Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Engineering 00 (2015) 000–000

Procedia Engineering

www.elsevier.com/locate/procedia

24th International Meshing Roundtable (IMR24)

Moving mesh methods on parametric surfaces

Benjamin Crestel[a,*], Robert D. Russell[b], Steven J. Ruuth[b]

[a]*Institute for Computational Engineering & Sciences, The University of Texas at Austin, Austin, TX 78712.*
[b]*Department of Mathematics, Simon Fraser University, Burnaby, BC, Canada.*

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 Winslow's adaptation functional. 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 using deal.II. The results, for both fixed and dynamically adapting meshes, demonstrate the effectiveness of the method.

Peer-review under responsibility of organizing committee of the 24th International Meshing Roundtable (IMR24).

Keywords: Adaptivity; moving mesh methods; moving mesh partial differential equations; surfaces; deal.II

1. Introduction

The solutions to partial differential equations (PDEs) defined on surfaces are of great interest to many researchers in the applied and natural sciences. Applications emerge from fields as diverse as biology (e.g., pattern formation on surfaces [3]), material science [20], image processing (e.g., segmentation on surfaces [19]) and fluid dynamics [21], to name but a few. But while numerical methods for PDEs in $\mathbb{R}^d$ have been developed and analyzed in a vast and thorough body of 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 all require the definition of a mesh. To accurately solve these PDEs, it is often necessary to adapt the mesh to the specifics of the problem. Adaptive techniques for PDEs are traditionally sorted into three categories:

1. We can adapt the mesh by adding or removing grid points in selected parts of the domain. This is called *h*-adaptivity.

* Corresponding author. Tel.: 1-512-232-7137 ; fax: 1-512-471-8694.
E-mail address: crestel@ices.utexas.edu

1877-7058
Peer-review under responsibility of organizing committee of the 24th International Meshing Roundtable (IMR24).

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 initial grid points may be evolved 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. This *moving mesh* approach is the adaptive technique that we consider here.

Moving mesh methods are efficient for a variety of problems in $\mathbb{R}^d$ [15], including those involving blow-up [4] and self-similarity [5]. A PDE explicitly involving the mesh speed, called a moving mesh PDE (MMPDE), is sometimes introduced to yield a particularly robust technique to compute the mesh transformation. The theory of MMPDEs for the adaptive computation of PDEs has been developed over the last decade by Huang, Russell and collaborators [7,14,15]. In these methods, a two-step approach is typically adopted, one for the solution of the mesh and another for the computation of the physical PDE solution. While in theory a simultaneous solution is possible, it is typically observed that little benefit is derived from such a coupling.

There has been very little research carried out on applying moving mesh methods to the solution of PDEs on surfaces. Indeed, to our knowledge there are presently no published articles on solving MMPDEs on surfaces. Our objectives are to study the feasibility of this idea and to propose methods to compute moving meshes on surfaces. As a first step to understanding these methods better, we consider the evolution of moving meshes on simple surfaces that accept a continuously invertible parameterization.

This paper is organized as follows. In section 2, we introduce the mathematical background of MMPDEs and derive the equations and finite element formulation for a class of problems in the plane. In the conclusion of this section, we implement our method and solve a problem with the C++ finite element library deal.II. In the following section, 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. Section 4 gives numerical experiments for a selection of parameterized surfaces in 3D. Lastly, section 5 presents the conclusions of this work.

2. A primer on moving mesh partial differential equations

This section begins with a brief introduction to the theory and derivation of MMPDEs. Next, a method and its discretization are given, followed by a 2D numerical experiment. Further details on MMPDE methods may be found in [7,14,15,22].

2.1. Overview

Denote by Ω the given physical domain that we wish to adaptively mesh. Further, denote by Ω_c the computational domain; this domain is often chosen to be a simply connected domain that can be meshed easily. A moving mesh transformation is defined by the smooth mapping x that takes a coordinate in the computational domain and returns a coordinate in the physical domain:

$$\begin{aligned} x : \Omega_c &\to \Omega, \\ \boldsymbol{\xi} &\mapsto \boldsymbol{x}. \end{aligned} \tag{1}$$

The choice of transformation mapping x defines the moving mesh technique.

Moving mesh PDEs are partial differential equations whose solution is a transformation mapping that preserves certain mesh properties. A survey of these properties and the derivation of MMPDEs follows. Notably, MMPDEs should generally be based on the inverse coordinate transformation $\boldsymbol{\xi} = \xi(\boldsymbol{x})$ rather than the direct coordinate transformation as this guarantees existence and uniqueness of the solution, as well as prevents the eventual folding of the mesh [15]. Thus, in practice, we will solve the MMPDE formulated in terms of $\xi(\boldsymbol{x})$ and then determine the transformed coordinate $x(\boldsymbol{\xi})$ by a Newton's method approach, as described in [10].

2.1.1. Derivation of MMPDEs

MMPDEs are derived by minimizing an adaptation functional. Of interest to us are general adaptation functionals of the form

$$I[\boldsymbol{\xi}] = \int_{\Omega} F(\nabla\boldsymbol{\xi}, \boldsymbol{\xi}, \boldsymbol{x})\, d\boldsymbol{x}, \tag{2}$$

where the integrand F enforces the properties of the mesh that we wish to conserve.

Setting the first variation of the functional (2) to zero leads to the corresponding Euler-Lagrange equation. After lengthy calculations [15], we obtain the elliptic PDE

$$\sum_{i,j} A_{i,j} \frac{\partial^2 \boldsymbol{x}}{\partial\xi_i \partial\xi_j} + \sum_i B_i \frac{\partial \boldsymbol{x}}{\partial\xi_i} = 0, \tag{3}$$

where

$$\begin{aligned} A_{i,j} = &\sum_{k,s} \left((\boldsymbol{a}^i)^T \frac{\partial^2 F}{\partial \boldsymbol{a}^s \partial \boldsymbol{a}^k} \boldsymbol{a}^j \right) \boldsymbol{a}_s (\boldsymbol{a}^k)^T - J \sum_s \left((\boldsymbol{a}^i)^T \frac{\partial^2 F}{\partial \boldsymbol{a}^s \partial J} \right) \boldsymbol{a}_s (\boldsymbol{a}^j)^T \\ &- J \sum_k \left((\boldsymbol{a}^i)^T \frac{\partial^2 F}{\partial J \partial \boldsymbol{a}^k} \right) \boldsymbol{a}_i (\boldsymbol{a}^k)^T + J^2 \left(\frac{2}{J} \frac{\partial F}{\partial J} + \frac{\partial^2 F}{\partial J^2} \right) \boldsymbol{a}_i (\boldsymbol{a}^j)^T, \end{aligned} \tag{4}$$

$$B_i = -\sum_k \left((\boldsymbol{a}^k)^T \frac{\partial^2 F}{\partial \boldsymbol{a}^i \partial \boldsymbol{x}} \boldsymbol{a}_k \right) + J \boldsymbol{a}_i \left(\frac{\partial^2 F}{\partial J \partial \boldsymbol{x}} \right)^T. \tag{5}$$

