ABSTRACT

Title of dissertation: PARAMETRIC AFEM FOR GEOMETRIC EVOLUTION EQUATIONS AND COUPLED FLUID-MEMBRANE INTERACTION

Miguel Sebastian Pauletti
Doctor of Philosophy, 2008

Dissertation directed by: Professor Ricardo H. Nochetto
Department of Mathematics

When lipid molecules are immersed in aqueous environment at a proper concentration they spontaneously aggregate into a bilayer or membrane that forms an encapsulating bag called vesicle. This phenomenon is of interest in biophysics because lipid membranes are ubiquitous in biological systems, and an understanding of vesicles provides an important element to understand real cells. Also lately there has been a lot of activity when different types of lipids are used in the membrane. Doing mathematics in such a complex physical phenomena, as most problems coming from the bio-world, involves cyclic iterations of: modeling and analysis, design of a solving method, its implementation, and validation of the numerical results. In this thesis, motivated by the modeling and simulation of biomembrane shape and behavior, new techniques and tools are developed that allow us to handle large deformations of surface flows and fluid-structure interaction problems using the finite element method (FEM). Most simulations reported in the literature using this method are academic and do not involve large deformation. One of the questions

this work is able to address is whether the method can be successfully applied to more realistic applications. The quick answer is not without additional crucial ingredients. To make the method work it is necessary to develop a synergetic set of tools and a proper way for them to interact with each other. They include space refinement/coarsening, smoothing and time adaptivity. Also a method to impose isoperimetric constraints to machine precision is developed. Another use of the computational tools developed for the parametric method is mesh generation. A mesh generation code is developed that has its own unique features not available elsewhere as for example the generation of two and three dimensional meshes compatible for bisection refinement with an underlying coarse macro mesh. A number of interesting simulations using the methods and tools are presented. The simulations are meant first to examine the effect of the various computational tools developed. But also they serve to investigate the nonlinear dynamics under large deformations and discover some illuminating similarities and differences for geometric and coupled membrane-fluid problems.

PARAMETRIC AFEM FOR GEOMETRIC EVOLUTION EQUATION AND COUPLED FLUID-MEMBRANE INTERACTION

by

Miguel Sebastian Pauletti

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2008

Advisory Committee:
Ricardo H. Nochetto, Chair/Advisor
Radu Balan
Wolfgang Losert
Dionisios Margetis
John E. Osborn

Acknowledgments

First of all I would like to thank my adviser who was always available to help me not only academically but also as a friend when I needed it the most. I would also like to thank the following people. The oral exam committee members: professors Radu Balan, Wolfgang Losert, Dionisios Margetis and John E. Osborn for spending their valuable time to read the manuscript for the wise suggestions for improvements and for the typos detection. Andrea Bonito who read the draft notes that wanted to be the thesis giving me valuable suggestions and for the helpful discussions we maintained. Pedro Morin who during a visit to the IMAL (Instituto de Matematica Aplicada del Litoral) spent his valuable time introducing me to the software library ALBERTA. Daniel Koster who during a visit to the University of Maryland provided us with the not yet released ALBERTA 2.0 library with several code examples and oral explanations of the new undocumented features. Professor J. Yorke who allowed us to use one of his computer for some of our most demanding simulations. Professor Hugo Aimar who was of great help to make my graduate studies in the US possible and taught me what devotion to mathematics is with his life example. Soledad, my wife, whose effort and support during this time is incommensurable. Finally thank you Chiara for teaching me how to type at 3AM with a single hand.

Table of Contents

List of Abbreviations

$DF = (\frac{\partial F_i}{\partial x_j})_{ij}$	Jacobian Matrix
Γ	A surface or boundary
$\mathcal{P}^k$	Space of polynomial of degree $\leq k$
ω_i	Star at node i
h_K	Diameter of K
K	Finite Element,Simplex,Triangle
$\mathcal{T}$	Mesh Triangulation
meas	Measure of a set (area, volume, etc)
FEM	Finite Element Method
AFEM	Adaptive Finite Element Method
CG	Conjugate Gradient Method
PCG	Preconditioned Conjugate Gradient Method
MINRES	Minimal Residual Method
GMRES	Generalized Minimal Residual Method
$:=$	is defined by
$\mathcal{N}$	degrees of freedom or set of vertices (depend on context)
d	$d+1$ is the embedding space dimension
$\mathbf{Id}_\Gamma$	Identity funtion on the set Γ
$\vec{\mathbf{C}}$	Matrix whose components are matrices
$\mathbf{\Sigma}$	Cauchy Stress Tensor
q	Surface area element
$\mathbf{G}$	Matrix of the first fundamental form

Chapter 1

Introduction

When lipid molecules are immersed in aqueous environment at a proper concentration they spontaneously aggregate into a bilayer or membrane that forms an encapsulating bag called vesicle. This phenomenon is of interest in biophysics because lipid membranes are ubiquitous in biological systems, and an understanding of vesicles provides an important element to understand real cells. Also lately there has been a lot of activity when different types of lipids are used in the membrane. Doing mathematics in such a complex physical phenomena, as most problems coming from the bio-world, involves cyclic iterations of:

- modeling and analysis,
- design of a solving method,
- its implementation, and
- validation of the numerical results.

In this thesis, motivated by the modeling and simulation of biomembrane shape and behavior, new techniques and tools are developed that allow us to handle large deformations of surface flows and fluid-structure interaction problems using the finite element method (FEM). The type of FEM can be traced back to Dziuk [Dzi91] for the mean curvature flow. The method applies to evolutionary surfaces whose

flow can be written in Eulerian coordinates. But at the discrete level the method becomes parametric or Lagrangian in the sense that the new position of the mesh is a function defined in a reference or parameter domain. However, on a time iteration setting, the reference domain is the domain at the previous time step. Therefore the reference domain is changing at every time step and is close to an Eulerian domain. The advantage of such a method is that any standard finite element code can be adapted to handle it without much change. This is mostly due to the fact that a finite element code does all computations on a master element and the surface gradient becomes the master element gradient properly mapped (see [Dzi88] for details). The drawback is that as the computational domain is changing in time the underlying mesh can get easily distorted, to the extent of making the domain useless for computations and the method unusable. Reasons for the distortion are the initial mesh, the type of flow and the time step. Most simulations reported in the literature using this method are academic and do not involve large deformation. One of the questions this work is able to address is whether the method can be successfully applied to more realistic applications. The quick answer is not without additional crucial ingredients. To make the method work it is necessary to develop a synergetic set of tools and a proper way for them to interact with each other. They include space refinement/coarsening, smoothing and time adaptivity. Also a method to impose isoperimetric constraints to machine precision is developed.

Having these tools at hand allows us to implement geometric models for biomembranes in two and three dimensions and obtain simulations unseen before. These are the first reported simulations of Willmore flow with constraints using a

parametric piecewise linear and piecewise quadratic finite element method.

One step further in the degree of complexity is the extension of the geometric method to deal with the coupling of a free boundary surface with a bulk newtonian fluid. This in turn allows us to simulate the dynamics of biomembranes in two and three dimensions. Three dimensions non-axisymmetric simulations in this area have not been reported in the literature using this or any other method.

It is important to emphasize the computational cost of this approach. Small but still interesting three dimensional fluid simulations are obtained in a single processor computer. For the geometric three dimensional model a laptop with a 1.2GHz Celeron processor, 512MB of RAM takes about an hour to obtain relevant simulations (about 120K degrees of freedom). This computational appeal is a combination of the parametric method, the space and time adaptivity and to a lesser extend the fact that the code is written in the C language and uses state of the art library solvers.

Another use of the computational tools developed for the parametric method is mesh generation. A mesh generation code is developed that has its own unique features not available elsewhere as for example the generation of two and three dimensional meshes compatible for bisection refinement with an underlying coarse macro mesh.

1.1 Thesis Outline and Contributions

The outline of the reminder of this thesis is a follows

- In Chapter 2 we describe the physical background for the biomembrane models. This is one of the most important applications of the numerical methods developed here.

- In Chapter 3 we recall basic concepts and results from differential geometry, shape differential calculus and continuum mechanics. We do this from a unified point of view both in the notation as well as in the concepts.

- In Chapter 4 we present a set of applications that can be implemented within the computational framework of Chapter 7. The applications are divided into two groups (Sections 4.2 and 4.3). The first one is concerned with geometric problems. The second group focuses on models to describe a fluid-membrane interaction. The different problems are treated from a unified point of view and a link is made on how the second group builds on the first.

- In Chapter 5 we provide the basics of the finite element method, with emphasis on how it extends to parametric surfaces, and present the evolving parametric method. In section 5.2 we provide the basic tools to work with finite elements on a surface that we apply in the following sections 5.3, 5.4 and 5.5 to obtain interpolation results, discrete formulas for curvature and a priori estimates for the Laplace-Beltrami operator. In Section 5.6 we present a result for surface quadratic isoparametric elements that will justify some methods of Chapter 7. We finish the chapter with a section where we present the parametric finite element method for geometric evolution equations.

- In Chapter 6 we present space and time discretizations of the continuous problems described in chapter 4. First we treat the geometric case. Doing this provides helpful insight as to how to deal with the fluid-membrane schemes treated later in the chapter. We discuss the benefits and drawbacks of different possibilities and information about solving the discrete systems.

- In Chapter 7 we address the computational issues related to the implementation of the parametric AFEM for geometric evolution equations and coupled fluid-membrane problems. A set of computational tools including: space and time adaptivity; mesh enhancement and discrete constraints implementation is presented. These tools are crucial to successfully use the parametric FEM. In Section 7.1 we discuss the counterintuitive effect that a mismatch of the finite element spaces may have on problems involving curvature. In Section 7.2 we propose a suitable remedy. Also here we deal with the issue of geometric adaptivity as means of describing the surface accurately with the minimal number of degrees of freedom. First we propose a geometric estimator based on the pointwise error. Then we define a geometric compatibility condition that is key for the adaptivity not to deteriorate the flow. Based on this condition we provide a novel refinement procedure together with a theorem showing the benefits of it. In Section 7.3 we present a novel method to compute the solutions of discrete systems with isoperimetric constraints. In Section 7.4 we deal with the issue of mesh improvement. When a parametric FEM is used to discretize a geometric evolution equation it will create a discrete flow of

the mesh. Even if the initial mesh has a perfect quality, as it moves with the flow it will get distorted: the larger the overall domain deformation the larger the mesh deterioration. We present an optimization method novel in many aspects that improves the mesh quality, preserves the shape of its boundary, maintains the local mesh size, and produces negligible changes to the finite element functions defined on the mesh. Different cases are analyzed depending on the type of domain and the mesh degree. In Section 7.5 we describe a novel hybrid affine-quadratic approach to the surface/boundary isoparametric elements. The idea is to keep the quadratic element not far from its affine support, but still allow it to have the characteristic rounded shape coming from the quadratic bubble. Then the affine techniques for mesh improvement and time-step adaptivity can be used on quadratic meshes. In Section 7.6 a geometric timestep control is discussed. In general nonlinear time dependent fourth-order problems present a highly varying time scale during its evolution. Then a timestep control is indispensable to for computational success. Finally in Section 7.7 we present the general parametric AFEM algorithm with the incorporation of the computational tools previously developed in the chapter. The order in which the tools are applied is important to potentiate themselves in a synergic way.

- In Chapter 8 we present a number of interesting simulations using the methods and tools of chapter 7 to solve the problems discussed in chapter 4 with the schemes of chapter 6. The simulations are meant first to examine the effect of

the various computational tools developed. They also serve to investigate the nonlinear dynamics under large deformations and discover some illuminating similarities and differences with an without fluid.

- In Chapter 9 we present another application of the computational tools to the problem of mesh generation.

Our specific contributions are as follows:

- In Chapter 3 we extend the differential geometry matrix notation approach by K. Mekchay [Mek05], originally developed for piecewise linear affine surfaces to piecewise polynomial ones. The differential geometry, shape differential calculus and continuum mechanics are treated from a unified approach to concepts and notations pointing out the links.

- In Chapter 4 we obtain a non-dimensional formulation for the coupled Willmore and Fluid-Biomembrane model. Also here we provide a proof for the existence of multipliers.

- In Chapter 5 the treatment of the a priori estimate for the Laplace-Beltrami operator is different from the existing one done in [Dzi88] or [Dem] in the sense that we do not use the distance function and different from [Mek05] because we do not use macro elements and it is done for any order isoparametric representation of the surface. This proof will be crucial in the proof for the novel result of Section 7.2.2. We extend interpolation results from flat elements to surfaces. And we show a result to control surface quadratic isoparametric

elements by its affine bases. This is a generalization of a result for the flat case due to Ciarlet and Raviart.

- In Chapter 6 we analyze the different reasonable choices regarding the time discretization of the Willmore flow that can be made for the explicit treatment of certain terms. These new choices appear as a consequense of using quadratic isoparametric element (not explored before for this problem). Also we propose some methods to obtain an initial reasonable approximation to mean curvature which is not addressed in previous works. Even though the numerical methods presented are not new their application to this problem (constrained Willmore and fluid Willmore) is.

- In Chapter 7 we present a discussion about the counterintuitive effect that a mismatch of the finite element spaces may have on problems involving curvature that is not reported in the literature. In Section 7.2 we propose a geometric estimator for the pointwise error with a novel computational formula using the second fundamental form. A new geometric compatibility condition is defined that is key for the adaptivity not to deteriorate the flow. A novel refinement procedure together with a theorem showing the benefits of it is provided. In Section 7.3 we present a novel method to compute the solutions of discrete systems with isoperimetric constraints to machine precision. In Section 7.4 we deal with the issue of mesh improvement. We present an optimization method novel in many aspects (finite element function interpolation, definition of surface quality metric and optimization method to work

with a surface star) that improves the mesh quality, preserves the shape of its boundary, maintains the local meshsize, and produces negligible changes to the finite element functions defined on the mesh. In Section 7.5 we describe a novel hybrid affine-quadratic approach to the surface/boundary isoparametric elements. In Section 7.6 a geometric timestep control is discussed; the idea of using the element quality in its definition is new.

- In Chapter 8 many simulations are obtained for the first time using this method, and others for the first time with this or any other method.

- In Chapter 9, a mesh generation algorithm is presented that allows one to generate non trivial meshes in two and three dimensions which are compatible with bisection refinement and have and underlying coarse mesh. There is no other mesh generator with these features.

