n - \Delta t \cdot [K_2]^n$ is non-symmetric, because of $[K_2]$, which necessitates the use of a nonsymmetric solver such as GMRES or BiCGStab. The latter was successfully used in Cao et al. [CHR99] and is also used for this work.

We can similarly derive the matrix form corresponding to a forward Euler time discretization. In this situation, the whole right-hand side of equation (2.50) is evaluated at time $t_n$. This leads to

$$\begin{aligned} & [K_1]^n \cdot \frac{\{C\}^{n+1} - \{C\}^n}{\Delta t} = [K_2]^n \cdot \{C\}^n, \\ \Leftrightarrow \quad & [K_1]^n \cdot \{C\}^{n+1} = \left([K_1]^n + \Delta t [K_2]^n\right) \cdot \{C\}^n. \end{aligned} \tag{2.52}$$

Because matrix $[K_1]$ is symmetric and positive definite, we use the conjugate gradient method to solve equation (2.52) at each time step $t_n$.

### 2.2.4 Time-dependent monitor function

The MMPDE (2.45) is used, without modification, for both time-dependent and time-independent problems. In the latter case, the MMPDE works as a regularization technique. When the monitor function varies in time, the MMPDE determines the dynamic evolution of the mesh. The initial mesh is computed as the solution to a time-independent problem.

## 2.3 Computation of Moving Meshes

### 2.3.1 Implementation of the finite-element formulation

In this section, we look at the technical details of the moving mesh formulation on a computer.

#### Boundary conditions

We solve the one-dimensional equidistribution equation (2.3) along the boundaries with de Boor's algorithm, and use the discrete equidistribution quality measure as a stopping criterion. The convergence of the equidistribution quality measure depends on both the number of iterations and the number of grid points used along the boundary [XHRW11]. In our numerical computations, we observed that it is sometimes impossible to achieve convergence without selecting a large number of grid points.
In the case of a time-dependent monitor function, the numbers of grid points are decided during the computation of the initial mesh; for the subsequent time steps, only the number of iterations of de Boor's algorithm changes.

#### Integration

Independently of what time discretization is chosen, matrices $[K_1]^n$ and $[K_2]^n$ must be assembled at each time step. In order to compute each entry of the matrices, the integral over the whole domain $\Omega_{c_h}$ is decomposed as the sum of the integrals over each cell $\Omega_{c_k}$:

$$\int_{\Omega_{c_h}} \tilde{A}(\vec{x}_h^n)\left(\varphi_j \cdot \varphi_i\right) d\vec{\xi} = \sum_k \int_{\Omega_{c_k}} \tilde{A}(\vec{x}_h^n)\left(\varphi_j \cdot \varphi_i\right) d\vec{\xi}. \tag{2.53}$$

The integral over each cell is then transformed, via a change of variable, into the integral over a reference cell $\Omega_r$. Let $\vec{\xi}_r = (\xi_r, \eta_r)$ be the coordinates on the reference cell and $\vec{F}_k$ the mapping from the reference cell to cell $k$. The Jacobian of that mapping is defined by

$$J = |\det(\vec{J})| = \left|\det\left(\frac{\partial \vec{\xi}}{\partial \vec{\xi}_r}\right)\right|.$$

For ease of notation, we introduce the function

$$B(\vec{\xi}) = \tilde{A}\big(\vec{x}_h^n(\vec{\xi})\big).$$

Using these expressions, the contribution to (2.53) becomes

$$\int_{\Omega_{c_k}} B(\vec{\xi})\left(\varphi_j(\vec{\xi})\cdot\varphi_i(\vec{\xi})\right) d\vec{\xi} = \int_{\Omega_r} B(\vec{F}_k(\vec{\xi}_r))\left(\varphi_j(\vec{F}_k(\vec{\xi}_r))\cdot\varphi_i(\vec{F}_k(\vec{\xi}_r))\right) J\, d\vec{\xi}_r. \tag{2.54}$$

## Quadrature

In order to evaluate integral (2.54) numerically, we use a Gaussian quadrature:

$$\int_{\Omega_r} B(\vec{F}_k(\vec{\xi}_r))\left(\varphi_j(\vec{F}_k(\vec{\xi}_r))\cdot\varphi_i(\vec{F}_k(\vec{\xi}_r))\right) J\, d\vec{\xi}_r = \sum_s B(\vec{F}_k(\vec{q}_s))\left(\varphi_j(\vec{F}_k(\vec{q}_s))\cdot\varphi_i(\vec{F}_k(\vec{q}_s))\right) J w_s\, d\vec{\xi}_r.$$

This quadrature positions points so as to obtain the highest degree of accuracy possible for a given number of quadrature points. The choice of an exact quadrature could only be made for polynomial functions, by looking at the highest polynomial degree.
Assuming temporarily that the monitor function is constant in equation (2.50), the right-hand side becomes polynomial. Its highest polynomial degree will be 4 (see Tables 2.1 to 2.4). Alone it would require a Gauss quadrature of order strictly greater than 2. Indeed, a Gauss quadrature of order $n$ integrates exactly polynomial expressions of degree at most $2n-1$. In the case of MMPDEs however, we also have to include the influence of the monitor function, which is, generally speaking, non polynomial. Thus theoretically, we should use a Gaussian quadrature of order at least 3 to assemble (2.51). In practice we used higher orders, often around 6 (see Sections 2.3.2 and 3.4.3).

Table 2.1: Polynomial degree of the first term in (2.50).

| Element | $\xi$ | $\eta$ |
|---|---|---|
| $\tilde{X}_\eta$ | 1 | 1 |
| $\tilde{X}_\eta^2$ | 2 | 2 |
| $\varphi_i$ | 1 | 1 |
| $(\tilde{X}_\eta^2 + \tilde{Y}_\eta^2)\varphi_i$ | 3 | 3 |
| $\partial_\xi \left((\tilde{X}_\eta^2 + \tilde{Y}_\eta^2)\varphi_i\right)$ | 2 | 3 |
| $\partial_\xi \varphi_j$ | 0 | 1 |
| $\partial_\xi \left((\tilde{X}_\eta^2 + \tilde{Y}_\eta^2)\varphi_i\right) \cdot \partial_\xi \varphi_j$ | 2 | 4 |

Table 2.2: Polynomial degree of the second term in (2.50).

| Element | $\xi$ | $\eta$ |
|---|---|---|
| $\tilde{X}_\xi$ | 1 | 1 |
| $\tilde{X}_\eta$ | 1 | 1 |
| $\tilde{X}_\xi \tilde{X}_\eta + \tilde{Y}_\xi \tilde{Y}_\eta$ | 2 | 2 |
| $\varphi_i$ | 1 | 1 |
| $(\tilde{X}_\xi \tilde{X}_\eta + \tilde{Y}_\xi \tilde{Y}_\eta)\varphi_i$ | 3 | 3 |
| $\partial_\eta \left((\tilde{X}_\xi \tilde{X}_\eta + \tilde{Y}_\xi \tilde{Y}_\eta)\varphi_i\right)$ | 3 | 2 |
| $\partial_\xi \varphi_j$ | 0 | 1 |
| $\partial_\eta \left((\tilde{X}_\xi \tilde{X}_\eta + \tilde{Y}_\xi \tilde{Y}_\eta)\varphi_i\right) \cdot \partial_\xi \varphi_j$ | 3 | 3 |

Table 2.3: Polynomial degree of the third term in (2.50).

| Element | $\xi$ | $\eta$ |
|---|---|---|
| $\partial_\xi \left((\tilde{X}_\xi \tilde{X}_\eta + \tilde{Y}_\xi \tilde{Y}_\eta)\varphi_i\right)$ | 2 | 3 |
| $\partial_\eta \varphi_j$ | 1 | 0 |
| $\partial_\xi \left((\tilde{X}_\xi \tilde{X}_\eta + \tilde{Y}_\xi \tilde{Y}_\eta)\varphi_i\right) \cdot \partial_\eta \varphi_j$ | 3 | 3 |

Table 2.4: Polynomial degree of the fourth term in (2.50).

| Element | $\xi$ | $\eta$ |
|---|---|---|
| $\tilde{X}_\xi^2$ | 2 | 2 |
| $\varphi_i$ | 1 | 1 |
| $(\tilde{X}_\xi^2 + \tilde{Y}_\xi^2)\varphi_i$ | 3 | 3 |
| $\partial_\eta \left( (\tilde{X}_\xi^2 + \tilde{Y}_\xi^2)\varphi_i \right)$ | 3 | 2 |
| $\partial_\eta \varphi_j$ | 1 | 0 |
| $\partial_\eta \left( (\tilde{X}_\xi^2 + \tilde{Y}_\xi^2)\varphi_i \right) \cdot \partial_\eta \varphi_j$ | 4 | 2 |

**Recovery technique**

The implementation of the recovery technique in the C++-library deal.II represents a significant amount of work. Because the information on the geometry of the mesh, the value of the shape functions at the vertices, or the correspondence between vertices and global degrees of freedom is not directly available to the user, all of this information has to be retrieved. This in itself imposes the choice of a two-step process where the missing information is gathered first, then used at each step of the iteration. In order to simplify the algorithm, some assumptions were made:

- We work in two dimensions.
- Piecewise linear finite elements are used.
- The domain is square.
- All cells are identical squares, with edges parallel to the axes $\xi$ and $\eta$.

The first step, happening before the iteration starts, consists of building the matrix $A$ converting the solution at the degrees of freedom into the values of the smooth first derivatives at the same degrees of freedom. During the computation, the values of the smooth derivatives are interpolated at each time step, to obtain the smooth first derivatives at the quadrature points of the cells; we use a linear interpolant for this step. The second derivatives are obtained by differentiating the smooth first derivatives.

Because of the assumptions made, the equations of the shape functions are well-known. On a cell bounded by the coordinates $\xi_i, \xi_j, \eta_i, \eta_j$, the shape function corresponding to the vertex $(\xi_i, \eta_i)$ is

$$\varphi_i(\xi,\eta) = \frac{\xi-\xi_j}{\xi_i-\xi_j} \cdot \frac{\eta-\eta_j}{\eta_i-\eta_j}, \quad \forall(\xi,\eta) \in [\xi_i,\xi_j] \times [\eta_i,\eta_j].$$

In order to illustrate the technicalities of the recovery technique, we concentrate on a specific vertex $(\xi_i, \eta_i)$, surrounded by 4 cells numbered counter-clockwise from the top left (see Figure 2.2). The equations of the shape functions with value 1 at the vertex $(\xi_i, \eta_i)$ are given by

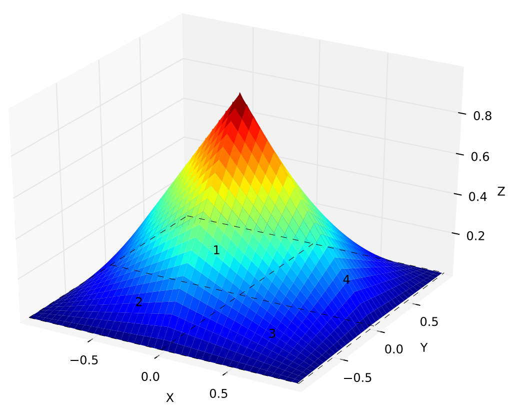
$$\begin{aligned}
\varphi_i^{(1)}(\xi,\eta) &= \frac{\xi-\xi_{i-1}}{h} \cdot \frac{\eta-\eta_{i+1}}{-h} = \frac{-1}{h^2}(\xi-\xi_{i-1}) \cdot (\eta-\eta_{i+1}),\\
\varphi_i^{(2)}(\xi,\eta) &= \frac{\xi-\xi_{i-1}}{h} \cdot \frac{\eta-\eta_{i-1}}{h} = \frac{1}{h^2}(\xi-\xi_{i-1}) \cdot (\eta-\eta_{i-1}),\\
\varphi_i^{(3)}(\xi,\eta) &= \frac{\xi-\xi_{i+1}}{-h} \cdot \frac{\eta-\eta_{i-1}}{h} = \frac{-1}{h^2}(\xi-\xi_{i+1}) \cdot (\eta-\eta_{i-1}),\\
\varphi_i^{(4)}(\xi,\eta) &= \frac{\xi-\xi_{i+1}}{-h} \cdot \frac{\eta-\eta_{i+1}}{-h} = \frac{1}{h^2}(\xi-\xi_{i+1}) \cdot (\eta-\eta_{i+1}).
\end{aligned} \tag{2.55}$$

We first derive the formulas for the partial derivatives with respect to $\xi$. After differentiating equations (2.55), we obtain the following:

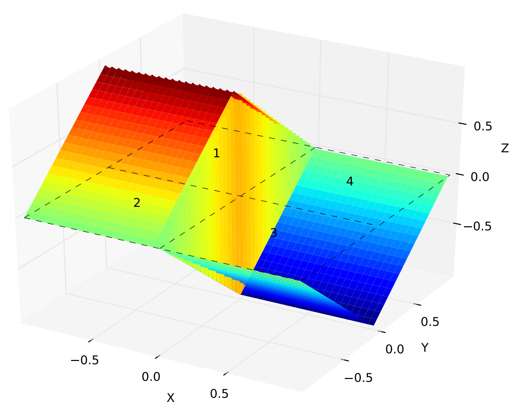
$$\begin{aligned}
\partial_\xi \varphi_i^{(1)}(\xi,\eta) &= \frac{-1}{h^2}(\eta-\eta_{i+1}),\\
\partial_\xi \varphi_i^{(2)}(\xi,\eta) &= \frac{1}{h^2}(\eta-\eta_{i-1}),\\
\partial_\xi \varphi_i^{(3)}(\xi,\eta) &= \frac{-1}{h^2}(\eta-\eta_{i-1}),\\
\partial_\xi \varphi_i^{(4)}(\xi,\eta) &= \frac{1}{h^2}(\eta-\eta_{i+1}).
\end{aligned} \tag{2.56}$$

The shape function surrounding the point $(0,0)$ and its $\xi$-derivative (2.56) are plotted in Figure 2.1 for unit cells ($h = 1$). The derivative in $\xi$ is discontinuous along the lines $\xi = -1$, 0 and 1 (see Figure 2.1b). Indeed, along the line $\eta = \eta_i$, we have the following equalities:

$$\begin{aligned}
\partial_\xi \varphi_i^{(1)}(\xi,\eta_i) &= \frac{1}{h} = \partial_\xi \varphi_i^{(2)}(\xi,\eta_i),\\
\partial_\xi \varphi_i^{(3)}(\xi,\eta_i) &= -\frac{1}{h} = \partial_\xi \varphi_i^{(4)}(\xi,\eta_i).
\end{aligned}$$



(a) Shape functions (2.55)



(b) First derivatives in $\xi$ (2.56)

Figure 2.1: Plots of the shape functions around vertex $(0,0)$ with cells of width $h=1$. Cell numbers are indicated.