The quantities $\boldsymbol{a}_i := \partial_{\xi_i} \boldsymbol{x}$ and $\boldsymbol{a}^i := \nabla\xi_i$ are called the covariant and contravariant base vectors, respectively. The matrix $\boldsymbol{J} := \partial\boldsymbol{x}/\partial\boldsymbol{\xi}$ is the standard Jacobian matrix and we denote its determinant by J. Finally, the 3×3 matrices $\frac{\partial^2 F}{\partial \boldsymbol{a}^s \partial \boldsymbol{a}^k}$ and $\frac{\partial^2 F}{\partial \boldsymbol{a}^s \partial \boldsymbol{x}}$ are defined componentwise by

$$\left(\frac{\partial^2 F}{\partial \boldsymbol{a}^i \partial \boldsymbol{a}^k} \right)_{(m,n)} = \frac{\partial^2 F}{\partial (\boldsymbol{a}^i)_m \partial (\boldsymbol{a}^k)_n}, \quad \left(\frac{\partial^2 F}{\partial \boldsymbol{a}^i \partial \boldsymbol{x}} \right)_{(m,n)} = \frac{\partial^2 F}{\partial (\boldsymbol{a}^i)_m \partial x_n}.$$

Typically, solving (3) is done efficiently by solving the corresponding gradient flow equation [9,13]. This turns the elliptic PDE (3) into an MMPDE involving the direct coordinate transformation $\boldsymbol{x}$,

$$\frac{\partial \boldsymbol{x}}{\partial t} = \frac{1}{\tau b(\boldsymbol{x}, t)} \left[\sum_{i,j} A_{i,j} \frac{\partial^2 \boldsymbol{x}}{\partial\xi_i \partial\xi_j} + \sum_i B_i \frac{\partial \boldsymbol{x}}{\partial\xi_i} \right]. \tag{6}$$

The *balancing function* $b(\boldsymbol{x}, t)$ introduced by the gradient flow should be chosen so that all mesh points move with a uniform time scale, while the user-controlled parameter $\tau > 0$ is chosen to adjust the time-scale of the mesh movement. See [15] for details.

2.1.2. Definition of the adaptation functional

In two or more dimensions, an effective adaptation functional (2) can be designed by combining the concepts of *equidistribution* and *alignment*. We now review these concepts, and give our choice of adaptation functional.

We begin by illustrating the concept of equidistribution for a simple one-dimensional example. Working on a closed interval $[a, b]$ and given a function $\rho(x) > 0$, an equidistributed mesh $\{x_i\} : a = x_1 < x_2 < \cdots < x_N = b$ satisfies

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

The function ρ controls the density of the mesh points and is referred to as the *monitor function*. To extend this concept to a continuous equidistribution principle, we introduce the coordinate transformation $x(\xi)$:

$$x = x(\xi) : [0, 1] \longrightarrow [a, b]. \tag{7}$$

The coordinate transformation (7) is an *equidistributing coordinate transformation* for $\rho(x)$ if it satisfies

$$\int_a^{x(\xi)} \rho(z)dz = \sigma\xi, \quad 0 \le \xi \le 1, \tag{8}$$

with $\sigma = \int_a^b \rho(z)dz$.

In two or more dimensions, equidistribution imposes that all elements have a constant volume, i.e.,

$$\int_K \rho(\boldsymbol{x})\, d\boldsymbol{x} = \frac{\sigma}{N}, \quad \forall K \in T_h,$$

where N is the number of the elements of T_h and $\sigma = \int_\Omega \rho(\boldsymbol{x})\, d\boldsymbol{x}$. While equidistribution uniquely defines a mesh transformation in the one dimensional case, in higher dimensions each cell will only be defined up to a rotation. To specify these degrees of freedom we apply the concept of *alignment*, which leads to preferred directions for the cell edges to follow. Specifically, we require that all elements are equilateral in a given metric M. Letting $\gamma_1, \ldots, \gamma_{d(d+1)/2}$ be the edges of an element K in $\mathbb{R}^d$, this condition requires that

$$|\gamma_1|_M = \ldots = |\gamma_{d(d+1)/2}|_M, \quad \forall K \in T_h,$$

where $|\gamma_i|_M$ denotes the length of the edge γ_i in the metric M.

To complete the formulation, we must incorporate equidistribution and alignment into an adaptation functional. Huang and Russell [15] present several adaptation functionals of this type. In this paper, we work with the simple and commonly used Winslow's adaptation functional given by $I_{\text{Win}} = \int_\Omega F_{\text{Win}}(\nabla\boldsymbol{\xi}, \boldsymbol{\xi}, \boldsymbol{x})\, d\boldsymbol{x}$ (see equation (2)) where

$$F_{\text{Win}}[\{\boldsymbol{a}_i\}_i, \{b_i\}_i, \{c_i\}_i] = \frac{1}{2\omega}\sum_i \boldsymbol{a}_i^T \boldsymbol{a}_i = \frac{1}{2\omega}\sum_i \|\boldsymbol{a}_i\|^2,$$

and $\omega = \omega(\boldsymbol{x})$ is a monitor function[1]. This leads to the functional

$$I_{\text{Win}}[\boldsymbol{\xi}] = \frac{1}{2}\int_\Omega \frac{1}{\omega}\sum_i (\nabla\xi_i)^T \nabla\xi_i\, d\boldsymbol{x} = \frac{1}{2}\int_\Omega \frac{1}{\omega}\sum_i \|\nabla\xi_i\|^2\, d\boldsymbol{x}. \tag{9}$$

2.2. Numerical solution of MMPDEs via the finite element method

In this section, we derive a finite element formulation for the MMPDE (6) from the Winslow's adaptation functional (9). The method is illustrated with an example of moving mesh in two dimensions. For the sake of clarity, the computational domain is referenced by the coordinates (ξ, η) while the physical domain is expressed in terms of the coordinates (x, y).

2.2.1. An MMPDE formulation with Winslow's adaptation functional

We begin by deriving the expression for the MMPDE (6) based on the Winslow's adaptation function (9), in $\mathbb{R}^2$. To that effect, we substitute the functional (9) into the equations for A_{ij} (4) and B_{ij} (5). This yields

$$B_1 = \frac{1}{J^2\omega^2}\left[\left(x_\eta^2 + y_\eta^2\right)\omega_\xi - \left(x_\xi x_\eta + y_\xi y_\eta\right)\omega_\eta\right], \quad B_2 = \frac{1}{J^2\omega^2}\left[-\left(x_\xi x_\eta + y_\xi y_\eta\right)\omega_\xi + \left(x_\xi^2 + y_\xi^2\right)\omega_\eta\right], \tag{10}$$

and

$$A_{1,1} = \frac{1}{J^2\omega}\left(x_\eta^2 + y_\eta^2\right) I_2, \quad A_{1,2} = -\frac{1}{J^2\omega}\left(x_\xi x_\eta + y_\xi y_\eta\right) I_2 = A_{2,1}, \quad A_{2,2} = \frac{1}{J^2\omega}\left(x_\xi^2 + y_\xi^2\right) I_2, \tag{11}$$

[1] In a physical application, this monitor function would be based on the solution of the physical problem of interest. Physical applications being outside the scope of this paper, we will use simple, artificial monitor functions that help illustrate the behaviour of the method.