1.2 Notation

In this work $\mathbb{R}$ is the set of real numbers, $\mathbb{N}$ the set of natural numbers and $\mathbb{N}_0 = \mathbb{N} \cup \{0\}$. The embedding dimension will be denoted by $d+1$. For a set $U \subset \mathbb{R}^{d+1}$, $\bar{U}$ denotes the closure and ∂U its boundary. We use meas(U) to denote its measure (volume, area, perimeter, etc depending on the context). Given the set U its *convex hull* is given by $\text{conv}(U) := \cap_{U \subset V} V$. And the diameter of U is $\text{diam}(U) = \sup\{|x-y| : x, y \in U\}$.

We usually use Ω to denote a domain in $\mathbb{R}^{d+1}$ and Γ to denote a surface. We use **bold type** for vectors and tensors. and $\mathbb{BLACKBOARDTYPE}$ for finite

element spaces. For convenience we provide a table of symbols and notation on page vi.

The concept of derivative as the best linear approximation allows to unify concepts from differential geometry and continuous mechanics.

Definition 1.2.1 (Differentiable function: Derivative)**.** Let $\mathcal{U}$ and $\mathcal{W}$ be Banach spaces, let $\mathcal{D}$ be an open subset of $\mathcal{U}$, and let

$$\boldsymbol{g} : \mathcal{D} \to \mathcal{W}.$$

We say that $\boldsymbol{g}$ is differentiable at $\boldsymbol{x}$ if there exists a linear transformation

$$D\boldsymbol{g}(\boldsymbol{x}) : \mathcal{U} \to \mathcal{W}$$

such that

$$\boldsymbol{g}(\boldsymbol{x} + \boldsymbol{u}) = \boldsymbol{g}(\boldsymbol{x}) + D\boldsymbol{g}(\boldsymbol{x})[\boldsymbol{u}] + o(|\boldsymbol{u}|)$$

as $\boldsymbol{u} \to 0$. $D\boldsymbol{g}(\boldsymbol{x})$ is called the derivative of $\boldsymbol{g}$ at $\boldsymbol{x}$.

1.3 Function Spaces

Definition 1.3.1 (Polynomials)**.** For a multi-index $\boldsymbol{\alpha} \in \mathbb{N}_0^{d+1}$, we define $|\boldsymbol{\alpha}| = \sum \alpha_i$ and $\boldsymbol{x}^{\boldsymbol{\alpha}} = \Pi x_i^{\alpha_i}$ for $x \in \mathbb{R}^{d+1}$. For $k \in \mathbb{N}_0$ we define

$$\mathcal{P}^k(\Omega) := \left\{ p : \Omega \to \mathbb{R} \ : p(\boldsymbol{x}) = \Sigma_{|\boldsymbol{\alpha}| \leq k} c_{\boldsymbol{\alpha}} \boldsymbol{x}^{\boldsymbol{\alpha}}, c_{\boldsymbol{\alpha}} \in \mathbb{R} \right\}$$

Definition 1.3.2 (Spaces of Smooth Functions)**.** Let $\boldsymbol{\alpha} \in \mathbb{N}_0^{d+1}$. If $f : \Omega \to \mathbb{R}$ is an $|\boldsymbol{\alpha}|$ times continuously differentiable function, then

$$D^{\boldsymbol{\alpha}} f := \frac{\partial^{|\boldsymbol{\alpha}|}}{\partial x_1^{\alpha_1} \dots \partial x_{d+1}^{\alpha_{d+1}}}$$

For $m \in \mathbb{N}_0$ we define the spaces of continuous and differentiable functions

$$C^m(\Omega) := \{f : \Omega \to \mathbb{R} : D^{\boldsymbol{\alpha}} f \text{ is continuous }, |\boldsymbol{\alpha}| \leq m\}.$$

We will say the f is smooth to indicate that it is in C^m for whatever m the context requires.

Definition 1.3.3 (Lebesgue Spaces)**.** Let $p \in [1, \infty]$ we define the Lebesgue spaces

$$L^p(\Omega) := \{f : \Omega \to \mathbb{R} \ : f \text{ is measurable, } \|f\|_{L^p} < \infty\}$$

where $\|f\|_{L^p} = \left(\int_\Omega |f|^p\right)^{1/p}$ if $p < \infty$ and $\|f\|_{L^\infty} = \text{ess sup}_\Omega |f|$.

Definition 1.3.4 (Sobolev Spaces)**.** Let $p \in [1, \infty]$ and $m \in \mathbb{N}$. We define the Sobolev spaces

$$W_p^m(\Omega) := \{f \in L^p(\Omega) : D^{\boldsymbol{\alpha}} f \in L^p(\Omega) \ \forall \ |\boldsymbol{\alpha}| \leq m\}$$

where $D^{\boldsymbol{\alpha}} f$ denotes the weak derivative of f and $\|f\|_{W_p^m} := \Sigma_{|\boldsymbol{\alpha}| \leq m} \|D^{\boldsymbol{\alpha}} f\|_{L^p}$.

We use the standard notation $H^m(\Omega) := W_2^m(\Omega)$.

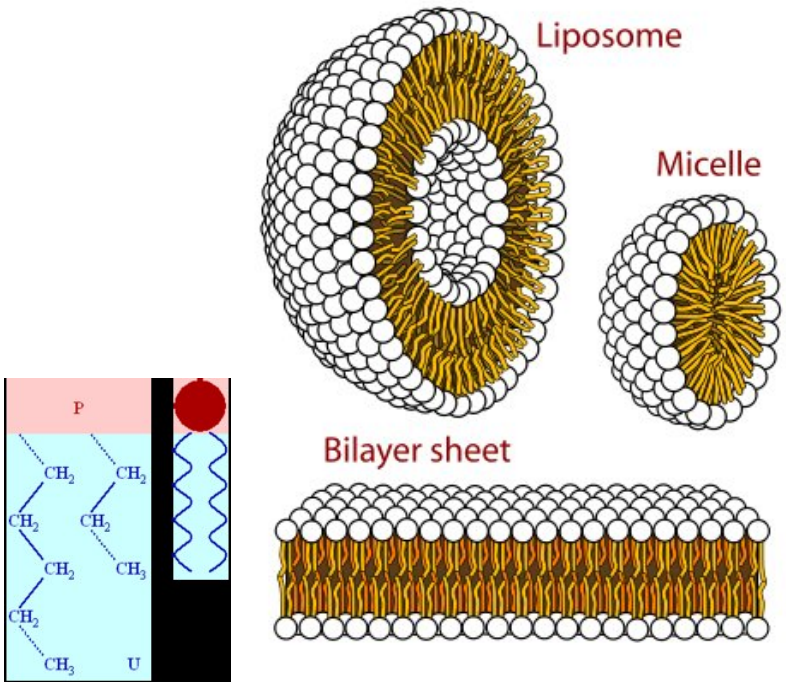
Chapter 2

Biomembranes: Physical Background

This Chapter gives a quick survey of the physical properties and reasonable models for biomembranes under a special set of conditions. This is not a chemical or physical discussion but rather a mathematical idealization trying to account for the most relevant properties of biomembranes. The biomembrane shape and dynamics is one of the most important applications that we have in mind for our numerical methods.

When lipid molecules are immersed in aqueous environment at a proper concentration and temperature they spontaneously aggregate into a bilayer or membrane that forms an encapsulating bag called vesicle. This happens because lipids consist of a hydrophilic head group and one or more hydrophobic hydrocarbon tails. Such a configuration allows the tails to be isolated from water, thus reducing the hydrophobic effect (see figure 2.1). This phenomenon is of interest in biology and biophysics because lipid membranes are ubiquitous in biological systems, and an understanding of vesicles provides an important element to understand real cells. We are interested in the case where the thickness of the membranes is negligible compared to the size of the vesicle (about three orders of magnitude). The elastic behavior under large deformations and the dynamics of such deformations are poorly understood.

Canhan and Helfrich [Can70, Hel73] over 35 years ago, were the first to intro-



(a) Lipid Molecule (b) Aqueous Configurations

Figure 2.1: Figure 1(a) shows a typical lipid molecule. Figure 1(b) depicts some aqueous configuration they like to take. We are concerned with the liposomes configuration when the thickness of the membrane is negligible compared to the size of the liposome. Taken from Wikipedia Commons.

duce a model for the equilibrium shape of vesicles where the *bending elasticity* or curvature energy had to be minimized. Jenkins in 1977 arrived to the same model

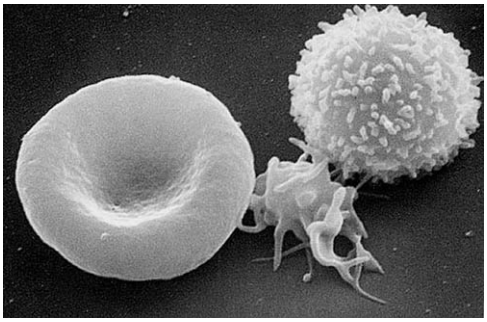


Figure 2.2: A three-dimensional ultra-structural image analysis of a T-lymphocyte (right), a platelet (center) and a red blood cell (left), using a Hitachi S-570 scanning with a super-duper electron microscope (SEM) equipped with a GW Backscatter Detector. Author: Electron Microscopy Facility at The National Cancer Institute at Frederick (NCI-Frederick). (Taken from Wikipedia Commons).

starting from continuum mechanical principles [Jen77a]; some later papers on the subject are [Ste03, HZE07]. An equilibrium shape model is important for two reasons. First, if the goal is to predict stationary shapes seen in the lab then this is the required model. The second reason is to build a necessary step of the ladder to reach more complex models, including its dynamics.

The next step consists of describing the dynamics of a vesicle. This is relevant when we are interested not only in the final shape but also in how the membrane evolves to that shape. We need to say here that there is no clear agreement as to what a good model is (if any is at all). But it seems to be consensus that a coupling between the membrane and the fluid is key in this respect [Sei97]. The analysis of interacting fluid-structure problems has been the subject of research since the late nineteenth century but only recently there are some results about local existence and

uniqueness of solutions when the elastic solid is a linear Kirchhoff elastic material [CS05] and when is endowed of a bending energy [CCS07]. All this results are for short time and assume high regularity of the membrane.

Below we briefly describe the membrane and fluid models. The mathematical equations are subject of chapter 4

2.1 Membrane Model

The structure of lipid membranes is that of a two dimensional, oriented, incompressible and viscous fluid [Jen77a]. We can find papers that starting from a pure phenomenological approach [Hel73, Can70] or a rigorous continuum mechanical one [Jen77a, Ste99, HZE07], agree that the membrane is endowed with a bending or elastic energy. The simplest form this energy can take is

$$\kappa \int_\Gamma h^2 + \kappa_G \int_\Gamma k, \tag{2.1}$$

where h and k are the mean and Gaussian curvature respectively; and κ and κ_G are the constant bending coefficients. In the case of a closed surface that does not change topology the above energy becomes (up to a constant and a scaling) the "Willmore" energy defined by (see [Wil93])

$$W(\Gamma) := \int_\Gamma h^2. \tag{2.2}$$

This is a consequence of the Gauss-Bonnet theorem which states that $\int_\Gamma k$ is a topological invariant (see [MR05, Section 8.5]). The bending coefficient κ is hard to determine experimentally. However, for lipid membranes (see Seifert [Sei97]), the

typical range is

$$\kappa \approx 2.10^{-20} - 2.10^{-19} \quad \left[kg\frac{m^2}{s}\right].$$

The combined effect of the bending elasticity with the surface and volume constraints generates a great variety of non-spherical shapes, in contrast to the characteristic spherical equilibrium shapes of simple liquids which are governed by isotropic surface tension.

If the temperature and osmotic pressure do not change, a convenient and reasonable simplification can be done by assuming that the enclosed volume and surface area are conserved. The former is a consequence of the assumed impermeability of the membrane. For the latter the fix number of molecules in the membrane ensures a fixed internal area because stretching or compressing the membrane involves much larger energies than the cost of bending deformations. Refer to [ES80, Sei97, SBR$^+$03] for more precision.

The area and volume constraints are good approximations for most cases. Still we should bear in mind that under certain circumstances if the constraints are imposed strictly the model is wrong. For example, in the lab spherical vesicles are deformed into some other shapes by means of laser tweezers. Having strict area and volume constraints makes this impossible to happen.

The local curvature energy model (2.1) is the basic building block. On top of this, some additions and modification have been proposed. Some variations of curvature model are:

- Spontaneous curvature model

- Bilayer couple model
- Area difference-elasticity model

A current area of active research is coexisting fluid domains. Membranes formed from multiple lipid components can laterally separate into coexisting liquid phases, or domains, with distinct compositions. Models for coupling the curvature and composition have been flourishing after the work of Baumgart et al. [BHW03].

In this work we consider the local curvature energy model with isoperimetric area and volume constraints. Describing the membrane by quantities all defined on the surface (energy, area and volume) allows us to model certain aspects of it, like the equilibrium shape, as a geometric evolution equation (see section 4.2). For other aspects one needs to take into account the dynamics and in particular the inertial and frictional effects provided by the bulk fluid. Obviously, the second approach is more complete than the former but also more expensive and tricky computationally.

2.2 Fluid Model

The fluid embedding the membrane is quite complex. However as a first approximation we assume it to be viscous, incompressible and homogeneous. Therefore the Navier-Stokes equation can be used to model it. The Reynolds number of a fluid, being the ratio between the inertial forces and the viscous forces, is defined by

$$Re = \frac{\rho V L}{\mu},$$

where ρ is the density of the fluid, V and L the characteristic speed and length respectively, and μ the dynamic fluid viscosity. Typical velocity and length for the

experiments considered are

$$V = 10^{-6}\ [m/s], \qquad L = 10^{-5}\ [m],$$

so that $Re \approx 10^{-8} \ll 1$ for a water like bulk fluid. Therefore, as proposed in [Sei97], only Stokes equation needs to be considered. The effect of the membrane appears as a immerse boundary force exerted by a massless object on the fluid. This force is given by the variational derivative of some free membrane energy. The membrane is transported by the fluid (adherence or no slip condition).

Chapter 3

Preliminaries

In this Chapter we summarize some concepts and results from differential geometry, shape differential calculus and continuum mechanics. We do this from a unified point of view both in the notational as well as in the concepts. The natural language for geometric evolution equations (section 4.2) or any problem where curvature plays an important role is the one of differential geometry. Depending on the way used to represent the surface one obtains different expressions for the quantities of interest. The very elegant and also abstract language of differential manifolds becomes unfriendly to do finite element computations. In section 3.1 we recall basic concepts and results from differential geometry. It consist of a mixture of several references adapted to the necessity and requirements of the present work; in particular something not standard is the use of matrix notation. A detailed treatment can be found in [dC76, Gig06, GT83]. In section 3.2 we describe some useful tools from the shape differential calculus in the spirit of [SZ92]. In section 3.3 we introduce some concepts and notation from continuum mechanics; a full treatment can be found in [Ant95, Gur81].