These derivatives have to be averaged at the 3 vertices $(-1,0)$, $(0,0)$ and $(1,0)$. The values of these averaged derivatives are given by

$$\begin{aligned} \partial_\xi \varphi_i(-1,0) &= \frac{1}{2h}, \\ \partial_\xi \varphi_i(0,0) &= 0, \\ \partial_\xi \varphi_i(1,0) &= -\frac{1}{2h}, \end{aligned} \tag{2.57}$$

while the values at the other vertices are all zero. Similar to equations (2.55), the smooth derivative consists of a linear interpolation between the values defined by equation (2.57); the values at the vertices are marked on Figure 2.2. The smooth derivative is plotted in Figure 2.3.

The averaged values shown on Figure 2.2 will be modified, based on the proximity of the cells to the boundary of the domain. The principal situations are detailed here, with the other cases being easy to deduce by symmetry.

- Figure 2.4 represents the case of cells $(4)$ and $(3)$ being along the left boundary of the domain (i.e, the vertex of interest on the left boundary).
- Figure 2.5 represents the case of cells $(1)$ and $(2)$ being along the left boundary of the domain (i.e, the vertex of interest one cell away from the left boundary).

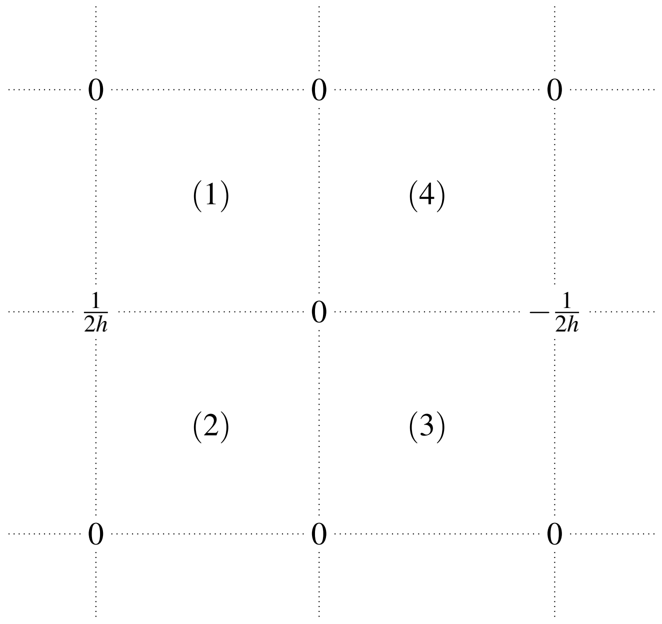


Figure 2.2: Sketch of the averaged values, at the vertices, of the $\xi$-derivative of the shape functions corresponding to the central vertex. Cell numbers are plotted.

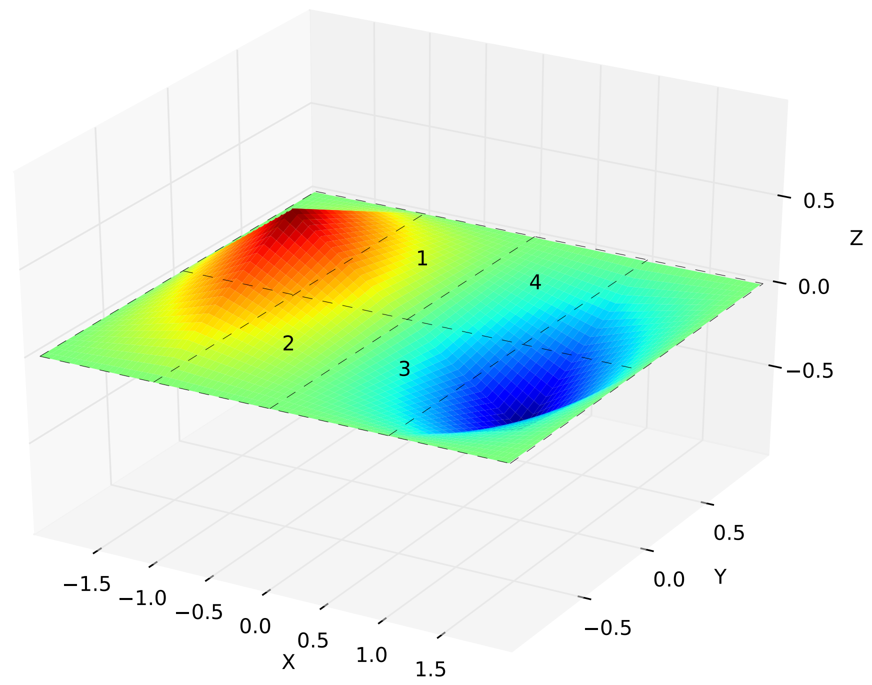


Figure 2.3: Plot of the smoothed first derivative in $\xi$ of the shape functions around vertex $(0,0)$, with cells of width $h = 1$. Cell numbers are indicated.

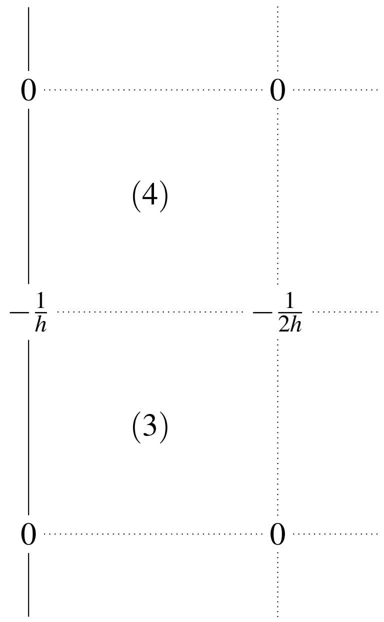


Figure 2.4: Cells $(4)$ and $(3)$ along the left boundary of the domain.

- Figure 2.6 represents the case of cells $(2)$ and $(3)$ being along the top boundary of the domain (i.e, the vertex of interest on the top boundary).
- Figure 2.7 represents the case of cell $(1)$ being in the lower right corner (i.e, the vertex of interest in the lower right corner).

In the $\eta$ direction, we obtain similar diagrams and smoothing patterns, but rotated at 90 degrees. The derivatives of the 4 shape functions surrounding a common vertex are given by

$$\begin{aligned}
\partial_\eta \varphi_i^{(1)}(\xi,\eta) &= \frac{-1}{h^2}(\xi-\xi_{i-1}),\\
\partial_\eta \varphi_i^{(2)}(\xi,\eta) &= \frac{1}{h^2}(\xi-\xi_{i-1}),\\
\partial_\eta \varphi_i^{(3)}(\xi,\eta) &= \frac{-1}{h^2}(\xi-\xi_{i+1}),\\
\partial_\eta \varphi_i^{(4)}(\xi,\eta) &= \frac{1}{h^2}(\xi-\xi_{i+1}).
\end{aligned}$$

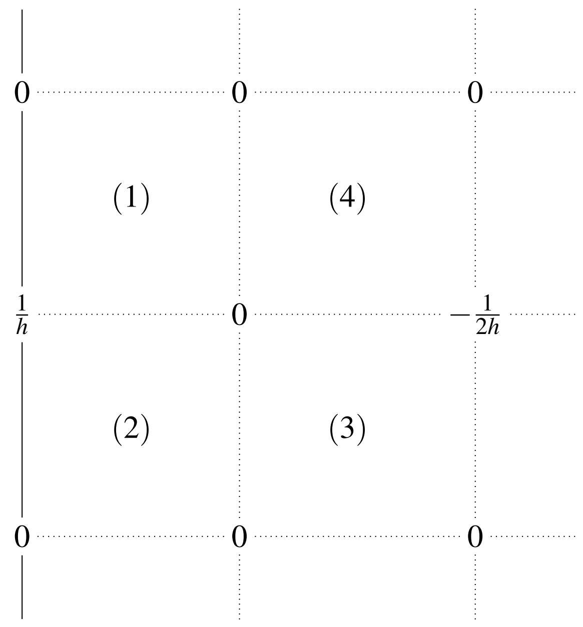


Figure 2.5: Cells $(1)$ and $(2)$ along the left boundary of the domain.

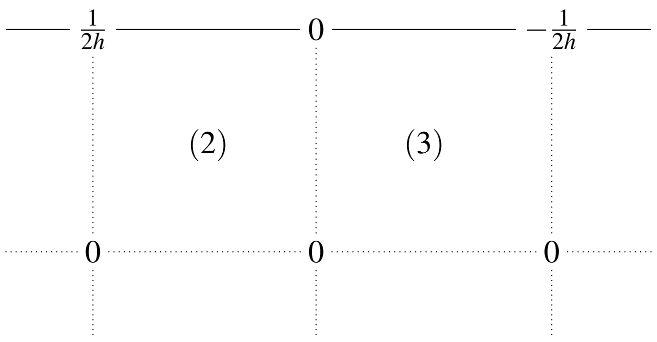


Figure 2.6: Cells $(2)$ and $(3)$ along the top boundary of the domain.

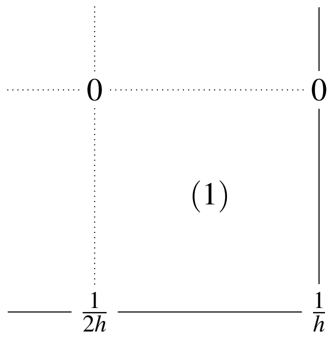


Figure 2.7: Cell $(1)$ in the lower right corner.

We now have a discontinuity along the boundary $\eta = \eta_i$,

$$\partial_\eta \varphi_i^{(1)}(\xi_i, \eta) = \frac{-1}{h} = \partial_\eta \varphi_i^{(4)}(\xi_i, \eta),$$
$$\partial_\eta \varphi_i^{(2)}(\xi_i, \eta) = \frac{1}{h} = \partial_\eta \varphi_i^{(3)}(\xi_i, \eta).$$

This is shown in the sketch in Figure 2.8, which corresponds to the case of a general vertex being at least two cells away from any boundaries of the domain. The modifications for a cell close to the boundary are similar to the ones for the $\xi$-derivative.

**deal.II: a finite element differential equations analysis library**

deal.II ([BK], [BHK07]) is a C++-library designed for the numerical solution of partial differential equations using adaptive finite-element methods. It is available free of charge, under an Open Source licence, from the deal.II website [BK]. It was developed at the Numerical Methods Group at Universität Heidelberg, Germany, which specializes in *h*, *p* and *hp*-adaptive methods. However, deal.II is not restricted to those types of applications. deal.II greatly simplifies the implementation and solution of a finite-element formulation by including a general mesher, a large library of commonly used finite elements (e.g, piecewise polynomials, Nedelec elements [Ned80] and Raviart-Thomas elements), a complete stand-alone linear algebra library and support for several output formats. The extensive use of C++ templates allows the user to write programs that are dimension independent.
deal.II is used in many academic or commercial projects, and the work of its original au-

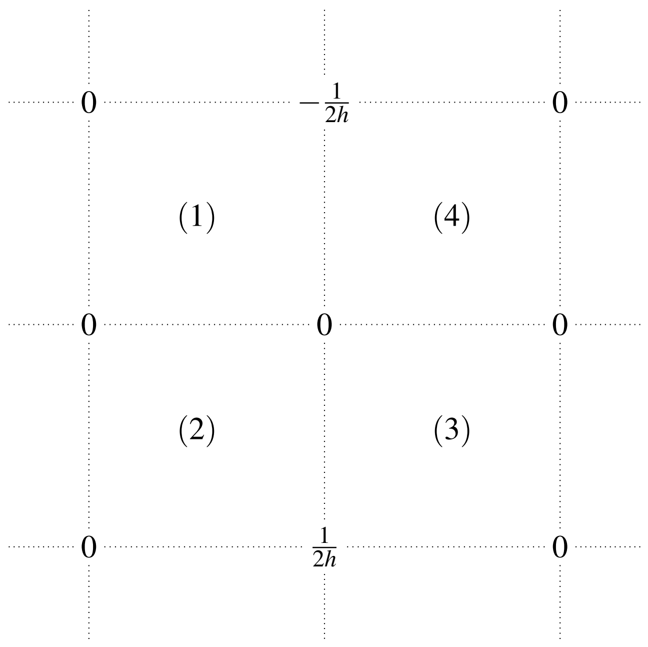


Figure 2.8: Sketch of the averaged values at the vertices of the $\eta$-derivative of the shape functions corresponding to the central vertex. Cell numbers are plotted.

thors was awarded the 2007 J. H. Wilkinson Prize for Numerical Software.

**Note on the parameters of MMPDE5**

The parameter $\tau$ is used to modify the time-scale of the moving mesh computation, when solving a physical PDE simultaneously. In this project, we are only interested in the computation of the mesh, and the factor $\tau$ affects solely the value of the time step $\Delta t$. Therefore, we define an effective time step

$$\tilde{\Delta t} = \frac{\Delta t}{\tau}.$$

In practice, the parameter $\tau$ will always be equal to 1 and we will not distinguish between $\Delta t$ and $\tilde{\Delta t}$.

Different choices for the balancing function $\tilde{p}(\vec{\tilde{X}},t)$ are possible, and we use

$$\tilde{p}(\vec{\tilde{X}},t) = \sqrt{\sum_{i,j} \|A_{i,j}\|_F^2} = \frac{\sqrt{2}}{J^2\omega}\sqrt{\tilde{\alpha}_1(\vec{\tilde{X}})^2 + 2\tilde{\alpha}_2(\vec{\tilde{X}})^2 + \tilde{\alpha}_3(\vec{\tilde{X}})^2},$$

with $\|\cdot\|_F$ the Froebenius norm.

### 2.3.2 Computational results

We present a convergence analysis of the deal.II implementation and various examples with time-varying monitor functions. The time-independent problem is solved using a regularization technique, and for it we only need to check the spatial convergence.

**Time-independent monitor function**

We analyze the convergence of our discretization toward an exact solution to MMPDE5 (2.45); because of the complexity of that PDE, we derive an exact solution to a simplified version of (2.45) that depends on only one spatial direction at a time.