where I_2 is the 2×2 identity matrix [10]. By substituting these coefficients into equation (6) and re-arranging terms we obtain the desired MMPDE:

$$\frac{\partial \boldsymbol{x}}{\partial t} = \frac{1}{J^2\omega^2\tau b(\boldsymbol{x},t)} \left\{ \left(x_\eta^2 + y_\eta^2\right) \frac{\partial}{\partial \xi}\left(\omega \frac{\partial \boldsymbol{x}}{\partial \xi}\right) - \left(x_\xi x_\eta + y_\xi y_\eta\right) \frac{\partial}{\partial \eta}\left(\omega \frac{\partial \boldsymbol{x}}{\partial \xi}\right) \right. \\ \left. - \left(x_\xi x_\eta + y_\xi y_\eta\right) \frac{\partial}{\partial \xi}\left(\omega \frac{\partial \boldsymbol{x}}{\partial \eta}\right) + \left(x_\xi^2 + y_\xi^2\right) \frac{\partial}{\partial \eta}\left(\omega \frac{\partial \boldsymbol{x}}{\partial \eta}\right) \right\}. \quad (12)$$

Grid points along the boundaries of the domain are also moved adaptively [7]. In our method [10], their positions are evolved by solving the one-dimensional equidistribution equation (8) using de Boor's algorithm [11]. Note that the monitor function for the boundary equation should be chosen so as to ensure that the adaptivity along the boundaries and interior of the domain are consistent; see [10] for details. In practice, we choose $\rho(\boldsymbol{x}) = \omega(\boldsymbol{x})$ for any $\boldsymbol{x} \in \partial\Omega$.

2.2.2. Variational formulation

The weak formulation is obtained by multiplying each side of equation (12) by a function $v \in (H_0^1(\Omega_c))^2$, where $H_0^1(\Omega_c)$ denotes the Sobolev space of $L^2(\Omega_c)$ functions that are compactly supported on Ω_c with a weak derivative that is in $L^2(\Omega_c)$. Integrating over the domain and applying integration by parts to the right-hand side yields:

$$\int_{\Omega_c} A(\boldsymbol{x})\,(\dot{\boldsymbol{x}}\cdot v)\,d\boldsymbol{\xi} = \int_{\Omega_c} \omega(\boldsymbol{x}) \left\{ -\frac{\partial}{\partial \xi}\left(\alpha_1(\boldsymbol{x})v\right) \cdot \frac{\partial \boldsymbol{x}}{\partial \xi} \right. \\ \left. + \left[\frac{\partial}{\partial \eta}\left(\alpha_2(\boldsymbol{x})v\right) \cdot \frac{\partial \boldsymbol{x}}{\partial \xi} + \frac{\partial}{\partial \xi}\left(\alpha_2(\boldsymbol{x})v\right) \cdot \frac{\partial \boldsymbol{x}}{\partial \eta} \right] - \frac{\partial}{\partial \eta}\left(\alpha_3(\boldsymbol{x})v\right) \cdot \frac{\partial \boldsymbol{x}}{\partial \eta} \right\} d\boldsymbol{\xi}, \quad (13)$$

where $\alpha_1(\boldsymbol{x}) = x_\eta^2 + y_\eta^2$, $\alpha_2(\boldsymbol{x}) = x_\xi x_\eta + y_\xi y_\eta$, $\alpha_3(\boldsymbol{x}) = x_\xi^2 + y_\xi^2$ and $A(\boldsymbol{x}) = J^2\omega^2\tau b(\boldsymbol{x},t)$.

Clearly, the weak form (13) requires partial derivatives of the α-coefficients, and consequently second partial derivatives of the solution. However finite-element solutions that are (globally) continuously differentiable are much more expensive (computationally speaking) and complex than continuous elements. In order to circumvent this difficulty, we compute second derivative information via a recovery technique [6]. This leads to an alternate MMPDE in which the nonlinear α-coefficients in (13) are replaced respectively by $\tilde{\alpha}_1(\boldsymbol{x}) = \tilde{X}_\eta^2 + \tilde{Y}_\eta^2$, $\tilde{\alpha}_2(\boldsymbol{x}) = \tilde{X}_\xi\tilde{X}_\eta + \tilde{Y}_\xi\tilde{Y}_\eta$, $\tilde{\alpha}_3(\boldsymbol{x}) = \tilde{X}_\xi^2 + \tilde{Y}_\xi^2$ and $\tilde{A}(\boldsymbol{x}) = \tilde{J}^2\omega^2\tau\tilde{p}(\tilde{\boldsymbol{X}},t)$. The quantities $\tilde{X}_\xi$, $\tilde{X}_\eta$, $\tilde{Y}_\xi$ and $\tilde{Y}_\eta$ are globally continuous functions that equal the average of the functions x_ξ, x_η, y_ξ and y_η at each cell boundary [15].

2.2.3. A finite element method

Let's define the time step-size Δt and the time discretization $t_n = t_0 + n\Delta t$, for any $n = 0, 1, \ldots, N$. The numerical solution at time t_n is denoted by $\boldsymbol{x}^n$. We use a *lagged backwards Euler scheme*, i.e., a backward Euler scheme in which the nonlinear term is evaluated at time t_n. This leads to the fully discrete scheme

$$\int_{\Omega_c} \tilde{A}(\boldsymbol{x}^n)\left(\frac{\boldsymbol{x}^{n+1} - \boldsymbol{x}^n}{\Delta t}\cdot v\right) d\boldsymbol{\xi} = \int_{\Omega_c} \omega(\boldsymbol{x}^n) \left\{ -\frac{\partial}{\partial \xi}\left(\tilde{\alpha}_1(\boldsymbol{x}^n)v\right) \cdot \frac{\partial \boldsymbol{x}^{n+1}}{\partial \xi} \right. \\ \left. + \left[\frac{\partial}{\partial \eta}\left(\tilde{\alpha}_2(\boldsymbol{x}^n)v\right) \cdot \frac{\partial \boldsymbol{x}^{n+1}}{\partial \xi} + \frac{\partial}{\partial \xi}\left(\tilde{\alpha}_2(\boldsymbol{x}^n)v\right) \cdot \frac{\partial \boldsymbol{x}^{n+1}}{\partial \eta} \right] - \frac{\partial}{\partial \eta}\left(\tilde{\alpha}_3(\boldsymbol{x}^n)v\right) \cdot \frac{\partial \boldsymbol{x}^{n+1}}{\partial \eta} \right\} d\boldsymbol{\xi}. \quad (14)$$