3.1 Differential Geometry

Intuitively a surface is locally a piece of plane smoothly bent.

Definition 3.1.1 (Hyper-Surface)**.** A set $\Gamma \subset \mathbb{R}^{d+1}$ is a hyper-surface of dimension d if, for each $q \in \Gamma$, there exist an open set $U \in \mathbb{R}^d$, an open neighborhood V of q in Γ, and a differentiable map $\boldsymbol{x} : U \to \mathbb{R}^{d+1}$ such that

1. $\boldsymbol{x}(U) = V$.
2. $\boldsymbol{x} : U \to V$ is a homeomorphism.
3. $D\boldsymbol{x}(q) : \mathbb{R}^d \to \mathbb{R}^{d+1}$ is injective for all $q \in U$.

A function $f : \Gamma \to \mathbb{R}$ is *differentiable* if for any parametrization $\boldsymbol{x} : U \to \Gamma$, the composition $f \circ \boldsymbol{x}$ is differentiable.

Definition 3.1.2 (Tangent Plane $T_q\Gamma$)**.** Given a surface Γ and $q \in \Gamma$, the tangent plane $T_q\Gamma$ at q is the set of all vectors $\boldsymbol{t}$ such that there exists a curve $\alpha : (-\epsilon, \epsilon) \to \Gamma$ with $\epsilon > 0$, $\alpha(0) = q$ and $\alpha'(0) = \boldsymbol{t}$.

It is easy to see that $T_q\Gamma$ is the range of $D\boldsymbol{x}(q)$.

Definition 3.1.3 (Normal vector $\boldsymbol{\nu}$)**.** Given a surface Γ and $q \in \Gamma$, a vector $\boldsymbol{\nu}$ is a normal vector at q if $\boldsymbol{\nu} \in (T_q\Gamma)^{\perp}$.

Definition 3.1.4 (Orientable)**.** A surface Γ is orientable if there is a continuous function $\boldsymbol{\nu} : \Gamma \to \mathbb{S}^d$ with $|\boldsymbol{\nu}| = 1$ and $\boldsymbol{\nu}(q)$ normal to $T_q\Gamma$ for each $q \in \Gamma$.

The useful result stated next about orientability, whose proof can be found in [Sam69], will allow us to simplify some required hypothesis later.

Theorem 3.1.1. *Every compact surface in* $\mathbb{R}^3$ *is orientable.*

The concept of derivative as the best linear approximation can be extended to a function defined on a surface by means of the tangent plane.

Definition 3.1.5 (Differential of a Function)**.** Let Γ be a surface and $f : \Gamma \to \mathbb{R}^m$ a differentiable vector valued function. For a point $q \in \Gamma$, we define the differential of f at q, which we denote by $Df(q)$, in the following manner: Given $\boldsymbol{t} \in T_q\Gamma$, we choose a curve $\alpha : (-\epsilon, \epsilon) \to \Gamma$ such that $\alpha(0) = q$ and $\alpha'(0) = \boldsymbol{t}$, and then $Df(q)\boldsymbol{t} = (f \circ \alpha)'(0)$

Remark 3.1.1. $Df(q) : T_q\Gamma \to \mathbb{R}^m$ is a well defined linear map, i.e. independent of the chosen curve (cf. definition 1.2.1).

Remark 3.1.2. The definition of differential can be extended in a similar way to mappings $f : \Gamma_1 \to \Gamma_2$ between surfaces.

Remark 3.1.3. The chain rule and inverse function theorem hold for differentials.

When a surface is given by local coordinates as in definition 3.1.1 we say that we have a parametric representation of it, U being the space of parameters. There are other ways to represent a surface that can be shown to satisfy definition 3.1.1. Two important examples are the level set and graph representations.

For computations on surfaces it is convenient to use matrices to describe geometric quantities. The following notational ideas were suggested by the works [Mek05, DD07]. Let $(\boldsymbol{x}, \hat{K})$ be a local parametric representation of a surface Γ, $\boldsymbol{x} : \hat{K} \to K \subset \Gamma$. Then we define $\mathbf{T} = D\boldsymbol{x} \in \mathcal{M}_{(d+1)\times d}(\hat{K})$ to be the Jacobian matrix of $\boldsymbol{x}$ and $\mathbf{G} = \mathbf{T}^\mathsf{T}\mathbf{T}$ the *first fundamental form*. It easy to see that $\mathbf{G}$ is symmetric and positive definite because $\mathbf{T}$ is full rank; then we define $\mathbf{D} = \mathbf{T}\mathbf{G}^{-1}$.

With these definitions $\mathbf{T}$ and $\mathbf{D}$ are pseudo-inverses, in particular we have

Lemma 3.1.2 (Pseudo Inverses)**.** *The matrices* $\mathbf{T}$ *and* $\mathbf{D}$ *satisfy*

$$\mathbf{T}^\mathsf{T}\mathbf{D} = \mathbf{I}_d, \tag{3.1}$$

$$\mathbf{T}\mathbf{D}^\mathsf{T} = \mathbf{I}_{d+1} - \boldsymbol{\nu} \otimes \boldsymbol{\nu}, \tag{3.2}$$

where $\boldsymbol{\nu}$ *is the outer normal.*

Proof. The first equation follows trivially from the definitions as $\mathbf{T}^\mathsf{T}\mathbf{D} = \mathbf{T}^\mathsf{T}\mathbf{T}\mathbf{G}^{-\mathbf{1}} = \mathbf{G}\mathbf{G}^{-\mathbf{1}} = \mathbf{I}_d$. For the second one observe that the columns of $\mathbf{T}$ and $\boldsymbol{\nu}$ form a basis of $\mathbb{R}^{d+1}$. Now, $\mathbf{T}\mathbf{D}^\mathsf{T} = \mathbf{T}\mathbf{G}^{-\mathbf{1}}\mathbf{T}^\mathsf{T}$ so $\mathbf{T}\mathbf{D}^\mathsf{T}\mathbf{T} = \mathbf{T}$ which implies that $\mathbf{T}\mathbf{D}^\mathsf{T}$ is the identity restricted to the tangent plane. Since $\mathbf{T}\mathbf{D}^\mathsf{T}\boldsymbol{\nu} = \mathbf{0}$, and the unique linear transformation with this mapping of the basis is $\mathbf{I}_{d+1} - \boldsymbol{\nu} \otimes \boldsymbol{\nu}$, (3.2) follows. □

With these notation we can give a formula for the the differential $Df(q)$ of the function $f : \Gamma \to \mathbb{R}^m$ in embedding space coordinates.

Lemma 3.1.3 (Differential in Embedding Space)**.** *Let* $f : \Gamma \to \mathbb{R}^m$ *be a differentiable function and* $\boldsymbol{x}$ *a local parametrization of* Γ*. Then the differential* $Df(q)$ *is given by*

$$\left(D(f \circ \boldsymbol{x})\mathbf{D}^\mathsf{T}\right) \circ \boldsymbol{x}^{-1}, \tag{3.3}$$

which is independent of the chosen parametrization $\boldsymbol{x}$*.*

Proof. If $\boldsymbol{t} \in T_q\Gamma$ then $\boldsymbol{t} = \Sigma t^i \partial_i \boldsymbol{x}$. Recalling Definition 3.1.5 we can define $\hat{\alpha}(s) = \hat{q} + s\hat{t}$ and $\alpha = \boldsymbol{x} \circ \hat{\alpha}$, where $\hat{\boldsymbol{t}} = \Sigma t^i \boldsymbol{e}_i$. Since $\boldsymbol{t} = D\boldsymbol{x}\ \hat{\alpha}' = \mathbf{T}\hat{\boldsymbol{t}}$, using equation (3.1) we get $\hat{\boldsymbol{t}} = \mathbf{D}^\mathsf{T}\boldsymbol{t}$. So finally, $Df(q)\boldsymbol{t} = (f \circ \alpha)'(0) = ((f \circ \boldsymbol{x}) \circ \hat{\alpha})'(0) = D(f \circ \boldsymbol{x})\mathbf{D}^\mathsf{T}\boldsymbol{t}$. The independence of the parametrization $\boldsymbol{x}$ follows from Remark 3.1.1. □

Remark 3.1.4 (Differential Geometry Notation). Classical differential geometry deals with coefficients $g_{ij} := \partial_i \boldsymbol{x} \cdot \partial_j \boldsymbol{x}$. According to our definitions see that $(\mathbf{G})_{ij} = g_{ij}$ and the coefficients g^{ij} verified $g^{ij} := (\mathbf{G}^{-1})_{ij}$. This shows the explicit connection between classical differential geometry and the previous matrix notation.

For integration on the surface we define the *area element* q by $q^2 = \det(\mathbf{G})$. Useful differential operators can be defined on a surface, thereby making rigorous the idea of tangential derivatives of any order. Invoking the Ritz representation theorem we define the surface gradient.

Definition 3.1.6 (Surface Gradient)**.** Given a differentiable function $f : \Gamma \to \mathbb{R}$, and a point $q \in \Gamma$, the surface gradient is the unique vector $\nabla_\Gamma f(q) \in T_q\Gamma$ such that $Df(q)\ \boldsymbol{t} = \nabla_\Gamma f \cdot \boldsymbol{t}$ for all $\boldsymbol{t} \in T_q\Gamma$.

Remark 3.1.5 (Surface Gradient in Embedding Space). Equation (3.3) gives the differential in embedding space coordinates as a matrix vector product. The matrix-vector representation transforms into scalar product representation by taking the transpose, so we get

$$\nabla_\Gamma f = (\mathbf{D}\nabla(f \circ \boldsymbol{x})) \circ \boldsymbol{x}^{-1}.$$

Following Remark 3.1.5 we define the *tangential derivatives* with respect to the i-th embedding coordinate $\underline{D}_i f := (\nabla_\Gamma f)_i$. This recursively defines tangential derivatives of any order. In abstract setting the *surface divergence* is defined as $\nabla_\Gamma \cdot \boldsymbol{w} = \mathcal{L}_{\boldsymbol{w}}$, where $\mathcal{L}_{\boldsymbol{w}}$ denotes the Lie derivative in the direction of the vector field $\boldsymbol{w}$. In our case we can use the previous definition of tangential derivatives and define $\nabla_\Gamma \cdot \boldsymbol{w} = \Sigma \underline{D}_i w^i$. Finally combining the last two definitions the surface

Laplacian or Laplace-Beltrami operator con be defined.

Definition 3.1.7 (Surface Laplacian)**.** Given a differentiable function $f : \Gamma \to \mathbb{R}$, and a point $q \in \Gamma$, the surface Laplacian of f is given by $\Delta_\Gamma f = \nabla_\Gamma \cdot \nabla_\Gamma f$.

Definitions 3.1.6 through 3.1.7 take the following computational form when the function on the surface is extended to embedding space or define in parameter space. If $\tilde{v}$ and $\tilde{\boldsymbol{q}}$ are smooth extensions of v and $\boldsymbol{q}$ to a neighborhood of Γ respectively, then

$$\nabla_\Gamma v = \nabla \tilde{v} - (\nabla \tilde{v} \cdot \boldsymbol{\nu})\boldsymbol{\nu}, \tag{3.4}$$

$$\nabla_\Gamma \cdot \boldsymbol{q} = \nabla \cdot \tilde{\boldsymbol{q}} - \boldsymbol{\nu}(D\tilde{\boldsymbol{q}})\boldsymbol{\nu}, \tag{3.5}$$

$$\Delta_\Gamma v = \nabla_\Gamma \cdot \nabla_\Gamma v = \Delta \tilde{v} - \boldsymbol{\nu}(D^2 \tilde{v})\boldsymbol{\nu} - (\nabla \tilde{v} \cdot \boldsymbol{\nu})\nabla \cdot \boldsymbol{\nu}, \tag{3.6}$$

where ∇,$\nabla\cdot$ and Δ denote the usual gradient, divergence and Laplacian in $\mathbb{R}^{d+1}$. In parametric coordinates we have the following formulas:

$$(\nabla_\Gamma f)^i = g^{ij}\partial_j f, \tag{3.7}$$

$$\nabla_\Gamma \cdot \boldsymbol{w} = \frac{1}{q}\partial_i \left(q w^i\right), \tag{3.8}$$

$$\Delta_\Gamma f = \frac{1}{q}\partial_i \left(q g^{ij}\partial_j f\right). \tag{3.9}$$

Applying definitions 3.1.6 through 3.1.7 to each component we can define the surface gradient of a vector field that will be a tensor. Also the surface divergence of a tensor, and consequently the surface Laplacian of a vector field. The curvature of a surface measures the change of its normal vector. The *second fundamental form* $\boldsymbol{\Pi} := \nabla_\Gamma \boldsymbol{\nu}$ is a tensor that defines the *principal curvatures* $\kappa_1, \ldots, \kappa_{d-1}$ as its eigenvalues. We denote by $h := \kappa_1 + \cdots + \kappa_{d-1}$ the *total mean curvature*, by $k := \kappa_1 \cdots \kappa_{d-1}$ the

Gauss curvature, and by $\boldsymbol{h} := h\boldsymbol{\nu}$ the *vector mean curvature*. It can be shown that with the previous differential operators definitions the following integration by parts formula for surfaces holds:

Theorem 3.1.4 (Integration by Parts on Surfaces)**.** *If f is a scalar function defined on the surface Γ, then*

$$\int_\Gamma \nabla_\Gamma f = \int_\Gamma f\boldsymbol{h} + \int_{\partial\Gamma} f\boldsymbol{\nu}_s, \tag{3.10}$$

where $\partial\Gamma$ is the boundary of Γ and $\boldsymbol{\nu}_s$ its conormal vector.

Proof. See [GT83, Lemma 16.1]. □

From here the divergence theorem for surfaces follows.