Let's assume that the monitor function only depends on the variable $x$, i.e $\omega(\vec{x}) = \omega(x)$, and we have a similar symmetry in the solutions $x$ and $y$:

$$\begin{aligned} x(\vec{\xi}) &= x(\xi), \\ y(\vec{\xi}) &= y(\eta). \end{aligned}$$

The monitor function, hence the adaptivity, only depends on the solution $x$, so the second solution $y$ is not affected by the mesh movement and we have

$$\begin{aligned} x(\vec{\xi}) &= x(\xi), \\ y(\vec{\xi}) &= \eta. \end{aligned}$$

This implies

$$\begin{aligned} x_\eta &= 0, \\ y_\xi &= 0, \text{ and } y_\eta = 1. \end{aligned}$$

Plugging these derivatives into equation (2.45), the steady-state solution for $x$ becomes

$$\begin{cases} \dfrac{d}{d\xi}\left(\omega \dfrac{dx}{d\xi}\right) = 0, \\ x(-1) = -1, \quad x(1) = 1. \end{cases} \tag{2.58}$$

We solve equation (2.58) using the method of separation of variables:

$$\begin{aligned} & \frac{d}{d\xi}\left(\omega \frac{dx}{d\xi}\right) = 0, \\ \Leftrightarrow \quad & \omega \frac{dx}{d\xi} = K_1, \quad K_1 \in \mathbb{R}, \\ \Leftrightarrow \quad & \omega(x)dx = K_1 d\xi, \quad K_1 \in \mathbb{R}, \\ \Leftrightarrow \quad & \int \omega(x)dx = K_1\xi + K_2, \quad K_1, K_2 \in \mathbb{R}. \end{aligned} \tag{2.59}$$

At this stage, the mesh only depends on the choice of the monitor function $\omega$; the only restriction being that $\omega(x) > 0$ on the domain. To illustrate the mathematical idea and keep the calculations simple, we choose

$$\omega(\vec{x}) = \exp(x). \tag{2.60}$$

The monitor function (2.60) leads to the solution

$$\begin{cases} x(\vec{\xi}) = \ln\left(K_1\xi + K_2\right), \\ y(\vec{\xi}) = \eta, \end{cases} \tag{2.61}$$

with $K_1$ and $K_2$ depending on the boundary conditions. Applying the boundary conditions (2.58) to equation (2.59), we obtain the system of equations

$$\begin{cases} e = K_1 + K_2, \\ e^{-1} = -K_1 + K_2, \end{cases}$$

which has the solution

$$K_1 = \frac{e^2-1}{2e}, \quad \text{and} \quad K_2 = \frac{e^2+1}{2e}. \tag{2.62}$$

Using (2.62), we start the computation with the initial conditions

$$\begin{cases} x(\vec{\xi}) = \xi, \\ y(\vec{\xi}) = \ln(K_1\eta + K_2). \end{cases}$$

The steady-state solution for $y$ is a uniform mesh.
To test the $y$-component, we use the monitor function

$$\omega(\vec{x}) = \exp(y), \tag{2.63}$$

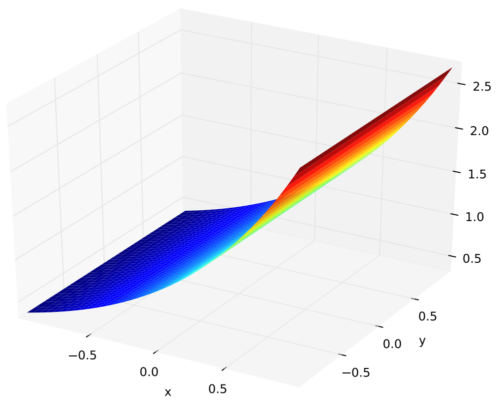
which leads to the solution

$$\begin{cases} x(\vec{\xi}) = \xi, \\ y(\vec{\xi}) = \ln(K_1\eta + K_2), \end{cases}$$

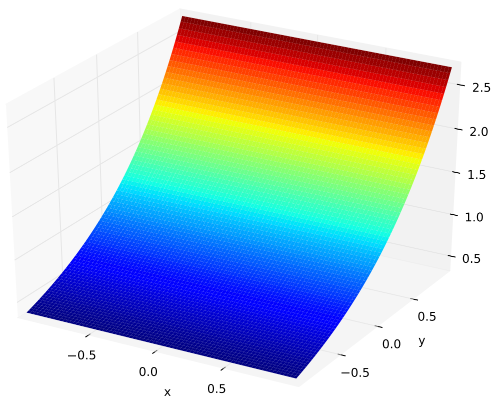
with $K_1$ and $K_2$ defined in equation (2.62). Both monitor functions are plotted in Figure 2.9.

Exact boundary conditions are used for the convergence analysis, so that the accuracy only depends on the interior of the domain. De Boor's algorithm is applied to a fixed refinement of $2^{10}+1 = 1025$ grid points along each boundary.

For the convergence analysis with respect to the space variable, here taken as the width of a cell $h$, we use a Gaussian quadrature with 8 quadrature points in each direction (i.e, a Gaussian quadrature of order 8). We solve to steady-state using a fixed time step-size of $\Delta t = 10^{-4}$. We solve on four meshes with different levels of refinement: $2^i \times 2^i$ grids with $i = 1,2,3,4$. We consider that the solution has converged to the steady-state when the $L_2$-norms of the coordinate transformations $x$ and $y$ have converged to their respective limit[2]. Results in the $L_2$-norm and $L_\infty$-norm are summarized in Table 2.5 and plotted in Figure 2.10.



(a) Monitor function (2.60) for convergence in $x$



(b) Monitor function (2.63) for convergence in $y$

Figure 2.9: Monitor functions (2.60) and (2.63) for convergence study.

The convergence analysis for the solution $y$, obtained with the monitor function (2.63), is carried out in the same manner as for the solution $x$. The convergence results for the $y$-component correspond to the convergence results obtained for the $x$-component with the monitor function (2.60) and are not reproduced to avoid repetition.

### Time-dependent monitor function

In our second test, the monitor function varies in time, inducing the movement of the mesh. We compute the initial mesh with the regularization technique used in the time-independent case, then let the monitor function evolve and compute the dynamic mesh movement.

**Convergence analysis.** We analyze the convergence of the dynamic mesh with respect to the time variable. Since the time-dependent and time-independent problems rely on the same code, the convergence analysis for the spatial variables is not repeated here. We derive an exact solution to a simplified time-dependent problem. In addition to the assumptions made for the spatial convergence analysis, we assume

$$p(\vec{x},t) = \frac{1}{J^2}, \text{ and } \omega(\vec{x},t) = 1.$$

[2]within a relative tolerance of $10^{-16}$.

Table 2.5: Convergence study in space for the solution $x$ with the monitor function (2.60).

| Refinement | $h$ | $L_2$-norm | | $L_\infty$-norm | |
|---|---|---|---|---|---|
| | | Error | Order | Error | Order |
| 1 | 1 | 0.260461 | – | 0.24898 | – |
| 2 | 0.5 | 0.0853078 | 1.61 | 0.112451 | 1.14 |
| 3 | 0.25 | 0.024547 | 1.80 | 0.0426604 | 1.40 |
| 4 | 0.125 | 0.00648759 | 1.92 | 0.014045 | 1.60 |

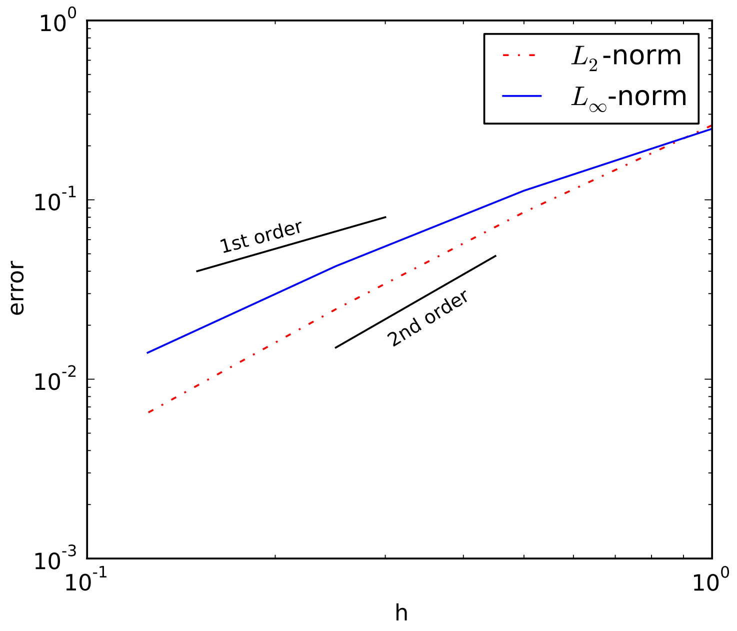


Figure 2.10: Convergence study in space for the solution $x$ with the monitor function (2.60). Results are summarized in Table 2.5.

MMPDE5 (2.45) then becomes

$$\begin{cases} x_t = x_{\xi\xi}, \qquad \forall(\xi,\eta) \in [-1,1]\times[-1,1], \\ x(-1,\eta,t) = -1, \quad x(1,\eta,t) = 1. \end{cases} \tag{2.64}$$

This is the heat equation for the coordinate transformation $x(\vec{\xi},t)$. We first modify equation (2.64) to solve it with homogeneous Dirichlet boundary conditions. Note that $u(\xi,t) = x(\xi,t) - \xi$ solves the PDE

$$\begin{cases} u_t = u_{\xi\xi}, \qquad \forall(\xi,\eta) \in [-1,1]\times[-1,1], \\ u(-1,\eta,t) = 0, \quad u(1,\eta,t) = 0, \end{cases} \tag{2.65}$$

and we can use the method of separation of variables to find $u(\xi,t) = \phi(t)\psi(\xi)$. Equation (2.65) becomes

$$\frac{\phi_t(t)}{\phi(t)} = \frac{\psi''(\xi)}{\psi(\xi)}.$$

Because each side of this equality only depends on one variable,

$$\frac{\phi_t}{\phi} = -c = \frac{\psi''}{\psi}, \quad c \in \mathbb{R}.$$

Thus,

$$\phi(t) = \phi(0)e^{-ct},$$

and

$$x(\xi,t) = \phi(0)e^{-ct}\psi(\xi) + \xi.$$

There are an infinite number of constants $c_n$ and modes $\psi_n$ satisfying

$$\begin{cases} \psi_n'' + c_n\psi_n = 0, \\ \psi_n(-1) = \psi_n(1) = 0. \end{cases}$$

The solution $\psi$ will be a sum of those modes, with the number of terms and the factors for each term depending on the initial conditions. To keep our solution simple, we choose the initial condition corresponding to the first mode. It is given by

$$\psi(\xi) = \cos\left(\frac{\pi}{2}\xi\right),$$

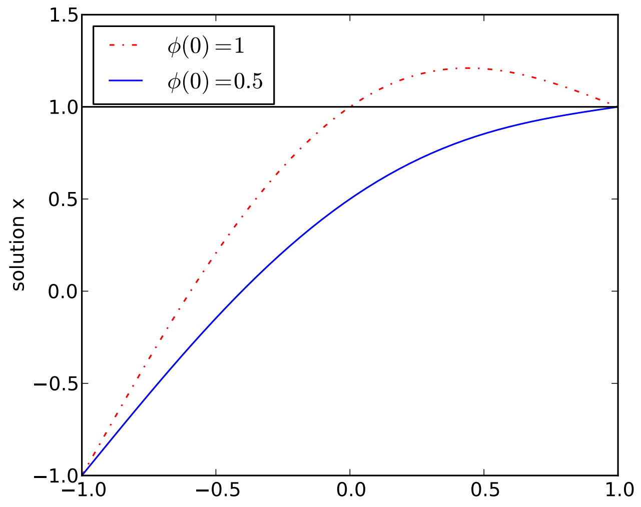


Figure 2.11: Examples of solution $x(\xi,t)$ (2.66) with $\phi(0) = 0.5,\ 1$. For large values of $\phi(0)$, the solution $x$ is not a valid transformation, as it moves points out of the domain (above line of equation $x = 1$).

which corresponds to the constant

$$c = \frac{\pi^2}{4}.$$

This leads to the exact solution

$$x(\xi,t) = \phi(0)\exp\left(-\frac{\pi^2}{4}t\right)\cos\left(\frac{\pi}{2}\xi\right) + \xi. \tag{2.66}$$

We need to be a bit cautious about the choice of $\phi(0)$, because values too large cause points to leave the domain and $x$ is not a valid transformation (see Figure 2.11). Based on these observations, we choose a value of $\phi(0) = 0.5$, which leads to the solution

$$x(\xi,t) = \frac{1}{2}\exp\left(-\frac{\pi^2}{4}t\right)\cos\left(\frac{\pi}{2}\xi\right) + \xi. \tag{2.67}$$

The initial condition is given by the exact solution evaluated at $t = 0$. Since we use a constant monitor function, the MMPDE will produce a uniform mesh in the limit. Indeed, the exact solution (2.67) satisfies

$$\lim_{t\to\infty} x(\xi,t) = \xi.$$

We run the convergence analysis on a $64 \times 64$ grid, and integrate with a Gaussian quadrature of order 6. We compare accuracy at a final time of 1. Results in the $L_2$-norm and $L_\infty$-norm are summarized in Table 2.6 and plotted in Figure 2.12.

**Case study.**

**Lateral travelling.** We first look at a monitor function with peak values concentrated along a vertical line travelling laterally,

$$\omega(\vec{x},t) = 10 \exp\left[-50(x+1-0.001t)^2\right] + 1. \tag{2.68}$$

The monitor function (2.68) at $t = 700$ is plotted in Figure 2.13.

We solve this problem on a $16 \times 16$ grid with a Gaussian quadrature of order 6 and a time step-size of $\Delta t = 0.1$. Boundary conditions are computed with de Boor's algorithm using $2^{10} + 1 = 1025$ grid points along boundaries $\xi = \pm 1$ and using $2^4 + 1 = 17$ grid points along boundaries $\eta = \pm 1$. Figure 2.14 shows the results. The mesh concentrates along vertical lines that move from the left to the right of the domain and the position of the vertical lines coincide with the maximum concentration of the monitor function (2.68).

**Concentric movement.** In our second example, we look at a monitor function with peak values concentrated at a fixed radius from the origin. This radius shrinks in time from 1 to 0, viz.,

$$\omega(\vec{x},t) = 10 \exp\left[-50\left|x^2+y^2-(1-0.001t)^2\right|\right] + 1. \tag{2.69}$$

Four sections of the monitor function (2.69) at $t = 300$ are given in Figure 2.15.

We solve problem (2.69) on a $32 \times 32$ grid with a Gaussian quadrature of order 6 and a time step-size $\Delta t = 0.01$. Boundary conditions are computed with de Boor's algorithm using $2^{12} + 1 = 4097$ grid points along each boundary. Figure 2.16 shows the results: the mesh concentrates around circular shapes that move toward the center of the domain, and the radii of those shapes coincide with the maximum concentration of the monitor function (2.69).