For the spatial discretization we use a square computational domain Ω_c with edges parallel to the axes ξ and η. The domain is tessellated into a uniform mesh of square elements. We choose the finite element space V_h to be the set of piecewise bilinear functions, i.e., functions defined on $Q_1(K) = \text{span}(1, x, y, xy)$ with K being the reference element. The problem is now to look for an approximation $\boldsymbol{x}_h^n$ of $\boldsymbol{x}^n$ in V_h. Let $\{\phi_j, j = 1, \ldots, m\}$ be an orthonormal basis for the finite-dimensional space V_h. We can then write x_h^n in that basis to get

$$x_h^n(\boldsymbol{\xi}) = \sum_j C_j^n \phi_j(\boldsymbol{\xi}).$$

A similar expression holds for y_h^n. Substituting these expressions into Equation (14) leads to two matrix systems (one for each of x_h^{n+1} and y_h^{n+1}) of the form

$$\left(K_1^n - \Delta t K_2^n\right) C^{n+1} = K_1^n C^n, \tag{15}$$

where the coefficient matrices are

$$\left(K_1^n\right)_{(i,j)} = \int_{\Omega_{C_h}} \tilde{A}(\boldsymbol{x}_h^n)\phi_i\phi_j d\boldsymbol{\xi}, \tag{16}$$

$$\left(K_2^n\right)_{(i,j)} = \int_{\Omega_{C_h}} \omega(\boldsymbol{x}_h^n)\left(-\frac{\partial}{\partial\xi}\left(\tilde{\alpha}_1(\boldsymbol{x}_h^n)\phi_i\right)\frac{\partial\phi_j}{\partial\xi} + \frac{\partial}{\partial\eta}\left(\tilde{\alpha}_2(\boldsymbol{x}_h^n)\phi_i\right)\frac{\partial\phi_j}{\partial\xi} + \frac{\partial}{\partial\xi}\left(\tilde{\alpha}_2(\boldsymbol{x}_h^n)\phi_i\right)\frac{\partial\phi_j}{\partial\eta} - \frac{\partial}{\partial\eta}\left(\tilde{\alpha}_3(\boldsymbol{x}_h^n)\phi_i\right)\frac{\partial\phi_j}{\partial\eta}\right)d\boldsymbol{\xi}, \tag{17}$$

and $C^n = [C_1^n\, C_2^n \cdots C_m^n]^T$ is the desired solution vector. The matrices (16)-(17) are evaluated by Gaussian quadrature [10] and the non-symmetric system (15) is solved using BiCGStab at each time step [7,10].

Remark 1. The recovery technique requires that at each iteration the necessary information is collected before the recovery coefficients $\tilde{\alpha_1}, \tilde{\alpha_2}, \tilde{\alpha_3}$ and $\tilde{A}$ can be computed. Our choice of bilinear basis functions and identical square mesh cells on a square computational domain make this step considerably simpler. We skip the details on how to compute the recovery variables in deal.II and instead refer the interested reader to [10].

Remark 2. Due to the iterative nature of the BiCGStab solver, the computational complexity of the system (15) is difficult to estimate precisely. It depends on the number of grid nodes, the order of the finite element bases and the regularity of the monitor function and of the mesh solution. Note that this system requires the re-assembly of both matrices K_1^n and K_2^n at each time-step, which can become expensive for large systems. This cost of re-assembly can be alleviated by re-using the sparsity pattern of both matrices from one time-step to another.

2.2.4. Numerical experiment in two dimensions

To illustrate our method, we compute an evolving mesh for Winslow's adaptation functional with a monitor function that attains its peak values in a circular region centered at the origin,

$$\omega(\boldsymbol{x}, t) = 10\exp\left(-50\left|x^2 + y^2 - (1 - 0.0001t)^2\right|\right) + 1.$$

The circle defined by the peak values shrinks in time from a radius of 1 down to 0. We time step via iteration (15) on a 32×32 grid with a time step-size of $\Delta t = 0.001$. Boundary conditions are computed with de Boor's algorithm using $2^{12} + 1 = 4097$ grid points along each boundary. Figure 1 shows the results: the mesh concentrates around a circular shape that moves toward the center of the domain, and the radius of the evolving density profile coincides with the maximum concentration of the monitor function. This example was solved using the finite element package deal.II [1,2].

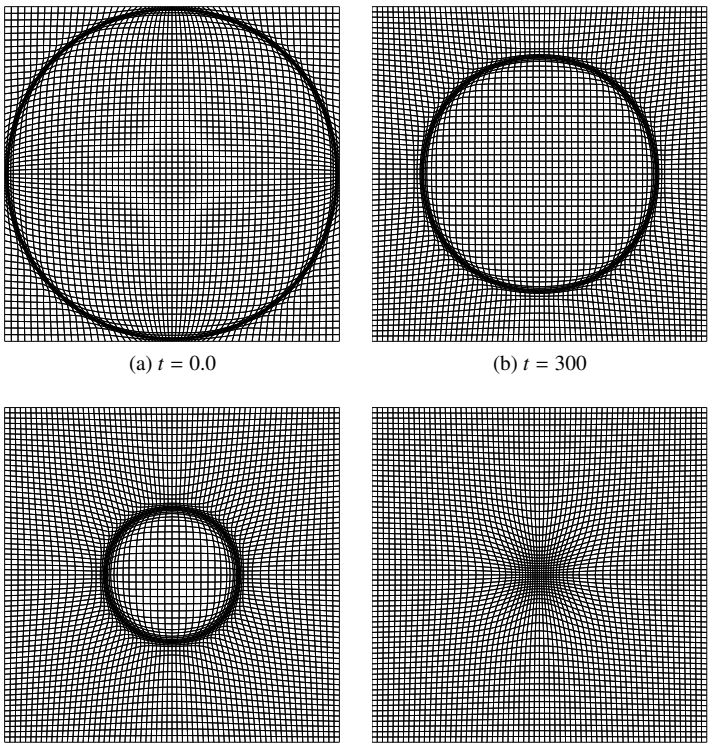
3. Moving mesh PDEs on general surfaces

The solution of PDEs on surfaces is of crucial importance for many scientific fields. Moreover, the efficient numerical solution of PDEs typically requires mesh adaptivity. In this section, we derive a class of MMPDEs on general surfaces by extending the framework introduced in section 2. We also present a formulation to solve these MMPDEs on surfaces that accept certain parameterizations.

3.1. Formulation of MMPDEs on general surfaces

This section derives a class of MMPDEs on Riemannian manifolds based on Winslow's adaptation functional. We begin with a review of some key concepts in differential and Riemannian geometry [8].

It is a well-known result of Riemannian geometry that the three classical derivative operators (gradient, divergence and Laplacian) have a generalization on Riemannian manifolds [8]. Consider a Riemannian manifold (M, g) where M



(a) $t = 0.0$ (b) $t = 300$

(c) $t = 600$ (d) $t = 1000$

Fig. 1: Example of an MMPDE solution with a time-dependent monitor function focused along a shrinking circle. This example uses Winslow's adaptation functional with $\omega(\boldsymbol{x}, t) = 10 \exp(-50\,|x^2 + y^2 - (1 - 0.001t)^2|) + 1$.

is a smooth manifold of dimension n and let (U, φ) be a chart with local coordinates $(x_1, \ldots, x_n) = \varphi(x)$ for $x \in M$. The $n \times n$ metric tensor g is defined componentwise as

$$g_{ij}(x) = \left\langle \left(\frac{\partial}{\partial x_i}\right)_x, \left(\frac{\partial}{\partial x_j}\right)_x \right\rangle_{g(x)},$$

where $\left\{\left(\frac{\partial}{\partial x_i}\right)_x, 1 \le i \le n\right\}$ is the natural basis for the tangent space at $x \in M$. Denote by $g^{ij}, 1 \le i, j \le n$, the entries of the inverse of g_{ij}. Then, for a map $f \in C^\infty(M)$ and a vector field $X = \sum_{i=1}^n b_i \frac{\partial}{\partial x_i} \in \Gamma_{C^\infty}$, the expressions for the three classical operators in local coordinates $(x_1, \ldots, x_n)$ are given by

$$\begin{aligned}
\nabla_g f &= \sum_{i,j=1}^n g^{ij} \frac{\partial f}{\partial x_i} \frac{\partial}{\partial x_j}, \\
\operatorname{div}_g X &= \operatorname{div}_g \left(\sum_{i=1}^n b_i \frac{\partial}{\partial x_i} \right) = \frac{1}{\sqrt{|\det g|}} \sum_{i=1}^n \frac{\partial}{\partial x_i} \left(b_i \sqrt{|\det g|} \right), \\
\Delta_g f &= \operatorname{div}_g(\nabla_g f) = \frac{1}{\sqrt{|\det g|}} \sum_{i=1}^n \frac{\partial}{\partial x_i} \left(\sum_{j=1}^n g^{ij} \frac{\partial f}{\partial x_i} \sqrt{|\det g|} \right).
\end{aligned} \tag{18}$$

The Laplacian on a manifold is often referred to as the Laplace-Beltrami operator.