Corollary 3.1.5 (Divergence Theorem for Surfaces)**.** *Let $\boldsymbol{q}$ be a vector field defined on the surface Γ. Then*

$$\int_\Gamma \nabla_\Gamma \cdot \boldsymbol{q} = \int_\Gamma \boldsymbol{q}\cdot\boldsymbol{h} + \int_{\partial\Gamma} \boldsymbol{q}\cdot\boldsymbol{\nu}_s. \tag{3.11}$$

Proof. Theorem 3.1.4 component wise reads

$$\int_\Gamma \underline{D}_i f = \int_\Gamma f\boldsymbol{h}^i + \int_{\partial\Gamma} f\boldsymbol{\nu}_s^i. \tag{3.12}$$

By definition $\nabla_\Gamma \cdot \boldsymbol{q} = \Sigma \underline{D}_i q^i$, then

$$\int_\Gamma \nabla_\Gamma \cdot \boldsymbol{q} = \Sigma \int_\Gamma \underline{D}_i \boldsymbol{q}^i = \Sigma \int_\Gamma \boldsymbol{q}^i\boldsymbol{h}^i + \Sigma \int_{\partial\Gamma} \boldsymbol{q}^i\boldsymbol{\nu}_s^i = \int_\Gamma \boldsymbol{q}\cdot\boldsymbol{h} + \int_{\partial\Gamma} \boldsymbol{q}\cdot\boldsymbol{\nu}_s. \tag{3.13}$$

□

Lemma 3.1.6 (Total Mean Curvatures)**.** *The following useful formulas to compute mean curvature are valid:*

$$h = \nabla_\Gamma \cdot \boldsymbol{\nu} \quad \textit{and} \quad \boldsymbol{h} = -\Delta_\Gamma \mathbf{Id}. \tag{3.14}$$

Proof. See [GT83, page 390]. □

Lemma 3.1.7 (Weak Formulas for Curvatures)**.** *Let Γ be a smooth surface without boundary. Then for all smooth functions $\boldsymbol{w} : \Gamma \to \mathbb{R}^{d+1}$ we have*

$$\int_\Gamma \boldsymbol{h} \cdot \boldsymbol{w} = \int_\Gamma \nabla_\Gamma \mathbf{Id} : \nabla_\Gamma \boldsymbol{w}, \tag{3.15}$$

$$\int_\Gamma \boldsymbol{\Pi}^j \cdot \boldsymbol{w} = -\int_\Gamma \boldsymbol{\nu}^j \nabla_\Gamma \cdot \boldsymbol{w} - \int_\Gamma \boldsymbol{\nu}^j \boldsymbol{h} \cdot \boldsymbol{w}. \tag{3.16}$$

Proof. First observe that $\nabla_\Gamma \mathbf{Id} = \mathbf{I}_{d+1} - \boldsymbol{\nu} \otimes \boldsymbol{\nu}$. Formula (3.14) gives $\int_\Gamma \boldsymbol{h} \cdot \boldsymbol{w} = \int_\Gamma -\Delta_\Gamma \mathbf{Id} \cdot \boldsymbol{w}$ and the product rule yields $\nabla_\Gamma \cdot (\nabla_\Gamma \mathbf{Id} \boldsymbol{w}) = \nabla_\Gamma \mathbf{Id} : \nabla_\Gamma \boldsymbol{w} + \boldsymbol{w} \cdot \Delta_\Gamma \mathbf{Id}$. Then using (3.10) we get $\int_\Gamma \nabla_\Gamma \cdot (\nabla_\Gamma \mathbf{Id} \boldsymbol{w}) = \int_\Gamma \nabla_\Gamma \mathbf{Id} \boldsymbol{w} \cdot \boldsymbol{h}$, whence

$$\int_\Gamma \boldsymbol{h} \cdot \boldsymbol{w} = -\int_\Gamma \nabla_\Gamma \mathbf{Id} \boldsymbol{w} \cdot \boldsymbol{h} + \int_\Gamma \nabla_\Gamma \mathbf{Id} : \nabla_\Gamma \boldsymbol{w}$$

But $\nabla_\Gamma \mathbf{Id} \boldsymbol{w} \cdot \boldsymbol{h} = (\mathbf{I} - \boldsymbol{\nu} \otimes \boldsymbol{\nu}) \boldsymbol{h} \cdot \boldsymbol{w} = \mathbf{0} \cdot \boldsymbol{w}$, and (3.15) follows. Equation (3.16) can be obtained in a similar way. □

With the definitions of tangential derivatives we use $H^1(\Gamma)$, $W_p^j(\Gamma)$, etc to denote the Sobolev spaces of functions defined on Γ possessing j weak derivatives in $L^p(\Gamma)$. For future reference we consider the following result that states that for subset of planes the tangential derivatives are equivalent to the partial derivatives of any rigid parametrization. We say that Γ is a *rigid deformation* of $\hat{\Omega}$ it there is $F : \hat{\Omega} \to \Gamma$ such that $F(\hat{\boldsymbol{x}}) = \mathbf{B}\hat{\boldsymbol{x}} + \boldsymbol{b}$ with $\mathbf{B}^\mathsf{T}\mathbf{B} = \mathbf{I}$.

Lemma 3.1.8. *(Tangential derivative of flat surfaces) Let the surface Γ be a rigid deformation of $\hat{\Omega} \subset \mathbb{R}^d$. then there exist constant $C_1, C_2 > 0$ depending on d such that*

$$C_1 |\hat{v}|_{k,p,\hat{\Omega}} \leq |v|_{k,p,\Gamma} \leq C_2 |\hat{v}|_{k,p,\hat{\Omega}}, \tag{3.17}$$

for any $v \in W_p^k(\Gamma)$, where $\hat{v} = v \circ F$ and F is the rigid deformation.

Proof. By definition $F(\hat{\boldsymbol{x}}) = \mathbf{B}\hat{\boldsymbol{x}} + \boldsymbol{b}$, where B is matrix with $\mathbf{B}^\mathsf{T}\mathbf{B} = \mathbf{I}$. For this parametrization on the surface we have $\mathbf{T} = \mathbf{I}$ then $\mathbf{D} = \mathbf{B}$ From Remark 3.1.5 $\nabla_\Gamma v = \mathbf{B}\nabla\hat{v}$ then $|\nabla_\Gamma v| = |\mathbf{B}||\nabla\hat{v}|$ but $|\mathbf{B}| = 1$ The other bound is obtain in a similar way using $\mathbf{B}^\mathsf{T}$. The previous argument can be repeated for higher order derivatives. □

3.2 Shape Differential Calculus

In this work we will use shape functionals $J(\Gamma)$ or $J(\Omega)$ as for example:

- the volume functional: $J(\Omega) = \int_\Omega 1 = \text{meas}(\Omega)$,
- the area functional: $J(\Gamma) = \int_\Gamma 1 = \text{meas}(\Gamma)$,
- the bending energy functional: $J(\Gamma) = \int_\Gamma h^2$.

The functional derivatives can be defined and computed in the context of the velocity method and shape differential calculus [SZ92]. A useful summary of results can be found in [Dog06]. Let $\mathcal{D} \subset \mathbb{R}^{d+1}$ be a domain containing Γ. Let $\boldsymbol{v}$ be a smooth vector field defined in $\mathcal{D}$. Then through an autonomous system of ODEs prescribed by $\boldsymbol{v}$ the surface $\Gamma = \Gamma_0$ is deformed in a sequence of perturbed surfaces $\{\Gamma_t\}_{t\geq 0}$. Under these considerations we define the shape derivative of J.

Definition 3.2.1 (Shape Derivative)**.** The Eulerian or *shape derivative* of the functional $J(\Gamma)$ at Γ, in the direction of the vector field $\boldsymbol{v}$ is defined as the limit

$$dJ(\Gamma; \boldsymbol{v}) = \lim_{t\to 0} \frac{1}{t}(J(\Gamma_t) - J(\Gamma)). \tag{3.18}$$

A similar definition holds if the surface Γ is replaced by a domain Ω. When the functional is of the form $J(\Gamma) = \int_\Gamma \psi$ with ψ not depending on the geometry we have

Lemma 3.2.1 ([SZ92],Prop. 2.45)**.** *Let $\psi \in W_1^1(\mathbb{R}^{d+1})$ and Ω be a smooth and bounded domain. The functional*

$$J(\Omega) = \int_\Omega \psi$$

is shape differentiable and if $v = \boldsymbol{v} \cdot \boldsymbol{\nu}$, then

$$dJ(\Omega; \boldsymbol{v}) = \int_\Omega \nabla \cdot (\psi \boldsymbol{v}) = \int_\Gamma \psi v.$$

Lemma 3.2.2 ([SZ92],Prop. 2.50 and (2.145))**.** *Let $\psi \in W_1^2(\mathbb{R}^{d+1})$ and Γ be of class C^2. Then the functional*

$$J(\Gamma) = \int_\Gamma \psi$$

is shape differential and

$$dJ(\Gamma; \boldsymbol{v}) = \int_\Gamma (\nabla \psi \cdot \boldsymbol{v} + \psi \nabla_\Gamma \cdot \boldsymbol{v}) = \int_\Gamma (\partial_{\boldsymbol{\nu}} \psi + \psi h) v.$$

Now we want to allow the function ψ to depend also on the geometry, i.e. $\psi = \psi(x, \Gamma)$. To carry on the previous result to this case we need the definition of the material and shape derivatives of ψ.

Definition 3.2.2 (Material Derivative)**.** The material derivative $\dot{\psi}(\Gamma; \boldsymbol{v})$ of ψ in the direction $\boldsymbol{v}$ is defined as follows

$$\dot{\psi}(\Gamma; \boldsymbol{v}) = \lim_{t \to 0} \frac{1}{t} \left(\psi(x(\cdot, t), \Gamma_t) - \psi(\cdot, \Gamma_0) \right). \tag{3.19}$$

Definition 3.2.3 (Shape Derivative)**.** The shape derivative $\psi'(\Gamma; \boldsymbol{v})$ of ψ in the direction $\boldsymbol{v}$ is defined as follows

$$\psi'(\Gamma; \boldsymbol{v}) = \dot{\psi}(\Gamma; \boldsymbol{v}) - \nabla\psi \cdot \boldsymbol{v}. \tag{3.20}$$

With these definitions a similar result to Lemma 3.2.2 follows.

Theorem 3.2.3 ([SZ92], Sect. 2.31, 2.33)**.** *Let $\psi = \psi(x, \Gamma)$ be given so that the material derivative $\dot{\psi}(\Gamma; \boldsymbol{v})$ and the shape derivative $\psi'(\Gamma; \boldsymbol{v})$ exist. Then $J(\Gamma)$ is shape differentiable and*

$$dJ(\Gamma; \boldsymbol{v}) = \int_\Gamma \psi'(\Gamma; \boldsymbol{v}) + \int_\Gamma (\partial_{\boldsymbol{\nu}}\psi + \psi h)v. \tag{3.21}$$

Now we state some geometric results that will be useful for our applications.

Lemma 3.2.4 ([Dog06],Lemma 2.1.3 and 2.1.4)**.** *The shape derivatives of the normal and mean curvature of a boundary Γ of class C^2 with respect to velocity $\boldsymbol{v}$ are given by*

$$\boldsymbol{\nu}' = \boldsymbol{\nu}'(\Gamma; \boldsymbol{v}) = -\nabla_\Gamma v \tag{3.22}$$

$$h' = h'(\Gamma; \boldsymbol{v}) = -\Delta_\Gamma v. \tag{3.23}$$

In addition the normal derivative of the mean curvature is

$$\partial_{\boldsymbol{\nu}} h = -\sum \kappa_i^2. \tag{3.24}$$

In the particular case of a two dimensional surface in $\mathbb{R}^3$, $\partial_{\boldsymbol{\nu}} h = -(h^2 - 2k)$.

Proof. See [SZ92, Sect. 2.31, 2.33]. □

Theorem 3.2.5 (Hadamard-Zolésio)**.** *The shape derivative of a domain or boundary functional always has a representation of the form*

$$dJ(\Gamma; \boldsymbol{v}) =< \delta J, v >_{\Gamma},$$

where $v = \boldsymbol{v} \cdot \boldsymbol{\nu}$.

Proof. See [SZ92] Section 2.11 and Theorem 2.27. □

We call δJ the *first variation* of J.

3.3 Continuum Mechanics

Let $\mathcal{E}$ be the Euclidean point space with the associated vector space $\mathcal{V}$ such that $\boldsymbol{y} - \boldsymbol{x} \in \mathcal{V}$ if $\boldsymbol{y}, \boldsymbol{x} \in \mathcal{E}$. A reference configuration $\hat{\Omega}$ is a chosen regular region of space. A point $\boldsymbol{p} \in \hat{\Omega}$ is called a material point. A deformation is a mapping $\boldsymbol{f} : \hat{\Omega} \to \mathcal{E}$ smooth, one to one and such that $\det(D\boldsymbol{f}) > 0$. Hereafter, $D\boldsymbol{f}$ is called the *deformation gradient*. A *motion* of $\hat{\Omega}$ is a smooth function $\mathbf{x} : \hat{\Omega} \times [0, T] \to \mathcal{E}$ such that $\mathbf{x}(\cdot, t)$ is a deformation for each $t \in [0, T]$. The point $\boldsymbol{x} = \mathbf{x}(\boldsymbol{p}, t)$ is the place occupied by material point $\boldsymbol{p}$ at time t, and $\Omega_t = \mathbf{x}(\hat{\Omega}, t)$ is the place occupied by the body $\hat{\Omega}$ at time t. Sometimes it is convenient to work with places and time instead of material points, for which we define the *trajectory*

$$G_T = \{(\boldsymbol{x}, t) : \boldsymbol{x} \in \Omega_t, t \in [0, T]\}.$$

The condition $\det(D\boldsymbol{f}) > 0$ implies the existence of the reference map $\mathbf{p} : G_T \to \hat{\Omega}$ such that $\mathbf{x}(\mathbf{p}(\boldsymbol{x}, t), t) = \boldsymbol{x}$ and $\mathbf{p}(\mathbf{x}(\boldsymbol{p}, t), t) = \boldsymbol{p}$. Given a motion $\mathbf{x}$, $\dot{\mathbf{x}}$ and $\ddot{\mathbf{x}}$ are

the *velocity* and *acceleration* respectively. A spatial description of the velocity is defined by

$$\boldsymbol{v}(\boldsymbol{x}, t) = \dot{\mathbf{x}}(\mathbf{p}(\boldsymbol{x}, t)).$$

In general any field associated with a motion can be described either as a function of space or material points. The Euclidean space $\mathcal{E}$ in not a normed vector space. However we can define the derivative of functions defined in a region of $\mathcal{E}$ or taking values on it by replacing $\mathcal{U}$ and/or $\mathcal{W}$, in Definition 1.2.1 by the associated vector space $\mathcal{V}$. For future reference we state the derivative of the determinant.