Table 2.6: Convergence analysis in time for the solution $x$ given by Equation (2.67).

| Refinement | $\Delta t$ | $L_2$-norm | | $L_\infty$-norm | |
|---|---|---|---|---|---|
| | | Error | Order | Error | Order |
| 1 | 1 | 0.0754374 | – | 0.0709011 | – |
| 2 | 0.1 | 0.00654177 | 1.06 | 0.00632705 | 1.05 |
| 3 | 0.01 | 0.000595232 | 1.04 | 0.000585637 | 1.03 |
| 4 | 0.001 | 4.16668e-05 | 1.15 | 5.02903e-05 | 1.07 |

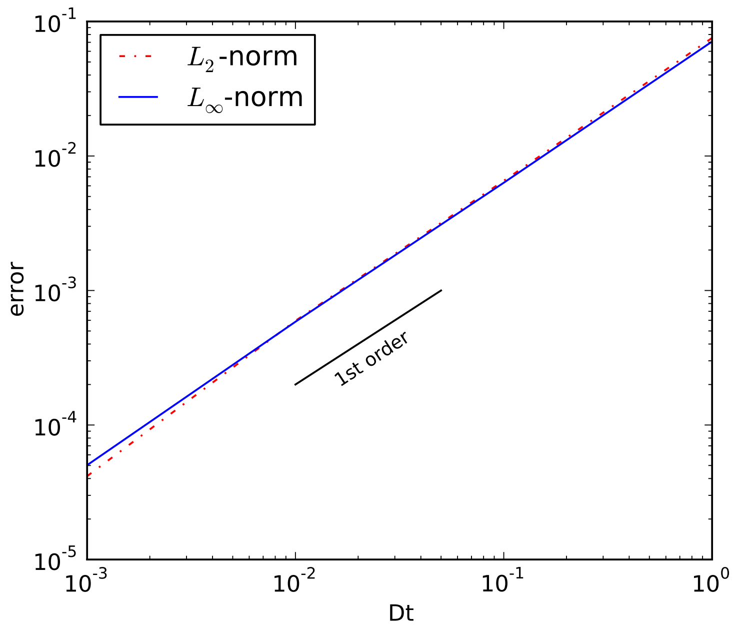


Figure 2.12: Convergence analysis in time for the solution $x$ given by Equation (2.67). Results are summarized in Table 2.6.

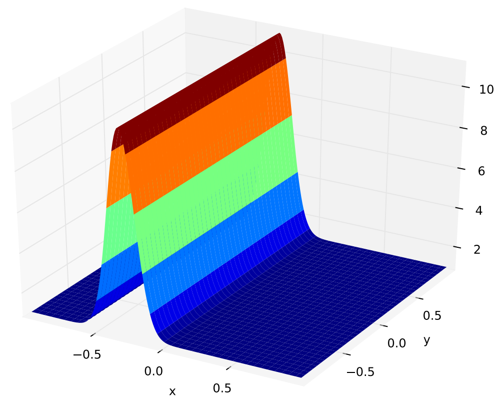
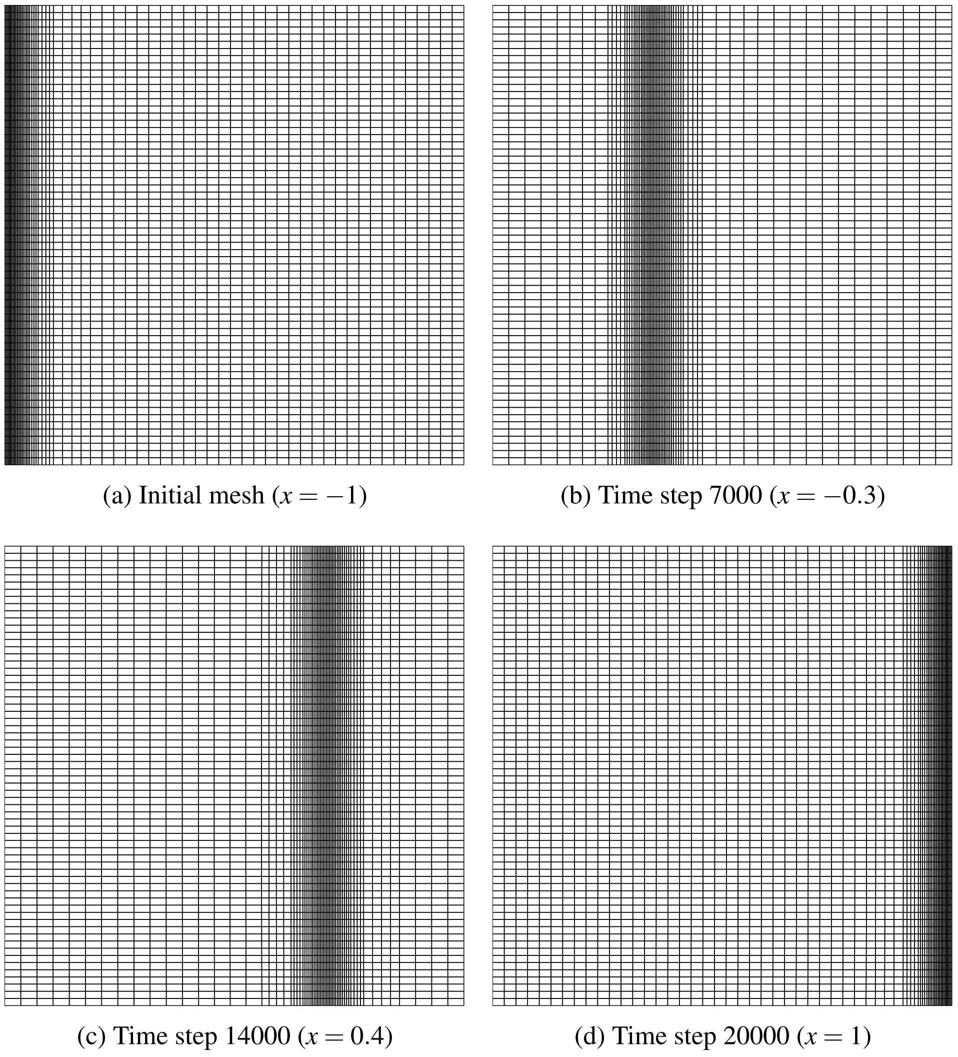


Figure 2.13: Monitor function (2.68) when concentrated at $x = -0.3$ (time step 700).

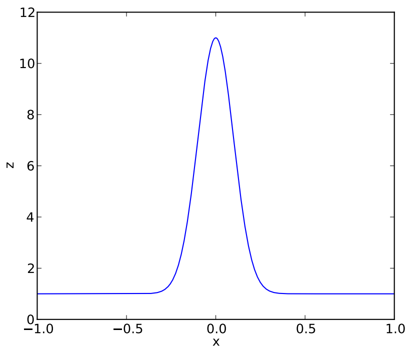

(a) Initial mesh ($x = -1$)

(b) Time step 7000 ($x = -0.3$)

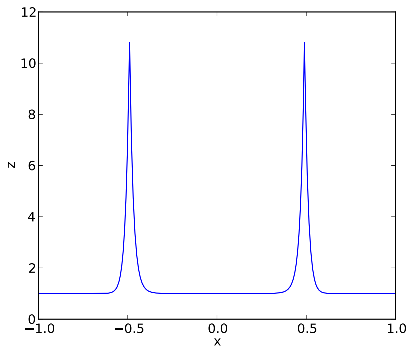
(c) Time step 14000 ($x = 0.4$)

(d) Time step 20000 ($x = 1$)

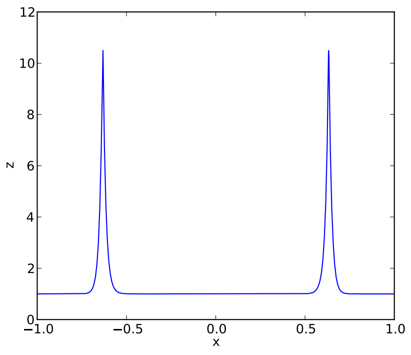
Figure 2.14: Mesh movement induced by the monitor function (2.68). The highest density of the mesh coincides with the maximum concentration of the monitor function at that time step; the position of the maximum concentration is indicated in parenthesis.



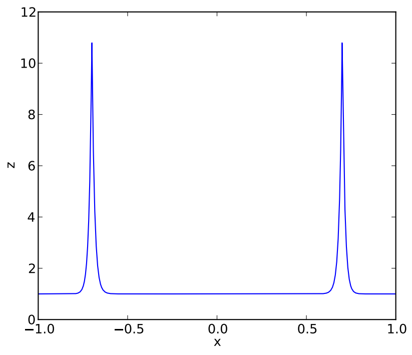
(a) at $y = 0.7$



(b) at $y = 0.5$

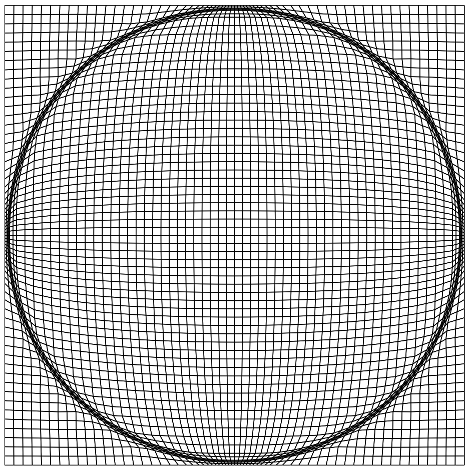


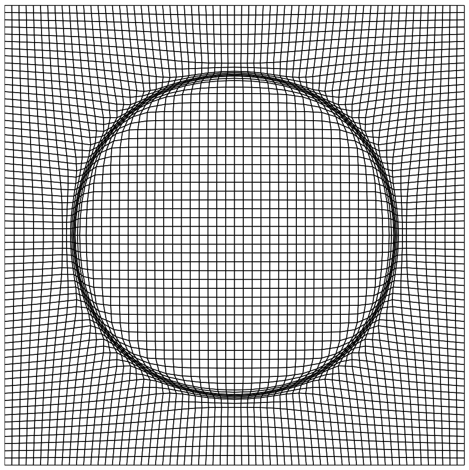
(c) at $y = 0.3$

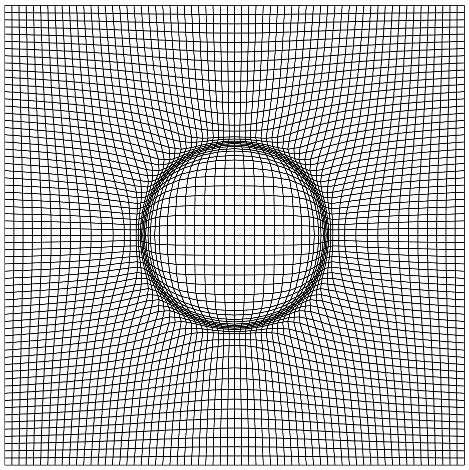


(d) at $y = 0$

Figure 2.15: Sections of monitor function (2.69) at $t = 300$.


(a) Initial mesh ($R = 1$)


(b) Time step 30000 ($R = 0.7$)


(c) Time step 60000 ($R = 0.4$)

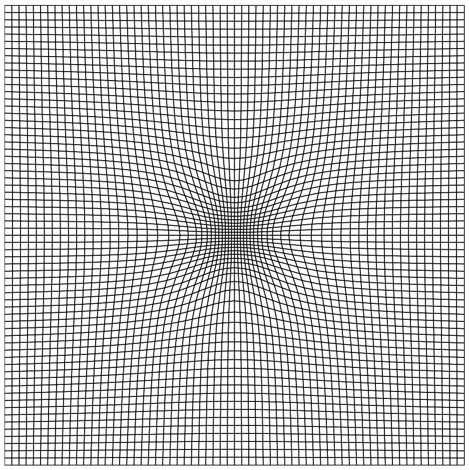

(d) Time step 100000 ($R = 0$)

Figure 2.16: Mesh movement induced by the monitor function (2.69). The highest density of the mesh is located at a fixed radius from the origin, which coincides with the peak values of the monitor function; that radius is indicated in parenthesis.

# Chapter 3

# Moving Meshes on Surfaces

## 3.1 Partial Differential Equations on Surfaces

### 3.1.1 Theory of PDEs on surfaces

PDEs on surfaces are defined along a manifold $S$ of dimension $d$ embedded in a space of higher dimension $t$. All applications in this work deal with the case $d = 2$ and $t = 3$, but more general problems could be addressed, like for instance the Klein bottle, a well-known example of two dimensional surface in a four dimensional space.

#### Surface operators

Traditional differential operators (gradient, divergence and laplacian) can be defined as acting along a surface $S$.
The most intuitive definition for the surface gradient at a point $p$ of the surface $S$ is the projection of the spatial gradient onto the tangent plane at $p$ [Küh06]; it is described mathematically by

$$\nabla_S u = \nabla u - \hat{\mathbf{n}}(\hat{\mathbf{n}} \cdot \nabla u),$$

with $\hat{\mathbf{n}}$ the normal to the surface $S$ at the point considered. The divergence operator can be defined on a Riemannian manifold [Wik]. It is, however, very technical and necessitates the use of a specific type of affine connection named the Levi-Civita connection. The surface

Laplacian is defined, similarly to the general Laplacian, as the surface divergence of the surface gradient; it is more commonly referred to as the Laplace-Beltrami operator.

On a closed manifold, it is possible to define explicitly these surface operators. Thus, if we call $(g_{ij})$ the metric tensor for a closed surface $S$ and a given basis, $(g^{ij})$ its inverse and $G$ its determinant, we have the following definitions [Gao11]:

$$\begin{aligned}
\nabla_s f &= \sum_{i,j=1}^{d} g^{ij} \frac{\partial f}{\partial x^i} \partial_{x^j}, \\
\text{div}_s \vec{v} &= \text{div}_s \sum_{i=1}^{d} v^i \partial_{x^i} = \frac{1}{\sqrt{G}} \sum_{i=1}^{d} \frac{\partial}{\partial x^i} (\sqrt{G} v^i), \qquad (3.1)\\
\Delta_s f &= \text{div}_s (\nabla_s f) = \frac{1}{\sqrt{G}} \sum_{i=1}^{d} \frac{\partial}{\partial x^i} (\sqrt{G} \sum_{j=1}^{d} g^{ij} \frac{\partial f}{\partial x^i}).
\end{aligned}$$

Intuitively, these concepts could be extended to general surfaces. However, no references were found in the general case. Since the existence of a divergence theorem is crucial for the derivation of MMPDEs, we will restrict ourselves to the case of closed surfaces, as described in [Gao11].

### Divergence theorem on a closed surface

The divergence theorem can be extended to the case of a closed manifold [Gao11]. For any function $f, g : S \to \mathbb{R}$ and vector field $\vec{v}$ defined on the surface, we have

$$\int_S (\text{div}_s \vec{v}) f \, dS = - \int_S \vec{v} \cdot \nabla_s f \, dS, \qquad (3.2)$$

$$\int_S (\Delta_s f) g \, dS = - \int_S \nabla_s f \cdot \nabla_s g \, dS. \qquad (3.3)$$

## 3.1.2 Numerical methods for PDEs on surfaces

### Parameterization

We distinguish two types of representation, (1) for surfaces that accept a parametric representation and (2) for triangulated surfaces.