Many of the important theorems in Euclidean spaces extend naturally to Riemannian manifolds. In particular, extensions of the divergence theorem and Green's theorem are available [8]. These results enable the derivation of certain MMPDEs on Riemannian manifolds using the same strategy as in Euclidean spaces. We now illustrate this

idea for the case of Winslow's functional and a two-dimensional manifold embedded in $\mathbb{R}^3$. By analogy with the Euclidean case (9), we define the surface functional as

$$I_g[\boldsymbol{\xi}] = \int_M \frac{1}{2\omega} \sum_{i=1}^{3} \|\nabla_g \xi_i\|^2 \, dM. \tag{19}$$

The first variation of functional $I_g[\boldsymbol{\xi}]$ becomes

$$\delta I_g[\boldsymbol{\xi}] = \int_M \frac{1}{\omega} \left[\sum_{i=1}^{3} \nabla_g \xi_i \cdot \nabla_g \delta\xi_i \right] dM.$$

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

$$I_g[\boldsymbol{\xi}] = -\int_M \sum_{i=1}^{3} \nabla_g \cdot \left(\frac{1}{\omega} \nabla_g \xi_i \right) \cdot \delta\xi_i \, dM.$$

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

$$-\nabla_g \cdot \left(\frac{1}{\omega} \nabla_g \xi_i \right) = 0, \quad \forall i = 1, 2, 3. \tag{20}$$

We do not solve equation (20) directly, but instead evolve the corresponding gradient flow equation

$$\frac{\partial \xi_i}{\partial t} = \frac{1}{\tau b} \nabla_g \cdot \left(\frac{1}{\omega} \nabla_g \xi_i \right), \quad \forall i = 1, 2, 3, \tag{21}$$

to a steady state solution. Similar to the case of Euclidean space, the balancing function $b(\boldsymbol{p}, t)$, with $\boldsymbol{p} \in M$, is introduced by the gradient flow and should be chosen so that all mesh points move with a uniform time scale. Finally, we remark that there is a need to map from computational to physical coordinates; this requirement introduces constraints on the choice of computational domain when a complex geometry arises.

3.2. Solution of MMPDEs on parameterized surfaces

A variety of numerical methods have been developed for the solution of PDEs on manifolds. Broadly speaking, these methods may be classified according to the manifold representation that they use; common choices include parametric, triangulated (or more generally polygonal) mesh and embedded representations [18]. Because we wish to extend the methods and software that we derived in section 2 to manifolds in $\mathbb{R}^3$, we consider manifolds that accept a parameterization Φ.

Assume that for any $(p, q, r) \in M$, there exists $(x, y) \in \mathbb{R}^2$ such that $\Phi(x, y) = (p(x, y), q(x, y), r(x, y))$. In this case, the manifold accepts a unique chart (U, φ) for all points $x \in M$, where $\varphi^{-1} = \Phi$ and U is the entire manifold M. Focusing on the vector field term inside the divergence in (21) and using the distributional definition of a vector field, we observe that

$$\frac{1}{\omega} \nabla_g \xi_i(f)(\boldsymbol{p}) = \frac{1}{\omega(\boldsymbol{p})} \sum_{j,k} g^{jk}(\boldsymbol{p}) \frac{\partial(\xi_i \circ \varphi^{-1})}{\partial x_j}(\varphi(\boldsymbol{p})) \frac{\partial(f \circ \varphi^{-1})}{\partial x_k}(\varphi(\boldsymbol{p})), \quad \forall i = 1, 2, 3$$

at a point $\boldsymbol{p} \in M$, for any $f \in C^\infty(M)$. This highlights the role played by φ. We use this insight to derive a simpler MMPDE acting over the manifold but defined in the Euclidean space $\varphi(M)$. First, we introduce some notation. Let the (Euclidean) physical domain be $\Omega := \varphi(M) \subset \mathbb{R}^2$, and assume the (Euclidean) computational domain coincides with the physical domain, i.e., $\Omega_c = \Omega$. We denote the vector coordinates of the mesh by $\boldsymbol{\xi} = (\xi_1, \xi_2, \xi_3)^T$, and introduce the modified coordinate transformation $\tilde{\boldsymbol{\xi}} := \varphi \circ \boldsymbol{\xi} \circ \varphi^{-1} : \Omega \to \Omega$, the modified monitor function $\tilde{\omega} :=$

$\omega \circ \varphi^{-1} : \Omega \to \mathbb{R}$ and the modified balance function $\tilde{b} := b \circ \varphi^{-1} : \Omega \to \mathbb{R}$. Our method is to replace equation (21), which is defined on the surface, with the following expression,

$$\frac{\partial \tilde{\xi}_j}{\partial t} = \frac{1}{\tau \tilde{b}(\boldsymbol{x}, t)} \nabla \cdot \left(\frac{1}{\tilde{\omega}(\boldsymbol{x})} \nabla \tilde{\xi}_j \right), \quad \forall j = 1, 2, \tag{22}$$

which is defined at any point $\boldsymbol{x} \in \Omega$. All of the differential operators appearing in (22) are now defined in Euclidean space. Note that we evaluate the monitor function $\tilde{\omega}(\boldsymbol{x})$ using values defined on the manifold. Specifically, for any $\boldsymbol{x} \in \Omega$ there exists $\boldsymbol{p} \in M$ such that $\boldsymbol{x} = \varphi(\boldsymbol{p})$. Thus, $\tilde{\omega}(\boldsymbol{x}) = \omega(\varphi^{-1}(\boldsymbol{x})) = \omega(\boldsymbol{p})$. After solving for $\tilde{\boldsymbol{\xi}} = (\tilde{\xi}_1, \tilde{\xi}_2)^T$, we recover the transformation for the mesh with the formula $\boldsymbol{\xi} = \varphi^{-1} \circ \tilde{\boldsymbol{\xi}} \circ \varphi$.