Lemma 3.3.1 (Derivative of Determinant)**.** *If* $\mathbf{A}$ *is an invertible tensor, then*

$$D(\det)(\mathbf{A})[\mathbf{U}] = (\det(\mathbf{A}))\,\mathrm{tr}(\mathbf{U}\mathbf{A}^{-1}).$$

Proof. See [Gur81] page 23. □

Given a smooth vector field $\boldsymbol{w}$ the divergence operator is defined by

$$\nabla \cdot \boldsymbol{w} := \mathrm{tr}(\nabla \boldsymbol{w}), \tag{3.25}$$

and for a tensor field $\mathbf{S}$, as the unique linear operator $\nabla \cdot \boldsymbol{S}$ such that

$$(\nabla \cdot \boldsymbol{S})\boldsymbol{a} = \nabla \cdot (\boldsymbol{S}^{\mathsf{T}}\boldsymbol{a}) \text{ for all vector } \boldsymbol{a}. \tag{3.26}$$

Next we summarize some derivatives of products that will be used extensively.

Proposition 3.3.2 (Product Rules)**.** *Let* ϕ*,* $\boldsymbol{u}$ *and* $\mathbf{S}$ *be spatial scalar, vector and*

tensor fields. Then

$$\begin{aligned}
\nabla(\phi \boldsymbol{u}) &= \phi \nabla \boldsymbol{u} + \boldsymbol{u} \otimes \nabla \phi, \\
\nabla \cdot (\phi \boldsymbol{u}) &= \phi \nabla \cdot \boldsymbol{u} + \boldsymbol{u} \cdot \nabla \phi, \\
\nabla \cdot (\mathbf{S}^{\mathsf{T}} \boldsymbol{u}) &= \mathbf{S} : \nabla \boldsymbol{u} + \boldsymbol{u} \cdot \nabla \cdot \mathbf{S}, \\
\nabla \cdot (\phi \mathbf{S}) &= \phi \nabla \cdot \mathbf{S} + \mathbf{S} \nabla \phi, \\
\nabla(\boldsymbol{u} \cdot \boldsymbol{w}) &= (\nabla \boldsymbol{u})^{\mathsf{T}} \boldsymbol{u} + (\nabla \boldsymbol{w})^{\mathsf{T}} \boldsymbol{u}.
\end{aligned} \tag{3.27}$$

Proof. See [Gur81] page 30. □

Given a spatial field it is convenient to define the material or convective time derivative. Roughly speaking, it represents the time derivative of the spatial field holding the material point fixed.

Definition 3.3.1 (Material Derivative II)**.** If ϕ is a spatial field, then the material derivative is given by

$$\dot{\phi}(\boldsymbol{x}, t) = \frac{\partial}{\partial t} \phi(\mathbf{x}(\boldsymbol{p}, t), t)|_{\boldsymbol{p}=\mathbf{p}(\boldsymbol{x},t)}.$$

This definition is quite similar to Definition 3.2.2. The difference is that there the velocity was an arbitrary parameter, here it is the specific velocity given by the motion.

Lemma 3.3.3. *If ϕ is a scalar and $\boldsymbol{w}$ a vector spatial fields, then*

$$\begin{aligned}
\dot{\phi} &= \frac{\partial \phi}{\partial t} + \boldsymbol{v} \cdot \nabla \phi, \\
\dot{\boldsymbol{w}} &= \frac{\partial \boldsymbol{w}}{\partial t} + \nabla \boldsymbol{w} \boldsymbol{v}.
\end{aligned}$$

Lemma 3.3.4 (Reynolds Transport Theorem)**.** *If Φ is a smooth spatial field, then*

$$\frac{d}{dt}\int_{\mathcal{P}_t}\Phi = \int_{\mathcal{P}_t}(\dot{\Phi}+\Phi\nabla\cdot\boldsymbol{v}) = \int_{\mathcal{P}_t}\Phi' + \int_{\partial\mathcal{P}_t}\Phi\boldsymbol{v}\cdot\boldsymbol{\nu}.$$

Proof. See [Gur81] page 78. □

So far all definitions and results have been stated independently of a coordinate system. Also we have been using what is known as the direct notation as opposed to component notation. To do computations a coordinate system is necessary. For this purpose we fix a *Cartesian frame*, i.e. an orthonormal basis of $\mathcal{V}$ and a point in $\mathcal{E}$ called the origin. Then any point in $\mathcal{E}$ and any vector in $\mathcal{V}$ have associated uniquely its Cartesian components in $\mathbb{R}^{d+1}$. Then the body becomes a region in $\mathbb{R}^{d+1}$ and the derivatives in the sense of Definition 1.2.1 become arrays of partial derivatives. In particular, if $f:\mathbb{R}^{d+1}\to\mathbb{R}$, $\boldsymbol{w}:\mathbb{R}^{d+1}\to\mathbb{R}^{d+1}$ and $\mathbf{S}:\mathbb{R}^{d+1}\to\mathbb{R}^{d+1}\times\mathbb{R}^{d+1}$ then

$$\begin{aligned}(Df)_{ij} &= \frac{\partial f^i}{\partial x_j}\\ (\nabla\boldsymbol{w})_{ij} &= \frac{\partial \boldsymbol{w}^i}{\partial x_j}\\ (\nabla\cdot\mathbf{S})_i &= \sum_j \frac{\partial S_{ij}}{\partial x_j}\end{aligned} \tag{3.28}$$

In this work we abuse notation and refer to a function defined in Euclidean space or in $\mathbb{R}^{d+1}$ for a given coordinate system with the same name.

So far we have been dealing with kinematics (description of motions), now we proceed to survey the dynamics (i.e. reasons for motion). An important property of bodies is that they posses mass. This is reflected in a motion by the existence of a density function $\rho(\boldsymbol{x},t)$.

During a motion mechanical interaction between parts of a body or between

a body and its environment are described by *forces*. Forces between parts of a body or exerted on the boundary by the environment are called *contact forces*. Forces that the environment exerts on the interior of the body are called *body forces* and we will denote them by $\boldsymbol{b}$. One of the most important axioms in continuum mechanics is Cauchy's hypothesis on the form of the contact forces, which together with the balance of momentum axioms, gives one of the central results of continuum mechanics.

Theorem 3.3.5 (Existence of the stress tensor)**.** *There exists a symmetric spatial tensor field* $\boldsymbol{\Sigma}$ *(called the Cauchy stress) such that it satisfies the equation of motion*

$$\rho \dot{\boldsymbol{v}} = \nabla \cdot \boldsymbol{\Sigma} + \boldsymbol{b} \tag{3.29}$$

Proof. See [Gur81] page 101. □

The equation of motion is true for all bodies in nature. But it does not distinguish between different types of material. Additional hypothesis called *constitutive equations* are then added to account for different type of material behaviors.

3.3.1 Newtonian Fluids

Friction in fluids generally manifests itself through shearing forces which retard the relative motion of fluid particles. This can be measured by the velocity gradient. When the stress tensor is a pressure plus a linear function of the velocity gradient the fluid is called *Newtonian*. They furnish the simplest model for viscous fluids and

consist of constitutive equations of the form:

$$\boldsymbol{\Sigma} = -p\mathbf{I} + \mu\mathbf{D}(\boldsymbol{v}),$$

$$\nabla \cdot \boldsymbol{v} = 0,$$

where $\mathbf{D}(\boldsymbol{v}) = \nabla\boldsymbol{v} + \nabla\boldsymbol{v}^{\mathsf{T}}$ is twice the symmetric part of the velocity gradient. This leads to the famous Navier-Stokes equations for incompressible fluids

$$\begin{aligned} \rho\dot{\boldsymbol{v}} - \nabla \cdot (-p\mathbf{I} + \mu\mathbf{D}(\boldsymbol{v})) &= \boldsymbol{b}, \\ \nabla \cdot \boldsymbol{v} &= 0. \end{aligned} \qquad \text{in } G_T \tag{3.30}$$

For numerical computations it is quite convenient to express the problem in dimensionless form. For this we define the new dimensionless variables

$$\bar{\boldsymbol{x}} = \frac{\boldsymbol{x}}{L} \quad \bar{\boldsymbol{v}} = \frac{\boldsymbol{v}}{V} \quad \bar{t} = \frac{tV}{L} \quad \bar{p} = \frac{p}{V^2\rho}, \tag{3.31}$$

where L is the characteristic length (size of the domain) and V the characteristic speed (main stream velocity). Using the change of variables (3.31) and the chain rule for differentiation in equation (3.30) we get

$$\begin{aligned} \dot{\bar{\boldsymbol{v}}} - \nabla \cdot \bar{\Sigma} &= \bar{\boldsymbol{b}} \\ \nabla \cdot \bar{\boldsymbol{v}} &= 0, \end{aligned} \qquad \text{in } \bar{G}_{\bar{T}} \tag{3.32}$$

where $\bar{\boldsymbol{\Sigma}} = -\bar{p}\mathbf{I} + \frac{1}{Re}\bar{\mathbf{D}}(\bar{\boldsymbol{v}})$, $\bar{\boldsymbol{b}} = \frac{\boldsymbol{b}L}{\rho V^2}$ and recall that Re is the *Reynolds number*

$$Re = \frac{\rho V L}{\mu}.$$

Also $\bar{G}_{\bar{T}}$ is naturally defined by the change of variables from G_T. The same change of variable gives the following relation for the stress tensors

$$\boldsymbol{\Sigma}(L\bar{\boldsymbol{x}}, \bar{t}(L/V)) = \rho V^2 \bar{\boldsymbol{\Sigma}}(\bar{\boldsymbol{x}}, \bar{t}). \tag{3.33}$$

In section 4.3.1 we will use the classical capillarity number Ca, and in section 4.3.3 we will introduce a similar bending number Be to put the bending force problem in a dimensionless form.

Chapter 4

Continuous Problems: Models

This chapter presents a set of applications that can be implemented within the computational framework of chapter 7. The applications are divided into two groups (section 4.2 and 4.3). The first one is concerned with geometric problems on a surface (or curve). The goal is to find a shape that minimizes certain geometric energy. This is studied in the context of what is known as geometric evolution equations: basically, a *gradient flow* that prescribes the evolution of the surface in a direction that decreases the energy. The second group focuses on models to describe a *fluid-membrane interaction*. The membrane is immersed and assumed to be endowed of a geometric energy, thus making the link back to the first group.

The addition of constraints is quite important for both groups of applications. So we start this chapter with a section describing how they are added. For each application the classical and a convenient weak formulation to use a FEM are provided together with some characteristic properties. Both types of models have applications to *biomembranes*. In the geometric model all quantities live on the surface. So from a computational point of view the number of degrees of freedom corresponds to a one dimensional (curve) or a two dimensional (surface) problem. On the other hand, for the coupled fluid-membrane model the number of degrees of freedom corresponds to two or three dimensional problems. The drawback of the geometric

model is that the generated dynamics is non-physical. If all that matters is to find the final equilibrium shape then a geometric model may be adequate. However, if what matters is the dynamics, then the coupled model is the proper one. In chapter 6 discrete schemes of these problems will be studied.

4.1 Constraints

Many times when dealing with geometric flows or fluid-membrane interactions coming from applications, they have an area or [and] volume constraint[s] attached. This global type of constraints is called *isoperimetric*. They are imposed to the continuous problem by means of Lagrange multipliers that define an augmented energy. To illustrate the idea consider the evolution of a surface $\{\Gamma(t)\}_{t\geq 0}$ prescribed as the gradient flow of some energy $E(\Gamma) = \int_\Gamma w$. Formally, it can be written as

$$\boldsymbol{v} = -\delta E, \tag{4.1}$$

where $\boldsymbol{v}$ is the velocity that prescribes the evolution $\{\Gamma(t)\}_{t\geq 0}$ and δE is the functional derivative of E. The augmented energy to account for area and volume constraints, using Lagrange multipliers λ and π, is

$$\tilde{E}(\Gamma, \lambda, \pi) := \int_\Gamma w + \lambda \left(\int_\Gamma 1 - \int_{\Gamma_0} 1 \right) + \pi \left(\int_\Gamma \mathbf{Id} \cdot \boldsymbol{\nu} - \int_{\Gamma_0} \mathbf{Id} \cdot \boldsymbol{\nu} \right), \tag{4.2}$$

where Γ_0 is the given initial shape, $\mathbf{Id}$ denotes the identity function and $\boldsymbol{\nu}$ the outer unit normal. Finally the geometric flow equations obtained from the new energy (4.2) are supplemented with the two scalar conservation equations (for the area and

volume) to give the new system

$$\begin{aligned} \boldsymbol{v} &= -\delta\tilde{E}(\Gamma,\lambda,\pi) = -\delta E - \lambda\ \delta A - \pi\ \delta V \\ A(\Gamma) &= \text{meas}(\Gamma) = \text{meas}(\Gamma_0) \\ V(\Gamma) &= \text{meas}(\Omega) = \text{meas}(\Omega_0), \end{aligned} \tag{4.3}$$

where Ω is the volume enclosed by Γ. For more details on imposing constraints using Lagrange multipliers see [GH96].

Remark 4.1.1. The λ term in equation (4.2) obviously imposes the area conservation. The π term imposes the volume conservation as long as the surface does not self intersect. Indeed, invoking the divergence theorem we obtain

$$\int_\Gamma \mathbf{Id}\cdot\boldsymbol{\nu} = \int_\Omega \nabla\cdot\mathbf{Id} = (d+1)\,\text{meas}(\Omega).$$

Remark 4.1.2 (Physical Interpretation of Multipliers). In the context of biomembrane modeling the λ multiplier can be interpreted as the uniform surface tension that the membrane should posses for the bending force not to change the surface area. Similarly the π multiplier can be interpreted as the uniform pressure difference between the interior and exterior of the membrane for the bending force not to change the enclose volume.

Similarly, for the general case if we want to impose N isoperimetric constraints of the form $F_i(\Gamma) = \int_\Gamma f_i$, the augmented system will be

$$\begin{aligned} \boldsymbol{v} &= -\delta\tilde{E}(\Gamma,\lambda_1,\ldots,\lambda_N) = -\delta E + \sum_{i=1}^{N}\lambda_i\ \delta F_i \\ F_i(\Gamma_t) &= F_i(\Gamma_0) \quad i = 1,\ldots,N. \end{aligned} \tag{4.4}$$

In section 7.3 a novel method to solve for the multipliers in the discrete approximation to the solution of system (4.4) is presented. For future use we state the variational derivative of the area and volume functionals.