**Definition 3.1.1** (Parametrized Surface [MT03]). A parameterization of a surface is a function $\Phi : D \subset \mathbb{R}^2 \to \mathbb{R}^3$, where $D$ is some domain in $\mathbb{R}^2$. The surface $S$ corresponding to the function $\Phi$ is its image: $S = \Phi(D)$. We can write

$$\Phi(u,v) = (x(u,v), y(u,v), z(u,v)) \, .$$

When the surface possesses a parameterization (definition 3.1.1), it is sometimes possible to express the surface differential operators in terms of the parametric coordinates. This approach has two major drawbacks. Parameterizations will most likely introduce distortion either in angles or in some subregion [FH05]. Furthermore, simple PDEs on a surface, like surface diffusion, become much more complicated after the introduction of a parameterization [RM08].

Another approach is to solve the PDE on a triangulated surface. Each triangulated cell $K$ is referenced by a mapping $F_K$ from a reference cell. This strategy fits naturally in the context of a finite-element formulation since we already use a mapping from a reference cell to assemble the matrix system. However, this technique leads, in general, to some difficulties with the discretization of the differential operators and the computation of surface curvatures.

### Closest-point method

The closest-point method (CPM) is a simple embedding method for directly solving PDEs on surfaces [RM08].

**Definition 3.1.2** (Embedding methods [Mac08])**.** Embedding methods are techniques for numerically solving PDEs or other processes on surfaces by computing on points in the embedding space $\mathbb{R}^d$ instead of the surface itself.

Embedding methods require a representation of the surface in the embedding space. A central element of CPM is the definition of a closest-point representation. Every point in the embedding space is defined by its distance to the surface. This simple choice has the additional benefit of allowing a natural extension of the phenomena described on the surface to the embedding space; this is called a closest-point extension.

**Definition 3.1.3** (Closest-point extension [Mac08])**.** Let $S$ be a surface embedded in $\mathbb{R}^d$ and let $\tilde{u} : S \to \mathbb{R}$ be a scalar function defined over $S$. Then the closest point extension of $\tilde{u}$ is a function $u : \mathbb{R}^d \to \mathbb{R}$ with $u(\vec{x}) = \tilde{u}(\mathrm{cp}(\vec{x}))$.

In order to state clearly the main results of CPM, we need to make the following observations [RM08]:

*Remark* 3.1.1. Suppose $u$ is any function defined on $\mathbb{R}^3$ that is constant along directions normal to the surface. Then at the surface,

$$\nabla u = \nabla_s u.$$

*Remark* 3.1.2. For any vector field $v \in \mathbb{R}^3$ that is tangent at $S$ and also tangent at all surfaces displaced by a fixed distances from $S$ (ie, all surfaces defined as level sets of the distance function to $S$),

$$\nabla \cdot v = \nabla_s \cdot v$$

at the surfaces.

If $u$ is a function defined on the surface, its closest-point extension is constant along directions tangent to the surface. Applying the result of remark 3.1.1, we obtain

$$\nabla u(\mathrm{cp}(\vec{x})) = \nabla_s u(\vec{x}). \tag{3.4}$$

Since $u(\mathrm{cp})$ is constant along directions tangent to the surface, its gradient will be tangent to all surfaces displaced by a fixed distance from $S$. We use the result of remark 3.1.2 to derive

$$\nabla \cdot (\nabla u(\mathrm{cp}(\vec{x}))) = \nabla_s \cdot (\nabla_s u(\vec{x})), \tag{3.5}$$

which corresponds to the Laplace-Beltrami operator.
The conversion formulas (3.4) and (3.5) are used to transform a PDE defined on a surface $S$ into a PDE defined in the embedding space, through the closest-point extension. Let's illustrate that idea with the PDE

$$\begin{cases} u_t(t,\vec{x}) = F\left(t,\vec{x},u(t,\vec{x}),\nabla_s u(t,\vec{x}),\Delta_s u(t,\vec{x})\right), \\ u_t(0,\vec{x}) = u_0(\vec{x}) \end{cases}$$

defined on a surface $S$, where $F$ is a general differential operator. After defining a closest-point extension for the surface $S$ and the corresponding embedding space, we can equivalently solve the Cartesian analog to the surface PDE

$$\begin{cases} u_t(t,\vec{x}) = F\left(t,\mathrm{cp}(\vec{x}),u(t,\mathrm{cp}(\vec{x}),\nabla u(t,\mathrm{cp}(\vec{x})),\Delta u(t,\mathrm{cp}(\vec{x}))\right), \\ u_t(0,\vec{x}) = u_0(\mathrm{cp}\vec{x})). \end{cases}$$

On the surface, the solutions to both PDEs are equal.

deal.II is dedicated specifically to the numerical solution of finite-element formulations and restricts access to some data of the model. For instance, the geometry of the mesh must be retrieved which introduces some challenges in the implementation of the CPM. Consequently this part of the project was left for future development.

## 3.2 MMPDEs on Surfaces

The derivation of an MMPDE on a surface requires the use of the divergence theorem. Since we only defined a divergence theorem on closed surfaces in Section 3.1.1, we restrict ourselves in this section to that specific case as well.

### 3.2.1 Equation of the MMPDE

PDEs on surfaces are most naturally derived in terms of the surface differential operators. Given the formulation of MMPDE5 (2.45) obtained in Chapter 2, we would not be able to apply that equation for a surface $S$. However, the original functional (2.33), from which that MMPDE is derived, is much more amenable for that purpose. On a two-dimensional surface embedded in $\mathbb{R}^3$, this functional is defined by

$$I[\vec{\xi}] = \int_\Omega \frac{1}{2\omega(\vec{x})} \sum_{i=1}^{3} \|\nabla \xi_i\|^2 \, d\vec{x}.$$

The corresponding surface functional is then given by

$$I_s[\vec{\xi}] = \int_S \frac{1}{2\omega(\vec{x})} \sum_{i=1}^{3} \|\nabla_s \xi_i\|^2 \, d\vec{S}. \tag{3.6}$$

The coordinate transformation solution $\vec{\xi}$ is a minimizer of the functional (3.6), satisfying

$$\delta I_s[\vec{\xi}] = 0.$$

Let $\vec{a}_s^i$ be the $i^{\text{th}}$ surface contravariant vector, defined by

$$\vec{a}_s^i = \nabla_s \xi_i,$$

and let

$$F(\vec{a}_s^1, \vec{a}_s^2, \vec{a}_s^3, \vec{x}) = \frac{1}{2\omega(\vec{x})} \sum_{i=1}^{3} \|\nabla_s \xi_i\|^2.$$

The first variation of functional $I_s[\vec{\xi}]$ is then

$$\delta I_s[\vec{\xi}] = \int_S \left[ \sum_{i=1}^{3} \frac{\partial F}{\partial \vec{a}_s^i} \cdot \delta \vec{a}_s^i \right] d\vec{S},$$

with the partial derivatives of the integrand $F$ given by

$$\frac{\partial F}{\partial \vec{a}_s^i} = \frac{1}{\omega(\vec{x})} \nabla_s \xi_i.$$

The first variation of functional $I_s[\vec{\xi}]$ becomes

$$\delta I_s[\vec{\xi}] = \int_S \frac{1}{\omega(\vec{x})} \left[ \sum_{i=1}^{3} \nabla_s \xi_i \cdot \nabla_s \delta \xi_i \right] d\vec{S}.$$

Using the divergence theorem (3.3), we re-write the first variation as

$$I_s[\vec{\xi}] = -\int_S \sum_{i=1}^{3} \nabla_s \cdot \left( \frac{1}{\omega(\vec{x})} \nabla_s \xi_i \right) \cdot \delta \xi_i \, d\vec{S}.$$

This being true for all admissible variations $\delta \xi_i$, our coordinate transformation solution must satisfy the equation

$$-\nabla_s \cdot \left( \frac{1}{\omega(\vec{x})} \nabla_s \xi_i \right) = 0, \quad i = 1,2,3. \tag{3.7}$$

### 3.2.2 Gradient flow equation

We do not solve equation (3.7) directly, but instead evolve the corresponding gradient flow equation to a steady state solution. The functional derivative is given by

$$\frac{\delta I_s}{\delta \xi_i} = -\nabla_s \cdot \left( \frac{1}{\omega(\vec{x})} \nabla_s \xi_i \right), \quad i = 1,2,3,$$

which leads to the gradient flow equation

$$\begin{aligned} & \frac{\partial \xi_i}{\partial t} = -\frac{1}{\tau p(\vec{x},t)} \frac{\delta I_s}{\delta \xi_i}, \quad i = 1,2,3 \\ \Leftrightarrow \quad & \frac{\partial \xi_i}{\partial t} = \frac{1}{\tau p(\vec{x},t)} \nabla_s \cdot \left( \frac{1}{\omega(\vec{x})} \nabla_s \xi_i \right), \quad i = 1,2,3. \end{aligned} \tag{3.8}$$

Equation (3.8) is formulated in terms of the inverse coordinate transformation $\vec{\xi} = \vec{\xi}(\vec{x})$. Unfortunately, it is not obvious that one can interchange the coordinates without losing the surface differential operators. For that reason, we solve equation (3.8) for $\vec{\xi}$ and then interpolate the solution from the computational to the physical grid.
It is not clear what would be a good candidate for the balancing function $p(\vec{x},t)$; one could try to use the same definition as in Section 2.3.1.

### 3.2.3 Finite-element solution

Let $v$ be a function defined on the closed surface $S$. We multiply each side of equation (3.8) by $v$, integrate over the whole surface, and then apply the divergence theorem to obtain

$$\begin{aligned} & \int_s \tau p(\vec{x},t) \frac{\partial \xi_i}{\partial t} v \, d\vec{S} = \int_s \nabla_s \cdot \left( \frac{1}{\omega(\vec{x})} \nabla_s \xi_i \right) v \, d\vec{S}, \\ \Leftrightarrow \; & \int_s \tau p(\vec{x},t) \frac{\partial \xi_i}{\partial t} v \, d\vec{S} = -\int_s \frac{1}{\omega(\vec{x})} \nabla_s \xi_i \cdot \nabla_s v \, d\vec{S}. \end{aligned} \tag{3.9}$$

Since the equations for each coordinate $\xi_i$ are identical and independent, we describe the formulation for only one coordinate $\xi$ without loss of generality. Let $S_h$ be a mesh defined on the surface $S$, and let $V_h$ be the space of piecewise linear functions on $S_h$ that are continuous over the whole surface, with a basis $\left\{\varphi_j\right\}_j$. The problem is now to find an approximation $\xi_h \in V_h$ of the solution $\xi$. Working in $V_h$, the problem becomes to find $v_h \in V_h$ such that

$$\int_{S_h} \tau p(\vec{x},t) \frac{\partial \xi_h}{\partial t} v_h \, d\vec{S} = -\int_{S_h} \frac{1}{\omega(\vec{x})} \nabla_s \xi_h \cdot \nabla_s v_h \, d\vec{S}, \quad \forall v_h \in V_h. \tag{3.10}$$

Equivalently equation (3.10) is satisfied for all vectors of the basis $\left\{\varphi_j\right\}_j$, i.e,

$$\int_{S_h} \tau p(\vec{x},t) \frac{\partial \xi_h}{\partial t} \varphi_i \, d\vec{S} = -\int_{S_h} \frac{1}{\omega(\vec{x})} \nabla_s \xi_h \cdot \nabla_s \varphi_i \, d\vec{S}, \quad \forall i = 1,2,\ldots$$

Using a time discretization $t_n = t_0 + n\Delta t$, let $\xi_h^n = \xi_h(t_n)$. Approximating the time-derivative of the solution with the backward Euler scheme gives

$$\int_{S_h} \tau p(\vec{x},t) \frac{\xi_h^{n+1} - \xi_h^n}{\Delta t} \varphi_i \, d\vec{S} = -\int_{S_h} \frac{1}{\omega(\vec{x})} \nabla_s \xi_h^{n+1} \cdot \nabla_s \varphi_i \, d\vec{S}, \quad \forall i = 1,2,\dots$$

Because $\xi_h^n$ has a unique decomposition

$$\xi_h^n = \sum_j C_j^n \varphi_j,$$

we have

$$\sum_j \left( C_j^{n+1} - C_j^n \right) \int_{S_h} \tau p(\vec{x},t) \varphi_j \varphi_i \, d\vec{S} = -\Delta t \sum_j C_j^{n+1} \int_{S_h} \frac{1}{\omega(\vec{x})} \nabla_s \varphi_j \cdot \nabla_s \varphi_i \, d\vec{S}$$

$$\Leftrightarrow \sum_j C_j^{n+1} \left( \int_{S_h} \tau p(\vec{x},t) \varphi_j \varphi_i \, d\vec{S} + \Delta t \int_{S_h} \frac{1}{\omega(\vec{x})} \nabla_s \varphi_j \cdot \nabla_s \varphi_i \, d\vec{S} \right) = \sum_j C_j^n \int_{S_h} \tau p(\vec{x},t) \varphi_j \varphi_i \, d\vec{S}, \quad \forall i = 1,2,\dots \tag{3.11}$$

Using the matrix notation

$$\{C\}^n = \begin{pmatrix} C_1^n \\ C_2^n \\ \vdots \\ \vdots \end{pmatrix}, \; [K_1] = \left( \int_{S_h} \tau p(\vec{x},t) \varphi_j \varphi_i \, d\vec{S} \right)_{ij}, \; [K_2] = \left( \int_{S_h} \frac{1}{\omega(\vec{x})} \nabla_s \varphi_j \cdot \nabla_s \varphi_i \, d\vec{S} \right)_{ij},$$

equation (3.11) becomes

$$\left( [K_1] + \Delta t [K_2] \right) \{C\}^{n+1} = [K_1] \{C\}^n . \tag{3.12}$$

We notice that matrices $[K_1]$ and $[K_2]$ are time-independent, since $p$ and $\omega$ are functions of the integration variable directly.

This completes the theoretical formulation of the method. As mentioned before, we must now perform an interpolation after solving (3.12) in order to obtain the direct coordinate transformation. However, a simpler method is possible. It is presented below.