Similar to the planar setting considered in section 2.2.1, we can define MMPDEs in terms of the transformed mesh coordinates, i.e., we can formulate MMPDEs in terms of the direct transformation $x(\boldsymbol{\xi}) = \boldsymbol{x}$. Define the coordinate transformation $\tilde{\boldsymbol{x}} = \varphi \circ \boldsymbol{x} \circ \varphi^{-1}$. Our approximate formulation on a parameterized surface is given by MMPDE (12) where the monitor function is defined on the physical manifold, i.e.,

$$\begin{aligned} \frac{\partial \tilde{\boldsymbol{x}}}{\partial t} = \frac{1}{J^2 \tilde{\omega}^2(\tilde{\boldsymbol{x}}) \tau \tilde{b}(\tilde{\boldsymbol{x}}, t)} \Bigg\{ & \left(\tilde{x}_{\tilde{\eta}}^2 + \tilde{y}_{\tilde{\eta}}^2 \right) \frac{\partial}{\partial \tilde{\xi}} \left(\tilde{\omega}(\tilde{\boldsymbol{x}}) \frac{\partial \tilde{\boldsymbol{x}}}{\partial \tilde{\xi}} \right) - \left(\tilde{x}_{\tilde{\xi}} \tilde{x}_{\tilde{\eta}} + \tilde{y}_{\tilde{\xi}} \tilde{y}_{\tilde{\eta}} \right) \frac{\partial}{\partial \tilde{\eta}} \left(\tilde{\omega}(\tilde{\boldsymbol{x}}) \frac{\partial \tilde{\boldsymbol{x}}}{\partial \tilde{\xi}} \right) \\ & - \left(\tilde{x}_{\tilde{\xi}} \tilde{x}_{\tilde{\eta}} + \tilde{y}_{\tilde{\xi}} \tilde{y}_{\tilde{\eta}} \right) \frac{\partial}{\partial \tilde{\xi}} \left(\tilde{\omega}(\tilde{\boldsymbol{x}}) \frac{\partial \tilde{\boldsymbol{x}}}{\partial \tilde{\eta}} \right) + \left(\tilde{x}_{\tilde{\xi}}^2 + \tilde{y}_{\tilde{\xi}}^2 \right) \frac{\partial}{\partial \tilde{\eta}} \left(\tilde{\omega}(\tilde{\boldsymbol{x}}) \frac{\partial \tilde{\boldsymbol{x}}}{\partial \tilde{\eta}} \right) \Bigg\}. \end{aligned} \tag{23}$$

The computation of the solution to equation (23) is identical to what was presented in section 2.2.3.

4. Numerical experiments

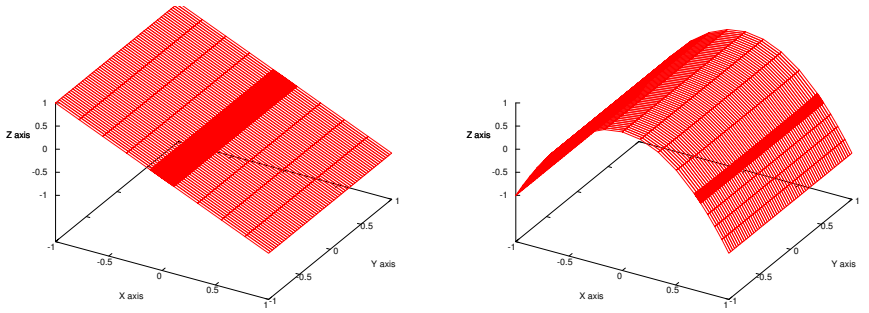
We now compute adaptive meshes for a selection of parametric surfaces using the methodology defined in section 3.2. We first present static meshing on four polynomial surfaces. Next, we evolve a mesh on a quadratic surface. Finally, we compute an adaptive mesh on two complex surfaces, a human face and a Möbius strip. In all examples, the discretization is identical to the one derived in section 2.2. In the first examples, the surface is assumed to be a graph. This leads to parameterizations of the simple form $\Phi(x, y) = (x, y, z(x, y))$. The main implication of this choice is that, for the sake of mesh movement computation, we can ignore the elevation $z(x, y)$. Since the parameterization Φ does not alter the other two coordinates, we have $\tilde{\boldsymbol{x}} = \boldsymbol{x}$. The elevation (and therefore the parameterization) still comes into play when evaluating the monitor function. In our last example, we compute an adapted mesh on a Möbius strip. This experiment demonstrates that our method is not restricted to graphs. As in section 2, all implementations are carried out using the finite element package deal.II [1,2].

4.1. Time-independent mesh-adaptivity on surfaces

We now construct adaptive meshes for four polynomial surfaces: a linear surface $z(x, y) = -y$, a quadratic surface $z(x, y) = 1 - 2x^2$, a cubic surface $z(x, y) = -x^3$ and a product surface $z(x, y) = xy$. For all four surfaces, we select the same time-independent monitor function,

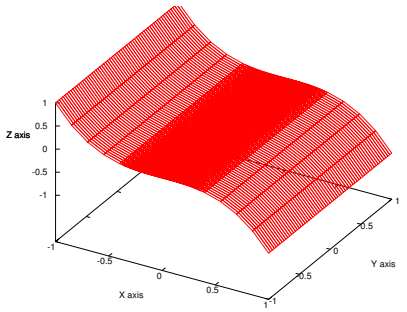
$$\omega(x, y, z) = 1000 \exp\left(-100 z^2\right) + 1.$$

For simplicity, we define the monitor function by its embedding representation, however, there is no technical difficulty to compute meshes when the monitor function is defined directly (and exclusively) on the surface. The computed meshes are plotted on Figure 2. In each case, the mesh density is greatest around the elevation $z = 0$, which agrees with the maximum value of the monitor function. Comparing the results for the different surfaces, we find that the greatest mesh densities are observed for the relatively steep linear and quadratic shapes. This result has an intuitive explanation: In moving mesh methods, the number of cells is fixed. Thus, we expect high mesh densities in regions where large monitor function values are spatially concentrated. (In this example, large monitor function values arise over a small region if the surface is steeply sloped around $z = 0$.) This example was computed on a 16×16 grid, using a time step-size of $\Delta t = 0.1$.

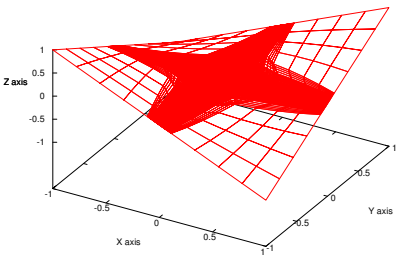


(a) linear surface $z(x, y) = -x$

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



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



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

Fig. 2: Steady-state meshes for a selection of polynomial surfaces. The smooth monitor function takes on its maximum value at $z = 0$. We observe that the greatest mesh density is obtained in a neighborhood of the horizontal plane $z = 0$.