Theorem 4.1.1. *Let Γ be a surface without boundary. Let $A(\Gamma) = \int_\Gamma 1$ and $V(\Gamma) = \frac{1}{d+1}\int_\Gamma \mathbf{Id}\cdot\boldsymbol{\nu}$ be the area and volume functionals. Then, for all smooth vector functions $\boldsymbol{\phi}$, we get*

$$dA(\Gamma;\boldsymbol{\phi}) = \int_\Gamma \delta A\cdot\boldsymbol{\phi} = \int_\Gamma \boldsymbol{h}\cdot\boldsymbol{\phi}, \tag{4.5}$$

$$dV(\Gamma;\boldsymbol{\phi}) = \int_\Gamma \delta V\cdot\boldsymbol{\phi} = \int_\Gamma \boldsymbol{\phi}\cdot\boldsymbol{\nu}. \tag{4.6}$$

Proof. To prove (4.5) invoke Lemma 3.2.2 with $\boldsymbol{v} = \boldsymbol{\phi}$. It follows that $dA(\Gamma;\boldsymbol{\phi}) = \int_\Gamma \nabla_\Gamma\cdot\boldsymbol{\phi}$, and using equation (3.10) with $\boldsymbol{q} = \boldsymbol{\phi}$ the result follows. For the second equation observe that $\frac{1}{d+1}\int_\Gamma \mathbf{Id}\cdot\boldsymbol{\nu} = V(\Omega) = \int_\Omega 1$ and use Lemma 3.2.1 with $\psi = 1$ to deduce $dV(\Omega,\boldsymbol{\phi}) = \int_\Omega \nabla\cdot\boldsymbol{\phi} = \int_\Gamma \boldsymbol{\phi}\cdot\boldsymbol{\nu}$. □

A sufficient condition for the Lagrange multipliers to exist (at the equilibrium), see [GH96, Theorem 2 p.91], is given by the existence of $\boldsymbol{\phi}$ and $\boldsymbol{\psi}$ such that

$$\det\begin{bmatrix} \int_\Gamma \boldsymbol{h}\cdot\boldsymbol{\phi} & \int_\Gamma \boldsymbol{h}\cdot\boldsymbol{\psi} \\ \int_\Gamma \boldsymbol{\nu}\cdot\boldsymbol{\phi} & \int_\Gamma \boldsymbol{\nu}\cdot\boldsymbol{\psi} \end{bmatrix} \neq 0. \tag{4.7}$$

Lemma 4.1.2 (Existence of Multipliers)**.** *Let Γ be an equilibrium shape, which is not the trivial case of the sphere. Then the multipliers exist.*

Proof. Recall the relation $\boldsymbol{h} = h\boldsymbol{\nu}$ and choose $\boldsymbol{\phi} = \boldsymbol{h}$ and $\boldsymbol{\psi} = \boldsymbol{\nu}$ to obtain

$$\left(\int_\Gamma h^2\right)\left(\int_\Gamma 1\right) - \left(\int_\Gamma h\right)^2 \neq 0.$$

Since h is not constant, because the sphere is the only surface with constant h, h is linearly independent to 1 and the Cauchy-Schwarz inequality implies

$$\left(\int_\Gamma h\right)^2 < \left(\int_\Gamma h^2\right)\left(\int_\Gamma 1\right).$$

Therefore, (4.7) holds with $\boldsymbol{\phi} = \boldsymbol{h}$ and $\boldsymbol{\psi} = \boldsymbol{\nu}$ ensuring the existence of the two multipliers λ and π at every non spherical equilibrium, as asserted. □

4.2 Geometric Evolution Laws

When the motion of a surface only depends on its geometry, the governing equations are called a *geometric evolution equation*. More specifically, given an initial surface Γ_0 a geometric evolution equation prescribes the normal velocity $v = \boldsymbol{v} \cdot \boldsymbol{\nu} = f(t, \boldsymbol{x}, \boldsymbol{\nu}, \nabla_\Gamma \boldsymbol{\nu})$ as a function of time, position, the normal $\boldsymbol{\nu}$, and the curvatures. The most famous example of such an equation may be the mean curvature flow (section 4.2.1) introduced by W. Mullins in 1956 [Mul56] to describe motion of grain boundaries. Geometric evolution equations have a wide range of applications such as in the field of material sciences, image processing and biophysics among others. They are in general nonlinear equations which can develop singularities in finite time, which make their study both interesting and challenging.

There are different approaches to deal with a geometric evolution equation on a surface rooted in the different ways a surface can be represented (cf. section 3.1). For example we have the graph, parametric, level set and phase field approaches. Each one has its advantages and disadvantages. In this work we are interested in the *parametric approach*.

A typical way to obtain a geometric evolution equation is from a *gradient flow* using the shape derivative of some energy J. More precisely given a Hilbert space $H(\Gamma)$ of functions defined on Γ and the associated scalar product $<,>$ one can define the following problem

$$< \boldsymbol{v}, \boldsymbol{w} >= -dJ(\Gamma; \boldsymbol{w}) \ \forall \boldsymbol{w} \in H(\Gamma).$$

This will give a geometric evolution equation.

4.2.1 Mean Curvature Flow

Flow by mean curvature is given by the geometric law

$$v = -h,$$

that prescribes the normal velocity to be the negative of the total mean curvature. It is the L^2-gradient flow of the area functional $A(\Gamma) = \int_\Gamma 1$. This follows from the definition (4.1) and Theorem 4.1.1 equation 4.5 that states that $\delta A = h$. In applications it represents an interfacial energy. In one dimension, it is referred to as the curve shortening flow. Mathematically, the problem is the following:

Problem 4.2.1 (Mean Curvature Flow: Lagrangian Formulation)**.** Given a reference surface Γ_0 find $\mathbf{x} : \Gamma_0 \times [0,T] \to \mathbb{R}^{d+1}$ such that

$$\begin{aligned} \frac{\partial \mathbf{x}}{\partial t}(\boldsymbol{p}, t) &= -\boldsymbol{h}(\mathbf{x}(\boldsymbol{p}), t) \quad \text{in } \Gamma_0 \times [0,T] \\ \mathbf{x}(\boldsymbol{p}, 0) &= \mathbf{Id}_{\Gamma_0}(\boldsymbol{p}) \quad \text{in } \Gamma_0. \end{aligned} \tag{4.8}$$

Assuming that the evolving surfaces $\{\Gamma_t\}$ are continuous in space and time and that all quantities which we shall use make sense, Problem 4.2.1 can be written in trajectory space $G_T = \{(\boldsymbol{x}, t) : \boldsymbol{x} \in \Omega_t, t \in [0,T]\}$ as:

Problem 4.2.2 (Mean Curvature Flow: Eulerian Formulation)**.** Given an initial surface Γ_0 and $\boldsymbol{u}_0 = \mathbf{Id}_{\Gamma_0}$ find $\boldsymbol{u} : G_T \to \mathbb{R}^{d+1}$ such that $\boldsymbol{u} = \mathbf{Id}_{\Gamma_t}$ and

$$\boldsymbol{v}(\boldsymbol{x}, t) = -\boldsymbol{h}(x, t) \quad \text{in } G_T. \tag{4.9}$$

where $\boldsymbol{v} : G_T \to \mathbb{R}^{d+1} = \dot{\boldsymbol{u}}$, the material derivative of $\boldsymbol{u}$.

Remark 4.2.1 (Special Solutions). A known exact solution for the mean curvature flow is dimension $d + 1$ is the shrinking sphere of initial radius R_0. The radius in time is given by $R(t) = \sqrt{R_0^2 - 2dt}$.

A useful property for numerical tests is the following result

Lemma 4.2.1. *Let $\boldsymbol{v}$ be the solution of Problem 4.2.2. Then*

$$\int_0^T \int_{\Gamma_t} |\boldsymbol{v}|^2 = \text{meas}(\Gamma_0) - \text{meas}(\Gamma_t). \tag{4.10}$$

Remark 4.2.2. Under reasonable hypothesis it was shown that this flow shrinks to a point in finite time [Gig06, page 4]. This is no longer the case if we add a volume constraint. We achieve this using the constraint methods of section 4.1. Another method to attain the volume constraint that appears in the computational literature is rescaling. This one is applied at the discrete level, after each time step the surface is rescaled to have a prescribed volume [DDE05].

Assume for simplicity that Γ_0 has no boundary (and as a consequence so do all Γ_t). A weak formulation for the mean curvature flow can be obtained by multiplying equation (4.9) by a smooth test function $\boldsymbol{\phi}$. Using formula (3.14) and then integrating by parts (Theorem 3.1.4), we obtain

Problem 4.2.3 (Mean Curvature Flow: Weak Form)**.** Given an initial surface Γ_0 and $\boldsymbol{u}_0 = \mathbf{Id}_{\Gamma_0}$, find $\boldsymbol{u} : G_T \to \mathbb{R}^{d+1}$ such that $\boldsymbol{u} = \mathbf{Id}_{\Gamma_t}$ and

$$\int_\Gamma \dot{\boldsymbol{u}} \cdot \boldsymbol{\phi} = -\int_\Gamma \nabla_\Gamma \mathbf{Id} : \nabla_\Gamma \boldsymbol{\phi}, \tag{4.11}$$

for all smooth $\boldsymbol{\phi}$.

Remark 4.2.3 (Discrete Surface). Observe that even though the velocity is normal we use a vector test function $\boldsymbol{\phi}$ in the weak formulation. The reason is that by doing this we get $\boldsymbol{\nu}$ as part of the solution. In the discretization presented in section 6.1.1 this will give a continuous normal, even though the representation of the surface has a discontinuous one because is piecewise polynomial.

Using the method of section 4.1 and Theorem 4.1.1 on Problem 4.2.3, the mean curvature flow with volume constraint can be formulated as follows:

Problem 4.2.4 (Mean Curvature Flow with Volume Constraint)**.** Given an initial surface Γ_0 and $\boldsymbol{u_0} := \mathbf{Id}_{\Gamma_0}$, find $\boldsymbol{u} : G_T \to \mathbb{R}^{d+1}$ and $\pi : [0,T] \to \mathbb{R}$ such that $\boldsymbol{u} = \mathbf{Id}_{\Gamma_t}$ and

$$\begin{gathered} \int_\Gamma \dot{\boldsymbol{u}} \cdot \boldsymbol{\phi} = -\int_\Gamma \nabla_\Gamma \mathbf{Id} : \nabla_\Gamma \boldsymbol{\phi} + \pi \int_\Gamma \boldsymbol{\phi} \cdot \boldsymbol{\nu}, \\ \operatorname{meas}(\Omega_t) = \operatorname{meas}(\Omega_0), \end{gathered} \tag{4.12}$$

for all smooth $\boldsymbol{\phi}$.

4.2.2 Willmore Flow

This is the L^2-gradient flow of the Willmore energy [Wil93]

$$W(\Gamma) = \frac{1}{2}\int_\Gamma h^2. \tag{4.13}$$

Using the tools of shape differential calculus we can compute the shape derivatives and the first variation of the shape functional W.

Lemma 4.2.2. *In 3D the Willmore shape derivative is given by*

$$dW(\Gamma, \boldsymbol{\phi}) = \int_\Gamma \left(-\Delta_\Gamma h - \frac{1}{2}h^3 + 2kh \right) \boldsymbol{\nu} \cdot \boldsymbol{\phi}, \tag{4.14}$$

whereas in 2D it is given by

$$dW(\Gamma, \boldsymbol{\phi}) = \int_\Gamma \left(-\Delta_\Gamma h - \frac{1}{2}h^3 \right) \boldsymbol{\nu} \cdot \boldsymbol{\phi}. \tag{4.15}$$

Proof. From Lemma 3.2.2 we obtain

$$dW(\Gamma, \boldsymbol{\phi}) = -\int_\Gamma h \Delta_\Gamma \phi + \int_\Gamma (h \partial_{\boldsymbol{\nu}} h + \frac{1}{2}h^3)\phi. \tag{4.16}$$

Using Lemma 3.2.4, $\partial_{\boldsymbol{\nu}} h = -h^2 + 2k$, and integrating by parts we deduce

$$dW(\Gamma, \boldsymbol{\phi}) = \int_\Gamma \left(-\Delta_\Gamma h - \frac{1}{2}h^3 + 2kh \right) \boldsymbol{\nu} \cdot \boldsymbol{\phi}, \tag{4.17}$$

which is (4.14). Upon taking $k = 0$ we arrive at (4.15). □

From here it follows that the functional derivative is

$$\delta W = \left(\Delta_\Gamma h + \frac{1}{2}h^3 - 2kh \right) \boldsymbol{\nu}. \tag{4.18}$$

It is also possible to show that

$$\delta W = \left(\Delta_\Gamma (h\boldsymbol{\nu}) - 2\nabla_\Gamma \cdot (h \nabla_\Gamma \boldsymbol{\nu}) + h \nabla_\Gamma h - \frac{1}{2}h^3 \boldsymbol{\nu} \right). \tag{4.19}$$

A geometric evolution equation describing this flow is

Problem 4.2.5 (Willmore Flow: Lagrangian Formulation)**.** Given a reference surface Γ_0, find $\mathbf{x} : \Gamma_0 \times [0,T] \to \mathbb{R}^{d+1}$ such that

$$\begin{aligned} \frac{\partial \mathbf{x}}{\partial t}(\boldsymbol{p},t) &= -\left(\Delta_\Gamma h + \frac{1}{2}h^3 - 2kh\right)\boldsymbol{\nu} \quad \text{in } \Gamma_0 \times [0,T] \\ \mathbf{x}(\boldsymbol{p},0) &= \mathbf{Id}_{\Gamma_0}(\boldsymbol{p}) \quad \text{in } \Gamma_0. \end{aligned} \tag{4.20}$$

Lemma 4.2.3 (Special Solutions)**.** *The sphere is a solution in three dimensions. An expanding circle is a solution in two dimensions.*

Proof. Consider a sphere in $\mathbb{R}^3$ of radius R then we have $h = \frac{2}{R}$ and $k = \frac{1}{R^2}$. Using formula (3.14) and (3.5) with $\boldsymbol{\nu} = \frac{\boldsymbol{x}}{|\boldsymbol{x}|}$ we get $\Delta_\Gamma h = 0$ and plugging this in (4.18) gives $\delta W = 0$, which implies that any sphere is a solution of the Willmore flow.