## 3.3 A simple method for surfaces with a one-to-one parameterization

### 3.3.1 Principle of the method

When a surface parameterization $\Phi$ (see Definition 3.1.1) is also a one-to-one mapping from $D \subset \mathbb{R}^2$ onto $S = \Phi(D)$, we call $\Phi$ a one-to-one parameterization. We can then describe $S$ equivalently with parameters on the surface $\Phi(D)$, or on the plane $D$. Hence, the movement of the mesh that takes place on the surface $S$ has a one-to-one correspondence with the movement that takes place on $D$, a subset of the plane. The idea is to solve the MMPDE on that subset of the plane; because the monitor function controls the density of the mesh, it must be defined and evaluated on the surface $S$.
We can define an exact mathematical formulation corresponding to this idea. Let $\vec{u} = \vec{u}(\vec{\xi})$ be the coordinates of the moved mesh in the plane $\mathbb{R}^2$, i.e,

$$\vec{u}(\vec{\xi}) = \begin{pmatrix} u(\xi,\eta) \\ v(\xi,\eta) \end{pmatrix}.$$

Therefore, the movement of the mesh on the surface $S$ will be governed by

$$\begin{aligned}\dot{\vec{u}} =& \frac{1}{J^2(\omega\circ\Phi)^2\,\tau p(\vec{u},t)}\left\{\left(u_\eta^2+v_\eta^2\right)\frac{\partial}{\partial\xi}\left((\omega\circ\Phi)\frac{\partial\vec{u}}{\partial\xi}\right)\right.\\ &-\left(u_\xi u_\eta+v_\xi v_\eta\right)\frac{\partial}{\partial\eta}\left((\omega\circ\Phi)\frac{\partial\vec{u}}{\partial\xi}\right)-\left(u_\xi u_\eta+v_\xi v_\eta\right)\frac{\partial}{\partial\xi}\left((\omega\circ\Phi)\frac{\partial\vec{u}}{\partial\eta}\right)\\ &\left.+\left(u_\xi^2+v_\xi^2\right)\frac{\partial}{\partial\eta}\left((\omega\circ\Phi)\frac{\partial\vec{u}}{\partial\eta}\right)\right\},\end{aligned} \tag{3.13}$$

where we use the notation

$$\omega\circ\Phi = (\omega\circ\Phi)(u,v) = \omega\left(\Phi(u,v)\right) = \omega\left(x(u,v),y(u,v),z(u,v)\right).$$

The computation of the solution to equation (3.13) is identical to what was presented in Section 2.2.

### 3.3.2 Choice of the parameterization $\Phi$

The parameterization $\Phi$ described in Section 3.3.1 is not unique. Ideally, our parameterization should turn a uniform mesh on $D$ into a uniform mesh on $S$. This translates into

imposing that our mapping preserves the relative area of each cell, in both $D$ and $S = \Phi(D)$. Looking back at the method described in Section 3.3.1, the monitor function will intervene in the finite-element formulation through its integral over the plane domain. Preserving the ratio of the areas guarantees that the planar equation will fairly represent the zones of high concentration of the monitor function.
Consider a simple example with a one-dimensional graph in $\mathbb{R}^2$ defined by the equation $y = f(x)$ on $[0,1]$. Given the uniform mesh $x_i = i/4$, for $i = 0,\dots,4$, we look for the coordinate transformation $\Phi$ such that the arc-length along the points of x-coordinates $\Phi(x_i)$ are all equal. Mathematically, this gives

$$\int_{x_0}^{\Phi(x_1)} \sqrt{1+[f'(x)]^2}dx = \ldots = \int_{\Phi(x_3)}^{x_4} \sqrt{1+[f'(x)]^2}dx.$$

We recognize the equidistribution formulation defined in Section 2.1.1. Indeed, we are looking for an equidistributing coordinate transformation $\Phi$ corresponding to the monitor function

$$\rho(x) = \sqrt{1+[f'(x)]^2}.$$

We can look for an exact equation or, as was done for this example, compute an approximation with de Boor's algorithm. The result for $f(x) = 2 - x^2$ is given in Figure 3.1.

In higher dimensions, the choice of a functional and a monitor function is not clear and should be further investigated. However, a simple method would be to impose equidistribution only and use functional (2.28). One would thus replace the factor $\rho$ in equation (2.28) with the function $A(S)$ evaluating the area of the surface $S$ (see Definition 3.3.1).

**Definition 3.3.1** (Area of a Parameterized Surface [MT03])**.** We define a parameterized surface $S$ to be the image of a function $\Phi : D \subset \mathbb{R}^2 \to \mathbb{R}^3$, written

$$\Phi(u,v) = (x(u,v), y(u,v), z(u,v)).$$

We define the surface area $A(S)$ of a parameterized surface by

$$A(S) = \iint_D \left\|\vec{T}_u \times \vec{T}_v\right\| dudv,$$

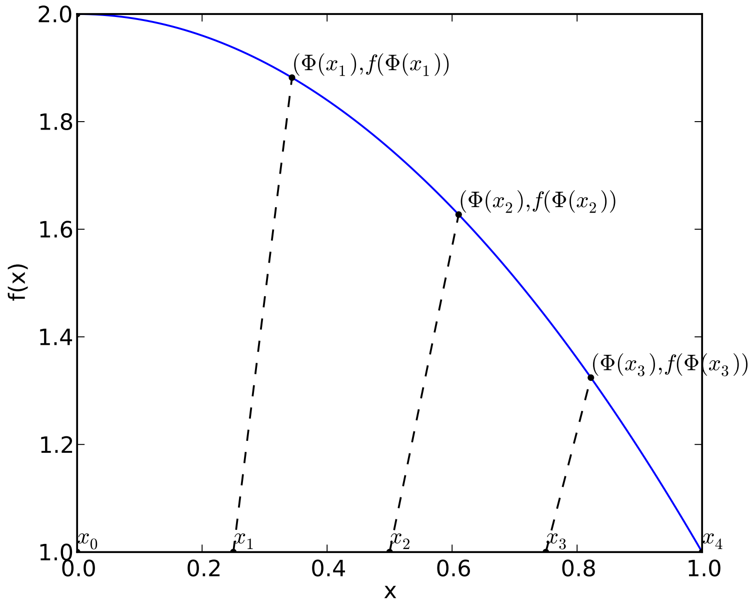


Figure 3.1: Mapping preserving a constant area ratio for mesh $\{x_i\}$ onto the surface of equation $y = 2 - x^2$.

with

$$\vec{T}_u = \frac{\partial x}{\partial u}(u,v)\vec{i} + \frac{\partial y}{\partial u}(u,v)\vec{j} + \frac{\partial z}{\partial u}(u,v)\vec{k},$$
$$\vec{T}_v = \frac{\partial x}{\partial v}(u,v)\vec{i} + \frac{\partial y}{\partial v}(u,v)\vec{j} + \frac{\partial z}{\partial v}(u,v)\vec{k}.$$

If the surface is a graph, that is, if the parameterization $\Phi$ can be written as

$$\Phi(u,v) = (u, v, z(u,v)),$$

we obtain the simpler formula

$$A(S) = \iint_D \sqrt{1 + \left(\frac{\partial z}{\partial u}\right)^2 + \left(\frac{\partial z}{\partial v}\right)^2}\, dudv.$$

## 3.4 Computation of Moving Meshes on Surfaces

We first present how the methodology developed in Section 3.3 was implemented, and then show a few characteristic examples.

### 3.4.1 Implementation of the parametric MMPDE

The implementation of the parametric MMPDE (3.13) in deal.II is built on top of the implementation previously developed for the two-dimensional MMPDE (2.50). The main task consists of turning the three-dimensional monitor function $\omega$ into a two-dimensional equivalent $\omega \circ \Phi$, seen through the surface $S = \Phi(D)$. This is done with a C++-class interfacing between the two-dimensional code from Section 2, the three-dimensional monitor function, and the equation of the surface $S$.

### 3.4.2 Computation of the parameterization $\Phi$

In this section, we detail the computation of the parameterization $\Phi$ defined in Section 3.3.2. In this thesis, we did not compute any two-dimensional parameterizations, even though we did compute parameterizations for two-dimensional surfaces. Using the symmetry of the surface, we can reduce the computation of certain parameterizations to a one-dimensional equidistribution problem.

Consider a two-dimensional graph defined by the equation

$$z(x,y) = f(x), \quad \forall (x,y) \in [-1,1] \times [-1,1].$$

Because of the particular symmetry of that surface, the equidistribution determining the parameterization $\Phi$ only acts in the x-direction. With $(\xi,\eta)$ the computational coordinates, we write the equation of the surface $S$ as

$$\begin{cases} x(\xi,\eta) = \Phi(\xi), \\ y(\xi,\eta) = \eta, \\ z(\xi,\eta) = f \circ \Phi(\xi) = f(x). \end{cases}$$

We then solve the one-dimensional equidistribution equation (2.9), i.e,

$$\frac{d}{d\xi}\left(\rho(x)\frac{dx}{d\xi}\right) = 0.$$

As in Section 2.3.2, we could start by solving the equidistribution equation using separation of variables. This leads to the general equation

$$H(x) = K_1\xi + K_2. \tag{3.14}$$

Generally speaking, we cannot solve equation (3.14) to obtain $x(\xi)$. However, practically speaking, we do not need an explicit formulation for $x(\xi)$; we simply need a way to compute $x(\xi)$ for a given value of $\xi$. Thus, for a given $\xi$, we define the function

$$F(x) = H(x) - K_1\xi + K_2 = 0,$$

and apply Newton's method to compute the needed root $x$ of $F$. Thus, using Newton's method we compute a sequence of approximations

$$x_{n+1} = x_n - \frac{F(x_n)}{F'(x_n)}.$$

Under suitable assumptions on $F$, Newton's method converges quadratically to the root $x$ if the initial guess $x_0$ is close enough. This initial value is computed via de Boor's algorithm on a refined grid, followed by a linear interpolation. Thus, the computation of a transformed coordinate $x(\xi)$, for a given $\xi$, is a three step process:

1. Run de Boor's algorithm on a refined grid; e.g, we used 1025 grid points. This rather large number of grid points is justified by the fact that de Boor's algorithm is computationally inexpensive to run and we need an initial guess for Newton's method that is close enough to the actual solution.
2. Apply linear interpolation in between the points obtained in step 1.
3. Use the value obtained in step 2 to start Newton's method.

To illustrate, we now derive the equations needed for a simple surface defined by the equation

$$z(x,y) = 1 - 2x^2, \quad \forall (x,y) \in [-1,1] \times [-1,1].$$

The monitor function $\rho(x)$ becomes

$$\rho(x) = \sqrt{1+16x^2},$$

and we solve the equation

$$\begin{cases} \int \sqrt{1+16x^2}dx = K_1\xi + K_2, \\ x(-1) = -1, \quad x(1) = 1. \end{cases} \tag{3.15}$$

To integrate the left-hand side, we introduce the substitution

$$\begin{aligned} u &= 4x \\ du &= 4dx, \end{aligned}$$

which gives

$$\int \sqrt{1+16x^2}dx = \frac{1}{4}\int \sqrt{1+u^2}du.$$

If $I = \int \sqrt{1+u^2}du$, then integration by parts gives

$$\begin{aligned} I &= u\sqrt{1+u^2} - \int \frac{u^2}{\sqrt{1+u^2}}du, \\ &= u\sqrt{1+u^2} - \int \frac{1+u^2-1}{\sqrt{1+u^2}}du, \\ &= u\sqrt{1+u^2} - I + \int \frac{1}{\sqrt{1+u^2}}du, \\ &= u\sqrt{1+u^2} - I + \operatorname{arcsinh}(u), \end{aligned}$$

so

$$I = \int \sqrt{1+u^2}du = \frac{1}{2}\left[u\sqrt{1+u^2} + \operatorname{arcsinh}(u)\right],$$

and

$$\int \sqrt{1+16x^2}dx = \frac{1}{8}\left[4x\sqrt{1+16x^2} + \operatorname{arcsinh}(4x)\right]. \tag{3.16}$$

Equation (3.15) becomes

$$\begin{cases} 4x\sqrt{1+16x^2} + \operatorname{arcsinh}(4x) = 8K_1\xi + K_2, \\ \quad x(-1) = -1, \quad x(1) = 1. \end{cases}$$

We evaluate integration constants $K_1$ and $K_2$ using the boundary conditions and the fact that arcsinh is an odd function:

$$\begin{aligned} -4\sqrt{17} - \operatorname{arcsinh}(4) &= -8K_1 + K_2, \\ 4\sqrt{17} + \operatorname{arcsinh}(4) &= 8K_1 + K_2. \end{aligned}$$

Consequently $K_2 = 0$, and

$$K_1 = \frac{\sqrt{17}}{2} + \frac{\operatorname{arcsinh}(4)}{8}. \tag{3.17}$$

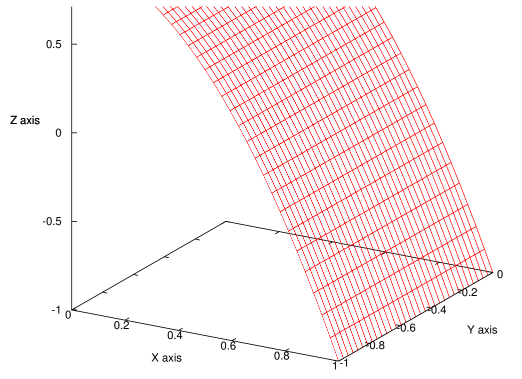
For Newton's method, we use the function $F$ defined for a given value of $\xi$ by

$$F(x) = 4x\sqrt{1+16x^2} + \operatorname{arcsinh}(4x) - 8K_1\xi,$$

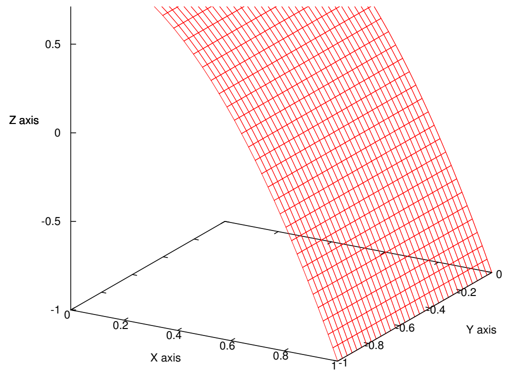
with $K_1$ defined in equation (3.17). From equation (3.16), we immediately obtain the derivative

$$F'(x) = 8\sqrt{1+16x^2}.$$

We compare the aspect of two uniform meshes on the surface $z = 1 - 2x^2$, obtained by two different choices of the parameterization $\Phi$. When using the parameterization $\Phi$ defined in Section 3.3.2, we obtain cells with equal area (Figure 3.2b). When we use the equation of the surface as the parameterization $\Phi$, we obtain larger cells where the surface is the most inclined, i.e for $|x| > 0.5$ (Figure 3.2a).



(a) Using the equation of the surface for Φ



(b) Using the parameterization Φ defined in Section 3.3.2

Figure 3.2: Comparison of the uniform mesh obtained from two choices of the parameterization Φ, for the surface of equation $z(x,y) = 1 - 2x^2$.

### 3.4.3 Computational results

We present the results from several computations for both the time-independent and time-dependent cases. Since both implementations use the solver developed in the planar case, no further convergence analysis is carried out. In the time-independent case, we look at the influence of the surface on the mesh computed for a given monitor function. Because we only focus on the area of the mesh that is the most dense, we do not resort to the parameterization defined in Section 3.3 in this parametric study. This avoids unnecessary work. In the time-dependent case, we provide an example which illustrates how the choice of parameterization affects the mesh.