4.2. Time-dependent mesh-adaptivity on surfaces

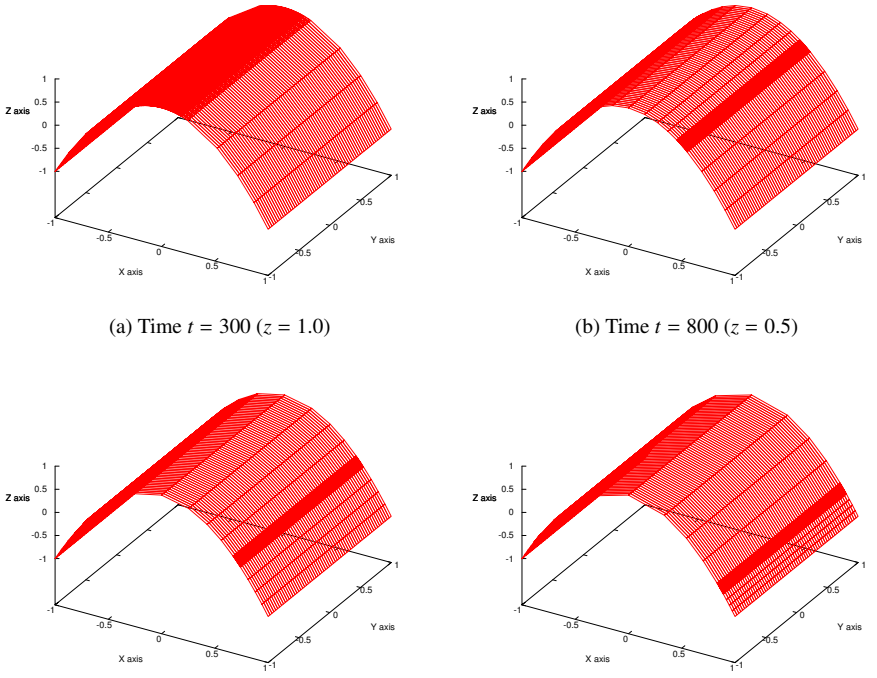
In the second example, we evolve a time-dependent monitor function

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

on the quadratic surface $z(x, y) = 1 - 2x^2$. The monitor function attains its maximal values on a horizontal plane that descends linearly with time. In an initial phase, the region of high mesh density appears at the top of the parabola, before splitting into two dense regions. The two regions of high mesh density then proceed down both branches synchronously. This splitting of the dense region and subsequent symmetrical evolution is handled by the method without any special treatment. This example was computed on a 16×16 grid, using a time step-size of $\Delta t = 0.1$.

4.3. Mesh-adaptivity on more complex surfaces

Our moving mesh method is versatile enough to be transposed easily onto more complex surfaces. Of particular interest is mesh adaptation on shapes defined by polygonal meshes. As a first illustration, consider mesh adaptation on the face of the human torso model provided courtesy of INRIA [17] by the AIM@SHAPE-VISIONAIR Shape Repository [12]. We use the parameterization $\Phi : [-1, 1]^2 \to M$ with $\Phi(x, y) = (0.055x, -0.005 + 0.055y, z)$, with the y-axis being defined by the line of the eyes, the x-axis oriented such that the face is facing the viewer when plotted in the (x, y)-plane and z describing the surface of the face. We specify our desired curvature-dependent mesh adaptivity by setting the monitor function equal to the root mean square value of the curvature in each cardinal direction. To emphasize the area around the right-eye, the monitor function is scaled by a large factor in the upper-right quarter of the face. Figure 4 shows the original and adapted meshes using our method. With this choice of monitor function, an increased mesh density is obtained for the curved regions bounding the right eye, to the detriment of the nose,



(a) Time $t = 300$ ($z = 1.0$)

(b) Time $t = 800$ ($z = 0.5$)

(c) Time $t = 1300$ ($z = 0.0$)

(d) Time $t = 1800$ ($z = -0.5$)

Fig. 3: Mesh movement on a parabola at various times t. The value of z that maximizes the monitor function ω is given in parentheses. We observe that the region of high mesh density descends linearly with time.

cheeks and left eye. Naturally, if other features are to be emphasized, a suitably modified monitor function should be specified. Our primary conclusion is that mesh adaptivity on a complex surface can be accomplished simply by modifying a planar MMPDE code. This example was computed on a 64×64 grid using a time step-size of $\Delta t = 0.1$.

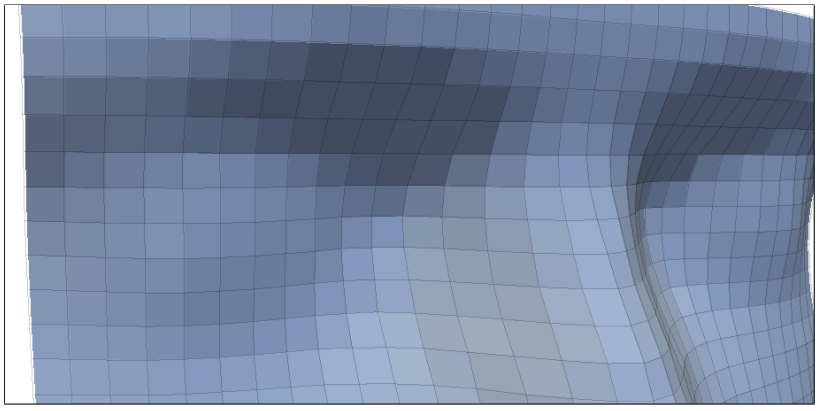
In our last application, we move a mesh on a Möbius strip. A Möbius strip is a non-orientable three-dimensional surface with only one side and one boundary. We use the parameterization $\Phi : [-1, 1]^2 \to M$ with $\Phi(u, v) = (x, y, z)$ and

$$\begin{cases} x(u,v) = \left(1 + \dfrac{v}{2}\cos\dfrac{(u+1)\pi}{2}\right)\cos(u+1)\pi, \\ y(u,v) = \left(1 + \dfrac{v}{2}\cos\dfrac{(u+1)\pi}{2}\right)\sin(u+1)\pi, \\ z(u,v) = \dfrac{v}{2}\sin\dfrac{(u+1)\pi}{2}. \end{cases}$$

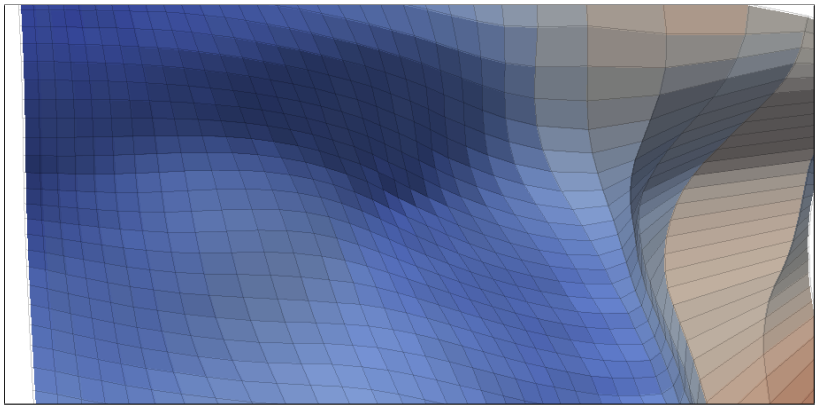
This defines a Möbius strip of width 1, with an inner circle of radius 1 centered at $(0, 0, 0)$. We use the monitor function $\omega(x, y, z) = 1 + 100\exp(-10z^2)$. This monitor function takes on its maximum value at the elevation $z = 0$, i.e., $v = 0$ or $u = \pm 1$ in the computational variables. Results for a 32×32 grid and a time step-size of $\Delta t = 0.1$ are shown in Figure 5. We observe that there is an increase in the mesh density towards the centre of the strip and that the maximum density occurs in the most vertical region. A similar effect was previously found in the results of Figure 2.