By the symmetry of the Willmore energy if the initial shape is a circle it should remain a circle. Under this restriction the Willmore flow (4.20) becomes an ODE

$$R'(t) = \frac{1}{2R^3} \tag{4.21}$$

and the solution is $R(t) = (2(t-t_0) + R_0^4)^{1/4}$. □

Lemma 4.2.4 (Rescaling Property in 3D)**.** *If Γ is an equilibrium solution of the Willmore flow in three dimensions, then $\alpha\Gamma = \{\alpha\boldsymbol{x} : \boldsymbol{x} \in \Gamma\}$ is another equilibrium for any $\alpha > 0$.*

Proof. This is a consequence of how the mean curvature rescales by dilations. It is simple to see that $h(\alpha\boldsymbol{x}) = \frac{1}{\alpha}h(\boldsymbol{x})$. Since the surface area is magnified by a factor of α^2 we get

$$W(\alpha\Gamma) = \frac{1}{2}\int_{\alpha\Gamma} h^2 = \frac{1}{2}\int_\Gamma \alpha^2 h^2 \frac{1}{\alpha^2} = W(\Gamma), \tag{4.22}$$

which is the assertion. □

Remark 4.2.4 (Untangling of Curves). For curves the Willmore flow has the unwinding property. This means that self intersecting curves evolved by the Willmore flow tend to finish in a circle like shape [DKS02].

Multiplying (4.18) by a smooth function $\boldsymbol{\phi}$ and using integration by parts, weak formulations for the Willmore energy can be obtained. We consider two of them. The first one is due to Rusu [Rus05].

$$\int_\Gamma \delta W \cdot \boldsymbol{\phi} = -\int_\Gamma \left[\left(I - 2\boldsymbol{\nu}\otimes\boldsymbol{\nu}\right)\nabla_\Gamma \boldsymbol{h}\right] : \nabla_\Gamma \boldsymbol{\phi} + \frac{1}{2}\int_\Gamma |\boldsymbol{h}|^2 \nabla_\Gamma \mathbf{Id} : \nabla_\Gamma \boldsymbol{\phi} \qquad \forall \boldsymbol{\phi}. \quad (4.23)$$

The second one is due to Dziuk [Dzi]

$$\int_\Gamma \delta W \cdot \boldsymbol{\phi} = -\int_\Gamma \nabla_\Gamma \boldsymbol{h} : \nabla_\Gamma \boldsymbol{\phi} + \int_\Gamma \nabla_\Gamma \boldsymbol{h} : [\mathbf{D}\nabla_\Gamma \mathbf{Id}] - \int_\Gamma \nabla_\Gamma \cdot \boldsymbol{h} \nabla_\Gamma \cdot \boldsymbol{\phi} - \frac{1}{2}\int_\Gamma |\boldsymbol{h}|^2 \nabla_\Gamma \cdot \boldsymbol{\phi}, \qquad \forall \boldsymbol{\phi}. \quad (4.24)$$

where $\mathbf{D}(\boldsymbol{\phi})_{ij} = (\nabla_\Gamma)_i \phi^j + (\nabla_\Gamma)_j \phi^i$. This second discretization is claimed to be more stable [Dzi]. We refer to our numerical experiments of chapter 8 that indicate similar performances of both formulations. To complete the previous equations recall from (3.14) that $\boldsymbol{h}$ and $\boldsymbol{x}$ are related by

$$\int_\Gamma \boldsymbol{h} \cdot \boldsymbol{\varphi} = \int_\Gamma \nabla_\Gamma \mathbf{Id} : \nabla_\Gamma \boldsymbol{\varphi} \qquad \forall \boldsymbol{\varphi}, \quad (4.25)$$

which becomes part of the weak formulations. Using either (4.24) or (4.23), the weak form of the Willmore flow becomes:

Problem 4.2.6 (Willmore Flow: Weak Form)**.** Given an initial surface Γ_0 and

$u_0 = \mathbf{Id}_{\Gamma_0}$, find $\boldsymbol{u} : G_T \to \mathbb{R}^{d+1}$ such that $\boldsymbol{u} = \mathbf{Id}_{\Gamma_t}$ and

$$\int_\Gamma \dot{\boldsymbol{u}} \cdot \boldsymbol{\phi} = -\int_\Gamma \delta W \cdot \boldsymbol{\phi}, \tag{4.26}$$

$$\int_\Gamma \boldsymbol{h} \cdot \boldsymbol{\varphi} = \int_\Gamma \nabla_\Gamma \mathbf{Id} : \nabla_\Gamma \boldsymbol{\varphi}, \tag{4.27}$$

for all smooth $\boldsymbol{\phi}$ and $\boldsymbol{\varphi}$.

4.2.3 Biomembranes

The geometric model for biomembranes is given by the Willmore flow (section 4.2.2) when it is subject to surface area and enclosed volume constraints. Also the Willmore energy has a bending rigidity coefficient κ. In this context we refer to the Willmore energy as bending energy (section 2.1). This energy is believed to be the main driving force in biomembranes (see [Hel73, Jen77b] and chapter 2) to such an extent that a minimizer of this energy, in the family of all surfaces that share the same enclosed volume and surface area, has been successful in predicting the shapes of artificial vesicles in the lab and even the shape of a red blood cell. Using the method of section 4.1 and Theorem 4.1.1 we have

Problem 4.2.7 (Geometric Flow for Biomembrane Modeling)**.** Given an initial surface Γ_0 and $u_0 = \mathbf{Id}_{\Gamma_0}$, find $\boldsymbol{u} : G_T \to \mathbb{R}^{d+1}$ such that $\boldsymbol{u} = \mathbf{Id}_{\Gamma_t}$; $\lambda : [0, T] \to \mathbb{R}$

and $\pi : [0,T] \to \mathbb{R}$

$$\int_\Gamma \dot{\boldsymbol{u}} \cdot \boldsymbol{\phi} = -\int_\Gamma \delta W \cdot \boldsymbol{\phi} + \lambda \int_\Gamma \boldsymbol{h} \cdot \boldsymbol{\phi} + \pi \int_\Gamma \boldsymbol{\phi} \cdot \boldsymbol{\nu}, \tag{4.28}$$

$$\int_\Gamma \boldsymbol{h} \cdot \boldsymbol{\varphi} = \int_\Gamma \nabla_\Gamma \mathbf{Id} : \nabla_\Gamma \boldsymbol{\varphi}, \tag{4.29}$$

$$\text{meas}(\Omega_t) = \text{meas}(\Omega_0), \tag{4.30}$$

$$\text{meas}(\Gamma_t) = \text{meas}(\Gamma_0), \tag{4.31}$$

for all smooth $\boldsymbol{\phi}$ and $\boldsymbol{\varphi}$.

4.2.4 Surface Diffusion

In a motion by surface diffusion the normal velocity is proportional to the surface Laplacian of mean curvature:

$$v = \Delta_\Gamma h.$$

In applications in material sciences they are used to model morphological changes in stressed epitaxial films [AT72, SDV93, CT94].

4.3 Fluid-Structure Interaction Model

The idea to couple an immerse membrane (structure) with a fluid is to balance the boundary force of the fluid with the force the membrane exerts on it. Recalling the notation of section 3.3, consider two motions: $\mathbf{x} : \hat{\Omega} \times [0,T] \to \mathcal{E}$ and $\boldsymbol{\chi} : \hat{\Gamma} \times [0,T] \to \mathcal{E}$, where $\hat{\Omega}$ is a reference domain and $\hat{\Gamma}$ a reference surface. Letting $\Omega_t = \mathbf{x}(\hat{\Omega}, t)$ and $\Gamma_t = \boldsymbol{\chi}(\hat{\Gamma}, t)$ we have the corresponding trajectories

$$G_T = \{(\boldsymbol{x}, t) : \boldsymbol{x} \in \Omega_t, t \in [0,T]\} \text{ and } \mathcal{G}_T = \{(\boldsymbol{x}, t) : \boldsymbol{x} \in \Gamma_t, t \in [0,T]\}.$$

For the coupling to take place we will require the following two coupling assumptions:

Assumption 4.3.1. The membrane is immersed, i.e. $\mathcal{G}_T \subset G_T$.

Assumption 4.3.2. The membrane is endowed of some energy E that only depends on geometric quantities of Γ_t (cf. with introduction of section 4.2). The force $\boldsymbol{g}_\Gamma$ the membrane exerts is given by the variational derivative of E, i.e. $\boldsymbol{g}_\Gamma = -\delta E$.

The coupling problem (see figure 4.1) can be stated as follows. Given the initial velocity $\boldsymbol{v}_0$ and domain Ω_0, find the velocity $\boldsymbol{v}$, pressure p and free boundary Γ_t such that:

$$\rho\dot{\boldsymbol{v}} - \nabla \cdot \Sigma = \boldsymbol{b} \qquad \text{in } \Omega_t, \tag{4.32}$$

$$\nabla \cdot \boldsymbol{v} = 0 \qquad \text{in } \Omega_t, \tag{4.33}$$

$$[\boldsymbol{\Sigma}]\boldsymbol{\nu} = \delta E \qquad \text{on } \Gamma_t, \tag{4.34}$$

$$\boldsymbol{v} = \boldsymbol{\vartheta} \qquad \text{on } \Gamma_t, \tag{4.35}$$

$$\boldsymbol{v}(\cdot, 0) = \boldsymbol{v}_0 \qquad \text{in } \Omega_0, \tag{4.36}$$

where $[\boldsymbol{\Sigma}] = \boldsymbol{\Sigma}_{\text{in}} - \boldsymbol{\Sigma}_{\text{out}}$ is the jump of the stress tensor and $\boldsymbol{\vartheta}$ is the membrane velocity. Equation (4.35) represents the *adherence* or *no slip condition*. Figure 4.2 gives and explanation for (4.34). Assumptions 4.3.1 and 4.3.2 are necessary for (4.34) to make sense.

In our applications it is always possible to define a constant α such that with the definitions (3.31), the problem (4.32)-(4.36) can be written in dimensionless

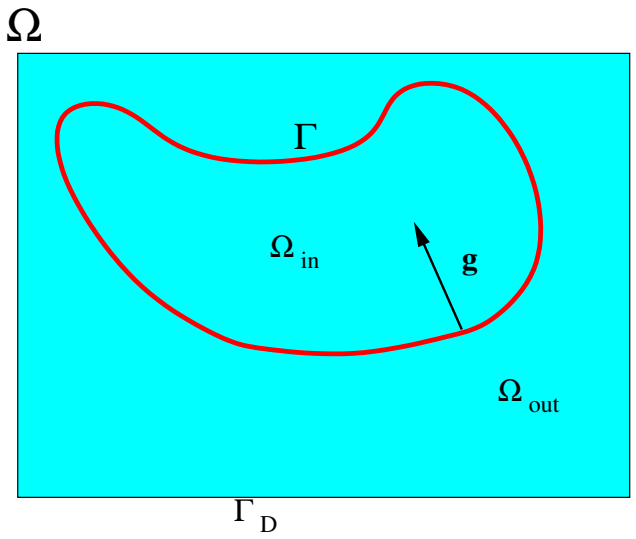


Figure 4.1: Schematic picture for the coupling problem. Ω is a fluid domain being split into two components by the immersed membrane Γ. The membrane exerts a force $\boldsymbol{g}$ to the fluid, which is the by-product of an endowed energy that only depends on geometric quantities of Γ. Γ_D is the (exterior) boundary of Ω that could be considered to have zero velocity.

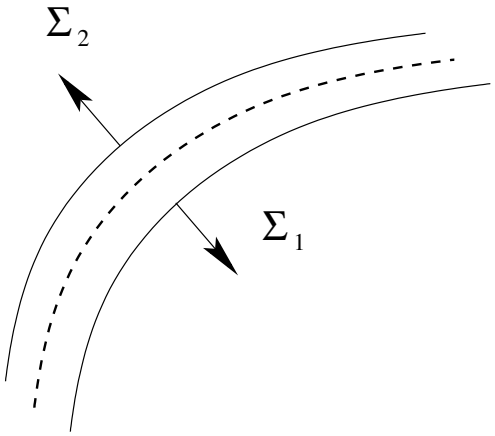


Figure 4.2: Justification of equation (4.34). We take a pillbox around the dotted membrane. The total force in the pillbox when the thickness goes to zero as the volume goes to zero is $-\boldsymbol{\Sigma}_2\boldsymbol{\nu}_2 - \boldsymbol{\Sigma}_1\boldsymbol{\nu}_1 = \boldsymbol{g}_\Gamma$. Let $\boldsymbol{\nu} = \boldsymbol{\nu}_2 = -\boldsymbol{\nu}_1$, then we get $(\boldsymbol{\Sigma}_2 - \boldsymbol{\Sigma}_1)\boldsymbol{\nu} = -\boldsymbol{g}_\Gamma$.

variables as:

$$\dot{\bar{\boldsymbol{v}}} - \nabla \cdot \bar{\boldsymbol{\Sigma}} = \bar{\boldsymbol{b}} \qquad \text{in } \bar{\Omega}, \tag{4.37}$$

$$\nabla \cdot \bar{\boldsymbol{v}} = 0 \qquad \text{in } \bar{\Omega}, \tag{4.38}$$

$$\alpha Re[\bar{\boldsymbol{\Sigma}}]\boldsymbol{\nu} = \delta E \qquad \text{on } \bar{\Gamma}_t, \tag{4.39}$$

$$\bar{\boldsymbol{v}} = \bar{\boldsymbol{\vartheta}} \qquad \text{on } \bar{\Gamma}_t, \tag{4.40}$$

$$\bar{\boldsymbol{v}}(\cdot, 0) = \bar{\boldsymbol{v}}_0 \qquad \text{in } \bar{\Omega}_0. \tag{4.41}$$

From now on, unless explicitly stated, we will drop the bar and work with the dimensionless system. Also we assume that Ω_2 is vacuum so that $[\boldsymbol{\Sigma}] = \boldsymbol{\Sigma}_{in}$ and use $\boldsymbol{\Sigma} = \boldsymbol{\Sigma}_{in}$. If we test (4.37) with a smooth test function $\boldsymbol{\phi}$ in $\Omega(t)$ we get

$$\begin{aligned} \int_\Omega \dot{\boldsymbol{v}} \cdot \boldsymbol{\phi} - \int_\Omega \boldsymbol{\phi} \cdot \boldsymbol{b} &= \int_\Omega \boldsymbol{\phi} \cdot \nabla \cdot \boldsymbol{\Sigma} \\ &= \int_\Omega \nabla \cdot (\boldsymbol{\Sigma}\boldsymbol{\phi}) - \int_\Omega \boldsymbol{\Sigma} : \nabla \boldsymbol{\phi} \\ &= \int_\Gamma (\boldsymbol{\Sigma}\boldsymbol{\phi}) \cdot \boldsymbol{\nu} - \int_\Gamma \boldsymbol{\Sigma} : \nabla \boldsymbol{\phi} \end{aligned} \tag{4.42}$$

For the first integral of the last term in (4.42) we use (4.39) to get

$$\int_\Gamma (\boldsymbol{\Sigma}\boldsymbol{\phi}) \cdot \boldsymbol{\nu} = \int_\Gamma (\boldsymbol{\Sigma}\boldsymbol{\nu}) \cdot \boldsymbol{\phi} = \frac{1}{\alpha Re} \int_\Gamma \delta E \cdot \boldsymbol{\phi}. \tag{4.43}$$

For the second integral after integration by parts

$$-\int_\Omega \boldsymbol{\Sigma} : \nabla \boldsymbol{\phi} = \int_\Omega p \, \nabla \cdot \boldsymbol{\phi} - \frac{1}{Re} D(\boldsymbol{v}) : \nabla \boldsymbol{\phi}. \tag{4.44}$$

Combining the last equations

$$\int_\Omega \dot{\boldsymbol{v}}(t) \cdot \boldsymbol{\phi} + \int_\Omega \frac{1}{Re} D(\boldsymbol{v}) : \nabla \boldsymbol{\phi} - \int_\Omega p \, \nabla \cdot \boldsymbol{\phi} = \int_\Omega \boldsymbol{\phi} \cdot \boldsymbol{b} - \frac{1}{\alpha Re} \int_\Gamma \delta E \cdot \boldsymbol{\phi}. \tag{4.45}$$

Equation (4.45) provides the starting point for the discrete formulations of chapter 6. Next we specialize this equation to the capillarity, Willmore and biomembrane forces deriving in each the value of α.