#### Time-independent monitor function

We look at the influence of the orientation of the surface on the density of the mesh for a monitor function concentrated around the horizontal plane $z = 0$,

$$\omega(\vec{x}) = 1000 \exp\left(-100z^2\right) + 1. \tag{3.18}$$

We choose a $16 \times 16$ grid, a time step-size of $\Delta t = 0.1$ and a Gaussian quadrature of order 5. We compute the adapted mesh for four different surfaces, with the equation of the surface giving the parameterization in each case:

- A linear surface:

$$z(x,y) = -x, \quad \forall (x,y) \in [-1,1] \times [-1,1].$$

- A quadratic surface:

$$z(x,y) = 1 - 2x^2, \quad \forall (x,y) \in [-1,1] \times [-1,1].$$

- A cubic surface:

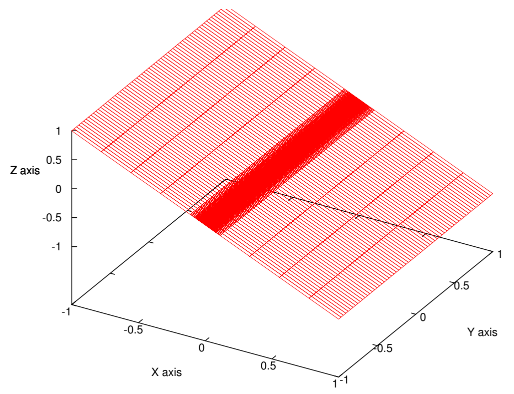
$$z(x,y) = -x^3, \quad \forall (x,y) \in [-1,1] \times [-1,1].$$

- A product surface:

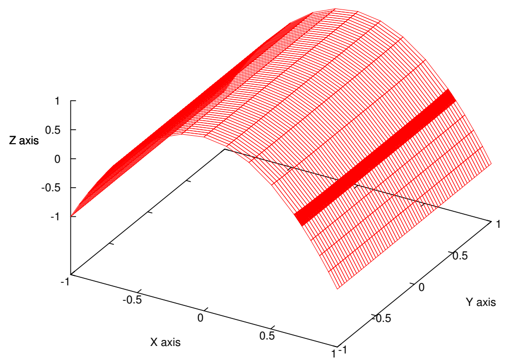
$$z(x,y) = xy, \quad \forall (x,y) \in [-1,1] \times [-1,1].$$

On the four surfaces, the mesh concentrates around the elevation $z = 0$. For the linear, quadratic and cubic surfaces, their intersection with the plane $z = 0$ creates a line. In Figures 3.3a, 3.3b and 3.3c, the density of the mesh increases toward that line, creating a stripe around that line. The intersection of the product surface and the plane $z = 0$ forms two orthogonal lines, which results in the mesh being concentrated around two orthogonal stripes (Figure 3.3d).

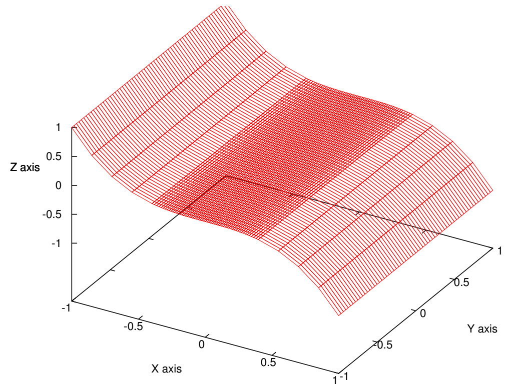
Comparing the linear, quadratic and cubic surfaces, we notice the role of the inclination of the surface on the resulting mesh. The linear surface intersects with the plane $z = 0$ for all $x = 0$ where its derivative $\partial z / \partial x$ is constant and equal to $-1$. The quadratic surface crosses the horizontal plane $z = 0$ for $x = \pm 1/\sqrt{2}$ where its slope along the x-direction is equal to $\pm 2\sqrt{2}$. Lastly, the cubic surface has a horizontal slope where it intersects with the horizontal plane. We observe, in Figures 3.3a, 3.3b and 3.3c, that the width of the stripe of the concentrated mesh is inversely proportional to the slope of its tangent surface. The larger the slope, the steeper the variation of the monitor function (3.18) along the surface, and hence the more concentrated the mesh is.



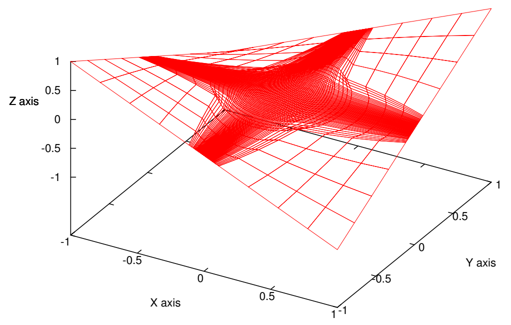
(a) The linear surface $z(x,y) = -x$



(b) The quadratic surface $z(x,y) = 1 - 2x^2$



(c) The cubic surface $z(x,y) = -x^3$



(d) The product surface $z(x,y) = xy$

Figure 3.3: Monitor function (3.18) on different surfaces.

**Time-dependent monitor function**

We outline the importance of the parameterization by solving the same time-varying problem twice, once with $\Phi$ taken as the surface equation and a second time with the parameterization $\Phi$ as defined in Section 3.3.2.

We work with the monitor function

$$\omega(\vec{x},t) = 100\exp\left(-100(z-1.3+0.001)^2\right)+1$$

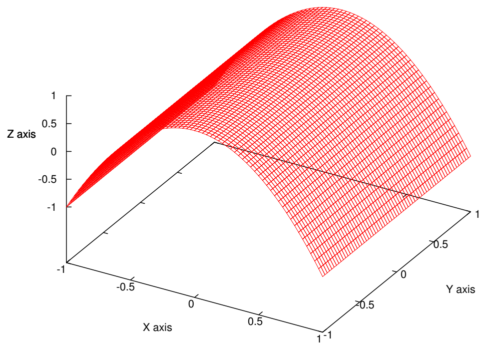
which is exponentially concentrated around a horizontal line "$z=$ constant" that decreases from the elevation $z=1.3$ to $z=-1.2$, at a velocity of 0.001. The surface on which we solve this example is the parabola given by

$$z(x,y) = 1-2x^2, \quad \forall(x,y)\in[-1,1]\times[-1,1]. \tag{3.19}$$

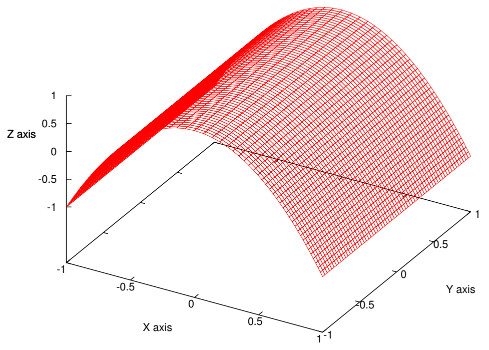
The mesh is a $16\times 16$ grid, the time step-size is 0.01 and the Gaussian quadrature has order 6. The results for both choices of the parameterization are shown in Figures 3.4 and 3.5 at six different time steps.

We observe that the mesh gathers at the top of the surface before splitting into two parts that move downward, i.e, in the negative $z$-direction. It is interesting to notice how the inclination of the surface influences the concentration of the mesh. The monitor function, being concentrated around a horizontal line, imposes steeper variations on the parts of the surface that are the most vertical. When the maximum concentration is at the top of the surface ($z=1.0$), the variations of the monitor function on the surface are more gradual and the stripe of the mesh consequently is wider.

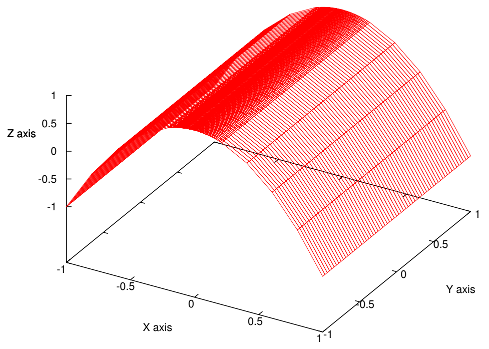
A closer observation of any results indicates differences between both types of parameterizations. Of particular importance is that the parameterization $\Phi$ spreads out the mesh more evenly over the whole surface (see Figures 3.4 and 3.5). Whether this is a desirable feature or not, however, might be controversial. The comparison between various parameterizations is at that stage, and in the absence of a reference computation to compare with, rather subjective. Nevertheless, remaining consistent with our earlier definition of an ideal parameterization, we consider the mesh distribution produced by the parameterization $\Phi$ as our reference solution.



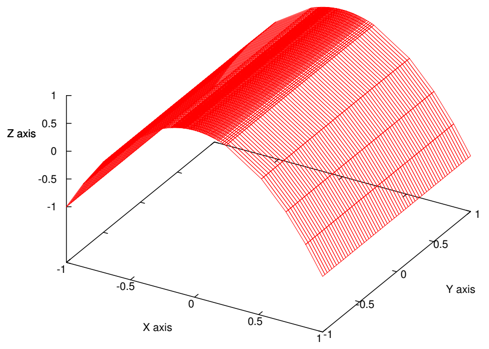
(a) Initial mesh ($z = 1.3$)



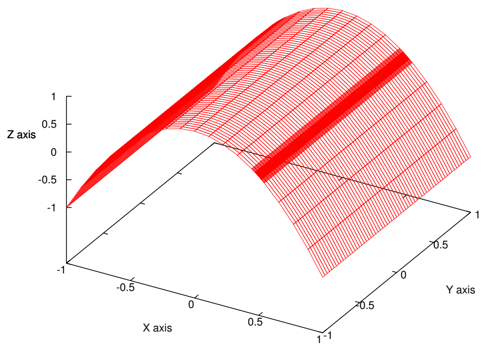
(b) Initial mesh ($z = 1.3$) with $\Phi$



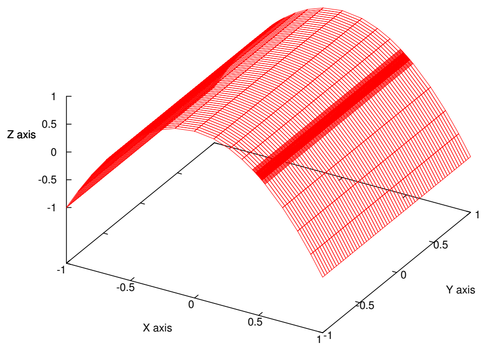
(c) Time step 30000 ($z = 1.0$)



(d) Time step 30000 ($z = 1.0$) with $\Phi$

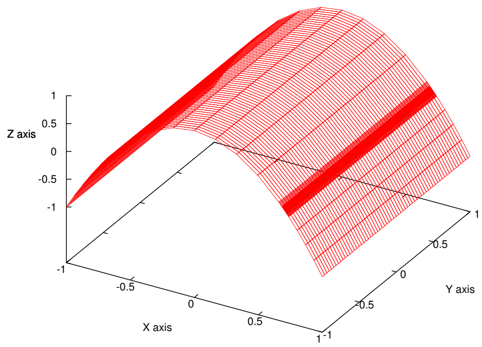


(e) Time step 80000 ($z = 0.5$)

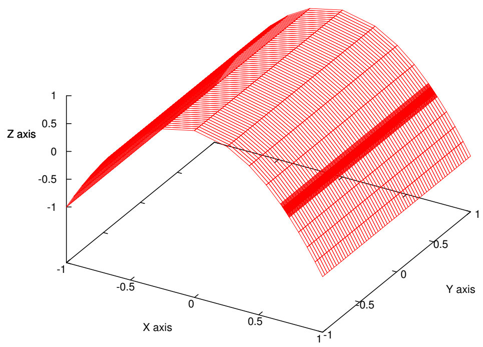


(f) Time step 80000 ($z = 0.5$) with $\Phi$

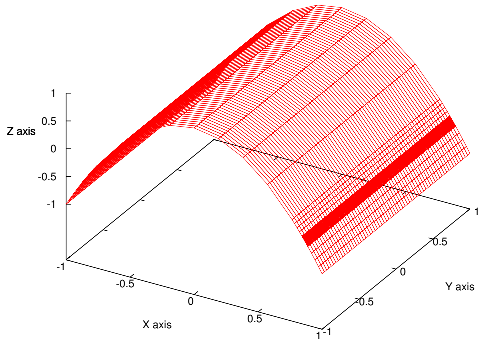
Figure 3.4: Comparison of mesh movement on surface (3.19) with and without the parameterization defined in Section 3.3.2.



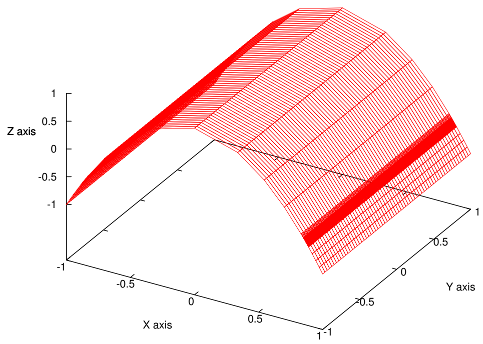
(a) Time step 130000 ($z = 0.0$)



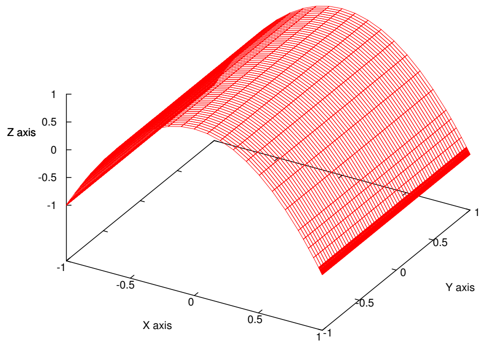
(b) Time step 130000 ($z = 0.0$) with $\Phi$



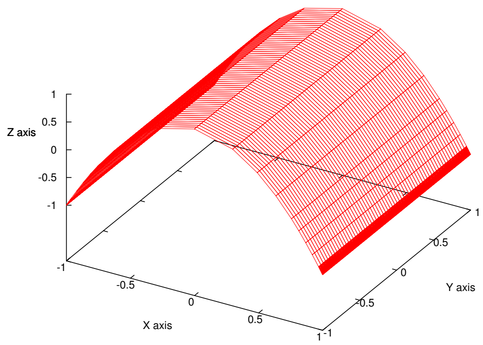
(c) Time step 180000 ($z = -0.5$)



(d) Time step 180000 ($z = -0.5$) with $\Phi$



(e) Time step 230000 ($z = -1.0$)



(f) Time step 230000 ($z = -1.0$) with $\Phi$

Figure 3.5: Comparison of mesh movement on surface (3.19) with and without the parameterization defined in Section 3.3.2 (continued from Figure 3.4).