5. Conclusions

This paper considers the computation of adaptive grids on surfaces using MMPDEs. In section 2, a brief introduction to the theory and derivation of MMPDEs is given. This is followed by a derivation of the gradient flow equations



(a) Original Face



(b) Adapted Face

Fig. 4: Mesh adaptation on a human face based on a curvature-dependent monitor function.

for Winslow's adaptation functional. This section also gives a finite element formulation for the corresponding MMPDE in $\mathbb{R}^2$ and provides details on our deal.II implementation. An example for the evolution of a 2D adaptive grid is given to validate our code. Section 3 derives the gradient flow for Winslow's adaptation functional on general surfaces. This section also gives a simple method to compute solutions on surfaces that accept a continuously invertible parameterization. Our approach builds on top of the methods and code developed for the planar case. In section 4, adaptive meshes are computed on a variety of surfaces including a quadrilateral mesh representation of a human face. The results are promising.

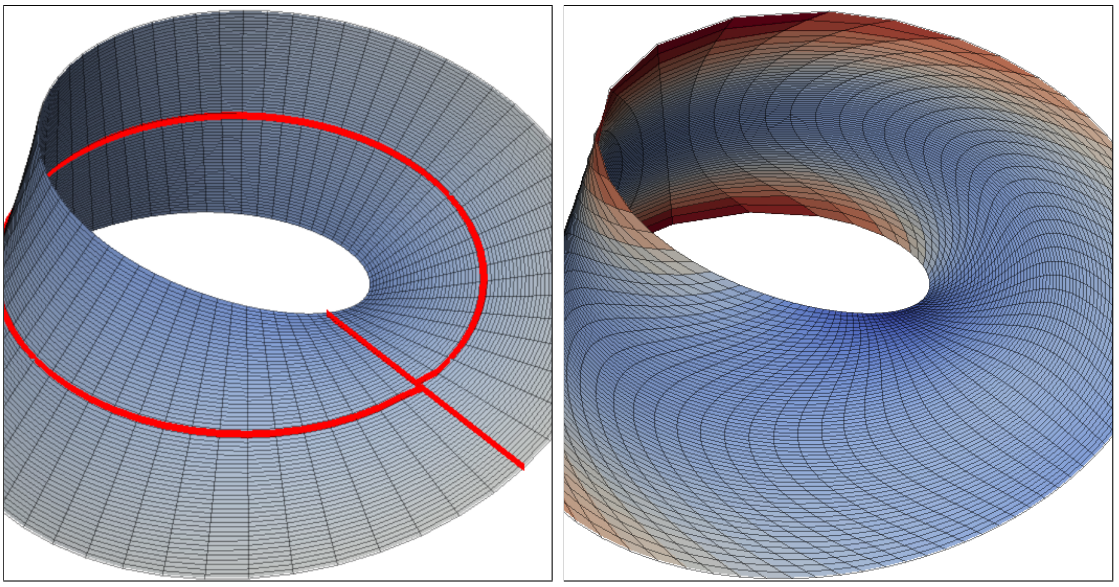
This research is to our knowledge the first to compute adaptive grids on surfaces using MMPDEs. The amount of work remaining is therefore vast. Interesting extensions include the design of fast parallel algorithms and methods for the direct solution of MMPDEs on general surfaces without parameterization. Also of great interest is the study of methods for the adaptive solution to PDEs defined on surfaces using MMPDEs.

Acknowledgements

This research was partially supported by NSERC Discovery Grants A8781 and 227823.

References

[1] W. Bangerth, R. Hartmann, G. Kanschat, deal.II – a General Purpose Object Oriented Finite Element Library, ACM Trans. Math. Softw. 33(4) (2007) 24/1–24/27.



(a) Original strip (b) Adapted strip

Fig. 5: Mesh adaptation on a Möbius strip using a monitor function that focuses the mesh at the elevation $z = 0$. The intersection of the plane $z = 0$ with the strip is drawn in red on the original strip.

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

[3] R. Barreira, C.M. Elliott, A. Madzvamuse, The surface finite element method for pattern formation on evolving biological surfaces, J. of Math. Biology. 63(6) (2011) 1095–1119.

[4] C.J. Budd, W. Huang, R.D. Russell, Moving Mesh Methods for Problems with Blow-Up, SIAM J. Sci. Comput. 17(2) (1996) 305-327.

[5] C.J. Budd, S. Chen, R.D. Russell, New self-similar solutions of the nonlinear Schrdinger equation with moving mesh computations, J. Comput. Phys. 152(2) (1999) 756–789.

[6] W. Cao, personal communication, 2011.

[7] W. Cao, W. Huang, R.D. Russell, An r-Adaptive Finite Element Method Based upon Moving Mesh PDEs, J. Comput. Phys. 149(2) (1999) 221–244.

[8] Y. Canzani, Analysis on manifolds via the Laplacian, Lecture Notes, Harvard University, `http://www.math.harvard.edu/~canzani/math253.html`.

[9] C. Cowan, The Cahn-Hilliard Equation as a Gradient Flow, M.Sc. Thesis, Simon Fraser University, 2005.

[10] B. Crestel, Moving Meshes on General Surfaces, M.Sc. Thesis, Simon Fraser University, 2011.

[11] C. de Boor, Good approximation by splines with variable knots. In A. Meir and A. Sharma, editors, Spline Functions and Approximation Theory, pages 57-73, Birkhäuser Verlag, Basel und Stuttgart, 1973.

[12] Digital Shape Workbench – Shape Repository. `http://visionair.ge.imati.cnr.it/ontologies/shapes/view.jsp?id=651-Human_torso#` (Oct. 2007)

[13] L.C. Evans, Partial Differential Equations, second ed., American Mathematical Society, Providence RI, 1998.

[14] W. Huang, R.D. Russell, Adaptive mesh movement – the MMPDE approach and its applications, J. Comput. Appl. Math. 128(1-2) (2001) 383–398.

[15] W. Huang, R.D. Russell, Adaptive Moving Mesh Methods, Springer-Verlag, New York, 2011.

[16] W. Kühnel, Differential geometry: curves - surfaces - manifolds, American Mathematical Society, 2006.

[17] W.-C. Li, N. Ray, B. Levy, Automatic and Interactive Mesh to T-Spline Conversion, 4th Eurographics Symposium on Geometry Processing - SGP 2006, Jun 2006, Sardinia/Italy, 2006.

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

[19] L. Tian, C.B. Macdonald, S.J. Ruuth. Segmentation on surfaces with the Closest Point Method, Proc. ICIP09, Int. Conf. on Image Processing, Cairo, Egypt, November 7-11, 2009.

[20] P. Tian, F. Qiu, H. Zhang, Y. Yang. Phase separation patterns for diblock copolymers on spherical surfaces: a finite volume method, Phys Rev E Stat Nonlin Soft Matter Phys, 72 (2005).

[21] J.-J. Xu, Z. Li, J. Lowengrub, H. Zhao. A Level-set Method for Interfacial Flows with Surfactant, J. Comput. Phys., 212(2) (2006) 590–616.

[22] X. Xu, W. Huang, R.D. Russell, J.F. Williams, Convergence of de Boor's algorithm for the generation of equidistributing meshes, IMA J. Numer. Anal. 31(2) (2011) 580–596.