# Chapter 4

# Conclusion

This thesis introduces the fundamental theory of MMPDEs and derives in detail the equations combining MMPDE5 and Winslow's monitor function. It also gives a finite-element formulation of that MMPDE in $\mathbb{R}^2$, implemented in C++ with the help of the finite-element library deal.II. The code was used to solve both time-independent and time-dependent problems, and convergence was analyzed with respect to the space and time variables. Chapter 3 presents an introduction to the field of PDEs on curved surfaces. It also gives equations for an MMPDE on general surfaces and derives the corresponding finite-element formulation. A simple method using the parameterization of the surface is presented to compute solutions to MMPDEs on surfaces. The definition of the parameterization is discussed and implemented, building on top of the code developed for the planar case. A time-dependent problem on a quadratic surface is solved, illustrating various properties of the technique. In particular, the mesh concentrates where the monitor function is maximum, and the density in that zone depends on the inclination of the surface.

This work is to our knowledge the first to consider adaptive solutions to PDEs on surfaces using MMPDEs. The amount of work remaining is therefore vast. A list of some potential future directions is the following:

- Parallelization and code optimization: Several steps of the computation could take advantage of multi-threading on multi-core computers: the initialization phase of the recovery technique, the assembling and the solution of the system, the computation of the boundary conditions, and the computation of the parametrized coordinates on the surface are just a few examples of steps that could highly benefit from a parallel implementation.

- Solution to MMPDEs defined on surfaces: This technique was presented in Section 3.2, where the equations and corresponding finite-element formulation were derived. deal.II's version 7.0 supports the definition of mappings from a reference cell in the plane onto the triangulation of a two-dimensional surface in $\mathbb{R}^3$. This allows the solution of MMPDEs defined on general surfaces, which would provide a way to better evaluate the parametric MMPDE method defined in Section 3.3.

- General surfaces: Parametric MMPDEs were only solved on certain specialized surfaces. Specifically, we only considered surfaces which are graphs depending on a single parameter. The definition of the parameterization for a general surface was not considered in detail in this work. A more general parameterization technique would allow computations to be made on more complex surfaces using the existing code.

- Adaptive solutions to PDEs on curved surfaces: After validation of the moving mesh strategy, it can be used as a component to the adaptive solution to physical PDEs on curved surfaces.

# Appendix A: Convention for vector differentiation

We present the convention used for vector differentiations, as defined in the book by Huang & Russell [HR10]. A vector is assumed to be a column vector.

- Differentiation of a vector by a scalar gives a vector. Example: the covariant base vector

$$\vec{a}_i = \frac{\partial \vec{x}}{\partial \xi_i} = \frac{\partial}{\partial \xi_i} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}.$$

- Differentiation of a vector by a vector gives a $n \times n$ matrix in $\mathbb{R}^n$. Example: the Jacobian matrix

$$\vec{J} = \frac{\partial \vec{x}}{\partial \vec{\xi}} = \nabla_{\vec{\xi}}\, \vec{x} = \frac{\partial(x_1, x_2, x_3)}{\partial(\xi_1, \xi_2, \xi_3)} = \begin{bmatrix} \partial x_1/\partial \xi_1 & \partial x_1/\partial \xi_2 & \partial x_1/\partial \xi_3 \\ \partial x_2/\partial \xi_1 & \partial x_2/\partial \xi_2 & \partial x_2/\partial \xi_3 \\ \partial x_3/\partial \xi_1 & \partial x_3/\partial \xi_2 & \partial x_3/\partial \xi_3 \end{bmatrix}.$$

- Differentiation of a scalar by a vector gives a vector. Example: the differentiation of a scalar function $F(\vec{a}^1, \vec{a}^2, \vec{a}^3, J, \vec{x})$ by the contravariant base vector ($\vec{a}^i = \nabla \xi_i = \left[a_1^i, a_2^i, a_3^i\right]^T$) gives

$$\frac{\partial F}{\partial \vec{a}^i} = \begin{pmatrix} \partial F/\partial a_1^i \\ \partial F/\partial a_2^i \\ \partial F/\partial a_3^i \end{pmatrix}.$$

- Differentiation of a scalar twice by vectors gives a $n \times n$ matrix in $\mathbb{R}^n$. Example: the differentiation of the integrand by two contravariant base vectors gives

$$\frac{\partial^2 F}{\partial \vec{a}^i \partial \vec{a}^k} = \begin{bmatrix} \partial^2 F/(\partial a_1^i \partial a_1^k) & \partial^2 F/(\partial a_1^i \partial a_2^k) & \partial^2 F/(\partial a_1^i \partial a_3^k) \\ \partial^2 F/(\partial a_2^i \partial a_1^k) & \partial^2 F/(\partial a_2^i \partial a_2^k) & \partial^2 F/(\partial a_2^i \partial a_3^k) \\ \partial^2 F/(\partial a_3^i \partial a_1^k) & \partial^2 F/(\partial a_3^i \partial a_2^k) & \partial^2 F/(\partial a_3^i \partial a_3^k) \end{bmatrix}.$$

# Appendix B: Verification of the MMPDE formulation

We check the MMPDE formulation derived in Section 2.2 by using alternate formulas.

**Verification for the coefficients $A_{i,j}$ and $B_i$**

Under the assumptions made in Section 2.2, the integrand can be decomposed into two parts [HR10],

$$F(\vec{a}^1,\vec{a}^2,J,\vec{x}) = F_1(\rho,\beta) + F_2(\rho,J),$$

with $\rho = \sqrt{\det(M)} = \omega$, and

$$\beta = \sum_i (\nabla\xi_i)^T \cdot M^{-1} \cdot \nabla\xi_i = \frac{1}{\omega}\left(\|\vec{a}^1\|^2 + \|\vec{a}^2\|^2\right).$$

When this decomposition of the integrand is possible, equation (4.1) gives the expression for coefficients $A_{i,j}$ and $B_i$ [HR10] of

$$\begin{aligned}
A_{i,j} =& 4\frac{\partial^2 F_1}{\partial\beta^2}(M^{-1}\cdot\vec{a}^i)(M^{-1}\cdot\vec{a}^j)^T \sum_k \vec{a}^k\cdot(\vec{a}^k)^T + 2\frac{\partial F_1}{\partial\beta}\left((\vec{a}^i)^T\cdot M^{-1}\cdot\vec{a}^j\right) I \\
&+ J^2\left(\frac{2}{J}\frac{\partial F_2}{\partial J} + \frac{\partial^2 F_2}{\partial J^2}\right)\vec{a}_i\cdot(\vec{a}^j)^T, \\
B_i =& -\left\{\sum_k \left[2\frac{\partial^2 F_1}{\partial\beta^2}\left((\vec{a}^k)^T\cdot M^{-1}\cdot\vec{a}^i\right)\left(\sum_l (\vec{a}^l)^T\cdot\frac{\partial M^{-1}}{\partial\xi_k}\cdot\vec{a}^l\right)\right.\right. \\
&\left.\left. + 2\frac{\partial^2 F_1}{\partial\beta\partial\rho}\left((\vec{a}^k)^T\cdot M^{-1}\cdot\vec{a}^i\right)\frac{\partial\rho}{\partial\xi_k} + 2\frac{\partial F_1}{\partial\beta}\left((\vec{a}^k)^T\cdot\frac{\partial M^{-1}}{\partial\xi_k}\cdot\vec{a}^i\right)\right] - J\frac{\partial^2 F_2}{\partial J\partial\rho}\frac{\partial\rho}{\partial\xi_i}\right\} I.
\end{aligned} \tag{4.1}$$

From equation (2.35), we deduce the expressions

$$\begin{cases} F_1(\rho,\beta) = \dfrac{\beta}{2}, \\ F_2(\rho,J) = 0. \end{cases}$$

We can now plug these into equations (4.1), yielding

$$\begin{aligned} A_{i,j} &= 2\frac{\partial F_1}{\partial \beta}\left((\vec{a}^i)^T \cdot M^{-1} \cdot \vec{a}^j\right) I_2, \\ &= \frac{1}{\omega}\left((\vec{a}^i)^T \cdot \vec{a}^j\right) I_2, \\ B_i &= -2\frac{\partial F_1}{\partial \beta}\sum_{k=1}^{2}\left((\vec{a}^k)^T \cdot \frac{\partial M^{-1}}{\partial \xi_k} \cdot \vec{a}^i\right), \\ &= -\left((\vec{a}^1)^T \cdot \frac{\partial}{\partial \xi}\left(\frac{1}{\omega}\right) \cdot I_2 \cdot \vec{a}^i + (\vec{a}^2)^T \cdot \frac{\partial}{\partial \eta}\left(\frac{1}{\omega}\right) \cdot I_2 \cdot \vec{a}^i\right), \\ &= \frac{1}{\omega^2}\left(\omega_\xi \cdot (\vec{a}^1)^T \cdot \vec{a}^i + \omega_\eta \cdot (\vec{a}^2)^T \vec{a}^i\right). \end{aligned} \tag{4.2}$$

For $A_{i,j}$, we immediately have the correspondence between the formulations (4.2) and (2.43). For $B_i$ though, a modification is required to make a comparison,

$$\begin{aligned} B_1 &= \frac{1}{J^2\omega^2}\left(\omega_\xi\left(x_\eta^2 + y_\eta^2\right) - \omega_\eta\left(x_\xi x_\eta + y_\xi y_\eta\right)\right), \\ B_2 &= \frac{1}{J^2\omega^2}\left(-\omega_\xi\left(x_\xi x_\eta + y_\xi y_\eta\right) + \omega_\eta\left(x_\xi^2 + y_\xi^2\right)\right). \end{aligned} \tag{4.3}$$

Equations (4.3) now correspond to the formulation (2.42).

## Verification of MMPDE5

In Huang & Russell [HR01], an alternate equation for MMPDE5 with Winslow monitor function is proposed,

$$\omega^3 \tau \dot{\vec{x}} = \sum_{i,j}\left(\vec{a}^i \cdot \vec{a}^j\right)\frac{\partial}{\partial \xi^j}\left(\omega \frac{\partial \vec{x}}{\partial \xi^i}\right). \tag{4.4}$$

In two dimensions, equation (4.4) becomes

$$\begin{aligned} \omega^3 \tau \dot{\vec{x}} = \|\vec{a}^1\|^2 \frac{\partial}{\partial \xi}\left(\omega \frac{\partial \vec{x}}{\partial \xi}\right) + \vec{a}^1 \cdot \vec{a}^2\left[\frac{\partial}{\partial \xi}\left(\omega \frac{\partial \vec{x}}{\partial \eta}\right) + \frac{\partial}{\partial \eta}\left(\omega \frac{\partial \vec{x}}{\partial \xi}\right)\right] \\ + \|\vec{a}^2\|^2 \frac{\partial}{\partial \eta}\left(\omega \frac{\partial \vec{x}}{\partial \eta}\right). \end{aligned}$$

Using the relations between covariant and contravariant vectors, we get

$$\begin{aligned}\dot{\vec{x}} = \frac{1}{J^2\omega^3\tau}\Bigg\{&\left(x_\eta^2+y_\eta^2\right)\frac{\partial}{\partial\xi}\left(\omega\frac{\partial\vec{x}}{\partial\xi}\right)-\left(x_\xi x_\eta+y_\xi y_\eta\right)\frac{\partial}{\partial\eta}\left(\omega\frac{\partial\vec{x}}{\partial\xi}\right)\\ &-\left(x_\xi x_\eta+y_\xi y_\eta\right)\frac{\partial}{\partial\xi}\left(\omega\frac{\partial\vec{x}}{\partial\eta}\right)+\left(x_\xi^2+y_\xi^2\right)\frac{\partial}{\partial\eta}\left(\omega\frac{\partial\vec{x}}{\partial\eta}\right)\Bigg\}.\end{aligned} \tag{4.5}$$

We notice two differences between (4.5) and (2.45). The function $p(\vec{x},t)$ does not appear in equation (4.5), but it has an extra factor $1/\omega$. Because the factor $p$ is used to better distribute the movement of the mesh, we used the formulation (2.45) in this thesis.

# Bibliography

[BHK07] W. Bangerth, R. Hartmann, and G. Kanschat. deal.II – a general purpose object oriented finite element library. *ACM Trans. Math. Softw.*, 33(4):24/1 – 24/27, 2007.

[BK] W. Bangerth and G. Kanschat. *deal.II Differential Equations Analysis Library, Technical Reference*. `http://www.dealii.org`.

[Bra07] D. Braess. *Finite elements: theory, fast solvers, and applications in elasticity theory*. Cambridge University Press, 2007.

[CHR99] W. Cao, W. Huang, and R.D. Russell. An r-Adaptive Finite Element Method Based upon Moving Mesh PDEs. *Journal of Computational Physics*, 149(2):221 – 244, 1999.

[Cow05] C. Cowan. The Cahn-Hilliard equation as a gradient flow. Master's thesis, Simon Fraser University, 2005.

[FH05] M.S. Floater and K. Hormann. Surface parameterization: a tutorial and survey. In N.A. Dodgson, M.S. Floater, and M.A. Sabin, editors, *Advances in Multiresolution for Geometric Modelling*, Mathematics and Visualization, pages 157–186. Springer, Berlin, Heidelberg, 2005.

[Gao11] W. Gao. *The Laplace-Beltrami Operator in a Level Set Framework and Its Applications*. PhD thesis, University of California, Los Angeles, 2011.

[GF64] I.M. Gelfand and S.V. Fomin. *Calculus of variations*. Prentice-Hall, 1964.

[HR01] W. Huang and R.D. Russell. Adaptive mesh movement – the MMPDE approach and its applications. *Journal of Computational and Applied Mathematics*, 128(1-2):383 – 398, 2001.

[HR10] W. Huang and R.D. Russell. *Adaptive Moving Mesh Methods*. Applied Mathematical Sciences. Springer, 2010.

[Küh06] W. Kühnel. *Differential geometry: curves - surfaces - manifolds*. Student mathematical library. American Mathematical Society, 2006.

[Mac08] C.B. Macdonald. *The closest point method for time-dependent processes on surfaces*. PhD thesis, Simon Fraser University, Burnaby, 2008.

[MT03] J.E. Marsden and A. Tromba. *Vector calculus*. Number pp. 1-93. W.H. Freeman, 2003.

[Ned80] J. C. Nedelec. Mixed finite elements in $\mathbb{R}^3$. *Numerische Mathematik*, 35:315–341, 1980. 10.1007/BF01396415.

[Pra01] S. Prata. *C++ primer plus*. Sams, 2001.

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

[Wik] Wikipedia. *Divergence – Generalizations*. `http://en.wikipedia.org/wiki/Divergence#Generalizations`.

[XHRW11] X. Xu, W. Huang, R.D. Russell, and J.F. Williams. Convergence of de Boor's algorithm for the generation of equidistributing meshes. *IMA Journal of Numerical Analysis*, 31(2):580–596, 2011.