4.3.1 Capillarity

In this case α is the well known capillary number

$$Ca = \frac{\mu V}{\sigma},$$

where μ is the viscosity of the liquid, V is a characteristic velocity and σ is the surface or interfacial tension between the two fluid phases. See for example [Bän01]. In this case the energy E is the area functional, which enters the momentum equation (4.45) as $\delta E = \boldsymbol{h}$ (see Section 4.2.1).

4.3.2 Willmore

The coupled Willmore problem is by definition the replacement of E in equation (4.34) by W (equation (4.13)), which gives

$$\begin{aligned}
\rho\dot{\boldsymbol{v}} - \nabla\cdot\Sigma &= \boldsymbol{b} && \text{in } \Omega_t,\\
\nabla\cdot\boldsymbol{v} &= 0 && \text{in } \Omega_t,\\
[\boldsymbol{\Sigma}]\boldsymbol{\nu} &= \delta W && \text{on } \Gamma_t,\\
\boldsymbol{v} &= \boldsymbol{\vartheta} && \text{on } \Gamma_t,\\
\boldsymbol{v}(\cdot,0) &= \boldsymbol{v}_0 && \text{in } \Omega_0.
\end{aligned} \tag{4.46}$$

To write a non dimensional formulation of problem (4.46), we need to see how the curvature rescales. Using the definitions (3.31) it easy to see that

$$\begin{aligned} h(\bar{\boldsymbol{x}}) &= Lh(\boldsymbol{x}), \\ k(\bar{\boldsymbol{x}}) &= L^2 k(\boldsymbol{x}), \\ \Delta_{\bar{\boldsymbol{x}}} h(\bar{\boldsymbol{x}}) &= L^3 \Delta_{\boldsymbol{x}} h(\boldsymbol{x}). \end{aligned} \tag{4.47}$$

Using (4.47) in equation (4.18) gives

$$\delta W(\boldsymbol{x}) = \frac{1}{L^3}\left(\Delta_{\bar{\boldsymbol{x}}} h(\bar{\boldsymbol{x}}) + \frac{1}{2}h(\bar{\boldsymbol{x}})^3 - 2k(\bar{\boldsymbol{x}})h(\bar{\boldsymbol{x}})\right)\boldsymbol{\nu}(\bar{\boldsymbol{x}}). \tag{4.48}$$

Then equations (3.32), (3.33) and problem (4.46) yield the dimensionless coupled Willmore problem:

$$\begin{aligned} \dot{\bar{\boldsymbol{v}}} - \nabla \cdot \bar{\Sigma} &= \bar{\boldsymbol{b}} && \text{in } \bar{\Omega}, \\ \nabla \cdot \bar{\boldsymbol{v}} &= 0 && \text{in } \bar{\Omega}, \\ \rho V^2 L^3 \bar{\boldsymbol{\Sigma}} \boldsymbol{\nu} &= \left(\Delta_\Gamma h + \frac{1}{2}h^3 - 2kh\right)\boldsymbol{\nu} && \text{on } \bar{\Gamma}_t. \end{aligned} \tag{4.49}$$

As $\rho V^2 L^3 = Re\mu L^2 V$ then (4.49) gives $\alpha = \mu L^2 V$.

4.3.3 Biomembranes

Recall from section 4.2.3 that this is the coupled Willmore problem (4.49) with area, volume constraints and a bending rigidity coefficient. As the volume is preserved due to the incompressibility of the fluid, what remains to do is to add a Lagrange multiplier for the area. Using the result obtain in (4.49) and taking into account the bending rigidity coefficient κ we arrive at the dimensionless model for the fluid

biomembrane in terms of the Reynolds and bending numbers:

$$\begin{aligned} \dot{\bar{\boldsymbol{v}}} - \nabla \cdot \bar{\Sigma} &= \bar{\boldsymbol{b}}, \\ \nabla \cdot \bar{\boldsymbol{v}} &= 0, \\ ReBe\bar{\boldsymbol{\Sigma}}\boldsymbol{\nu} &= \left(\Delta_{\bar{\boldsymbol{x}}} h(\bar{\boldsymbol{x}}) + \frac{1}{2}h^3 - 2kh + \lambda h \right) \boldsymbol{\nu}, \end{aligned} \tag{4.50}$$

where the bending number is

$$Be = \frac{\mu V L^2}{\kappa}.$$

Chapter 5

Finite Element Method

The finite element method is one of the most successful tools to approximate solutions of PDEs. The theory of the method has reached a maturity level for fixed domains, as it is reflected by the number of advanced books on the subject [Cia78, BS02, AO00, BS01]. Still the theory for moving finite elements, in particular when the domain is part of the unknown is lacking behind. Some references on the latter can be found in [Bän01, BG04, FN04, DE07, BMN05]. This is the case of problems presented in Chapter 4.

First in Section 5.1 we gather some classical results of the finite element method in flat domains that are used later to define finite elements for surfaces and moving domains. Except for a slight change in notation most concepts are extracted from [Cia78, BS02]. In Section 5.2 we provide the concepts of surface finite elements and the basic tools to work with them. These concepts get applied in the subsequent sections 5.3, 5.4 and 5.5 to obtain interpolation results for surfaces, discrete formulas for curvature and a priori estimates for the Laplace-Beltrami operator. The proofs of the latter will be crucial in the proof of the refinement result (Section 7.2). The first a priori estimate for Laplace-Beltrami operator appears in [Dzi88], a posteriori results can be found in [Mek05, DD07, MMN] for piecewise linear elements. A priori estimates for higher order elements can be found in a preprint by Demlow [Dem].

The approach in Section 5.2 is different from the last in the sense that we do not use the distance function and different from [Mek05] because we do it for any order isoparametric representation of the surface. In Section 5.6 we present a result for surface quadratic isoparametric elements that will justify some methods of Chapter 7. We finish the chapter with Section 5.7 where we present the parametric finite element method for geometric evolution equations.

5.1 Finite Elements for Flat Domains

We recall the finite element method for the Poisson equation. When this problem is posed in variational form it can be stated as follows: given a domain Ω and a source $f \in L^2(\Omega)$ find $u \in \mathbb{V} = H_0^1(\Omega)$ such that

$$a(u,v) := \int_\Omega \nabla u \nabla v = \int_\Omega fv \quad \forall v \in \mathbb{V}.$$

The conforming Galerkin method for approximating the solution u consists of defining approximate problems in finite dimensional subspaces of $\mathbb{V}$. The finite element method is a specific process to construct finite dimensional subspaces $\mathbb{V}_h$ of $\mathbb{V}$ called finite element spaces. It is formally characterized by three basic aspects:

1. an underlying triangulation $\mathcal{T}$ of the domain;

2. piecewise polynomial elements in $\mathbb{V}_h$;

3. existence of basis functions with small support.

Formally the finite dimensional space $\mathbb{V}_h$ is defined as the set of functions $v \in \mathbb{V}$ such that $v|_K$ is a polynomial for each $K \in \mathcal{T}$. The typical finite element mesh-space pair

is a triangulated domain into simple elements such that a function in the space is a polynomial when restricted to each element of the triangulation. Very important is the existence of points in the triangles (degrees of freedom) such that a function in $\mathbb{V}_h$ is uniquely determined by its values at these points. We proceed to make the previous ideas rigorous.

Definition 5.1.1 (Triangulation)**.** Given a domain Ω a conforming triangulation $\mathcal{T}$ is a finite family of subsets $K \subset \Omega$ such that:

1. $\bar{\Omega} = \bigcup_{K \in \mathcal{T}} K$,
2. K is closed and with nonempty interior for all $K \in \mathcal{T}$,
3. $K_1^\circ \bigcap K_2^\circ = \emptyset$ for all $K_1, K_2 \in \mathcal{T}$,
4. ∂K is Lipschitz-continuous for all $K \in \mathcal{T}$,
5. Any face[1] of any $K \in \mathcal{T}$ is either a subset of the boundary or a face of another $K_2 \in \mathcal{T}$.

Remark 5.1.1 (Conformity property). Property 5 requires the set K to have faces, which is clear for simplices (definition 5.1.3 below). It also makes sense for other sets that are deformation of simplices as isoparametric elements.

Definition 5.1.2 (Finite Element)**.** Following Ciarlet [Cia78] a finite element in $\mathbb{R}^{d+1}$ is a triplet $(K, \mathcal{P}, \mathcal{N})$ where:

1. K is a closed subset of $\mathbb{R}^{d+1}$ with non empty interior and Lipschitz continuous boundary,

[1] See Remark 5.1.1.

2. $\mathcal{P}$ is a finite dimensional space of real valued functions (typically polynomials) defined on K,

3. $\mathcal{N}$ is a finite basis of $\mathcal{P}'$, the dual of $\mathcal{P}$.

Of extreme importance is the case when K is a simplex.

Definition 5.1.3 (n-simplex in $\mathbb{R}^{d+1}$)**.** Given $A = \{\boldsymbol{a}_0, \ldots, \boldsymbol{a}_n\}$ a set of $n+1$ points in $\mathbb{R}^{d+1}$, the set $K = \text{conv}(A)$ is known as the n-simplex spanned by A.

An n-simplex is nondegenerate if the set $\{\boldsymbol{a}_1 - \boldsymbol{a}_0, \ldots, \boldsymbol{a}_n - \boldsymbol{a}_0\}$ is linearly independent. A sub-simplex of a simplex K is the convex hull of any subset of A. If $d = 2$ and $n = 2$ a non-degenerate simplex is a triangle and its sub-simplices are its edges (n=1) and vertices (n=0). The size and shape of a $(d+1)$-simplex are characterized by the following quantities

$$\begin{aligned} h_K &:= \text{diam}(K), \\ \rho_K &:= \sup\{\text{diam}(B) : B \subset K, B \text{ is a ball}\}, \\ \sigma_K &:= \frac{h_K}{\rho_K}. \end{aligned} \tag{5.1}$$

Also useful are the associated global quantities $h := \max_{\{K \in \mathcal{T}\}}\{h_K\}$ and $\sigma := \sup_{\{K \in \mathcal{T}\}}\{\sigma_K\}$.

Given a finite element $(\hat{K}, \hat{\mathcal{P}}, \hat{\mathcal{N}})$, and given a function $v : K \to \mathbb{R}$, sufficiently smooth so that the degrees of freedom $\phi_i(v)$ are well defined, we let

$$\mathcal{I}_K v = \Sigma \phi_i(v) p_i,$$

denote the $\mathcal{P}$-*interpolant* of v.

Approximation results are an essential part of the finite element method theory, quite general results can be found for interpolation error in Sobolev spaces for domains in $\mathbb{R}^{d+1}$, later we see how these results can be extended to surfaces. Next we restrict to the case when the degree of freedoms are given by $\phi(p) = p(\boldsymbol{a}_i)$ with $\boldsymbol{a}_i \in K$.

Definition 5.1.4 (Affine Equivalent)**.** Two finite elements $(\hat{K}, \hat{\mathcal{P}}, \hat{\mathcal{N}})$ and $(K, \mathcal{P}, \mathcal{N})$ are affine equivalent if there exists an invertible affine mapping:

$$F(\hat{\boldsymbol{x}}) = \mathbf{B}\hat{x} + \boldsymbol{b} \tag{5.2}$$

such that

$$K = F(\hat{K}) \tag{5.3}$$

$$\mathcal{P} = \{p : K \to \mathbb{R} \ : \ p = \hat{p} \circ F^{-1}, \hat{p} \in \hat{\mathcal{P}}\} \tag{5.4}$$

$$a_i = F(\hat{a}_i) \tag{5.5}$$

Theorem 5.1.1. *Let $(\hat{K}, \hat{\mathcal{P}}, \hat{\mathcal{N}})$ be a finite element satisfying the following inclusions for some integers $m \geq 0$ and $k \geq 0$ and for some numbers $p, q \in [1, \infty]$,*

1. $W_p^{k+1}(\hat{K}) \subset C^0(\hat{K})$,
2. $W_p^{k+1}(\hat{K}) \subset W_q^m(\hat{K})$,
3. $\mathcal{P}^k(\hat{K}) \subset \hat{\mathcal{P}} \subset W_q^m(\hat{K})$.

Then there exists a constant $C(\hat{K}, \hat{\mathcal{P}}, \hat{\mathcal{N}})$ such that, for all affine equivalent finite elements $(K, \mathcal{P}, \mathcal{N})$ and all functions $v \in W^{k+1,p}(K)$

$$|v - \mathcal{I}_K v|_{m,q,K} \leq C(\hat{K}, \hat{\mathcal{P}}, \hat{\mathcal{N}}) \operatorname{meas}(K)^{(1/q)-(1/p)} \frac{h_K^{k+1}}{\rho_K^m} |v|_{k+1,p,K}. \tag{5.6}$$

Proof. See [Cia78] Theorem 3.1.5. □

5.1.1 Some Finite Elements

Here we define some examples of finite elements that will be used later, the Lagrange finite elements, the "mini" element and the Taylor-Hood elements.

Definition 5.1.5 (Lagrange nodes of order k on K)**.** If K is a simplex spanned by $\{\boldsymbol{a}_1, \ldots, \boldsymbol{a}_{n+1}\}$, and $k \in \mathbb{N}$, then we define the Lagrange nodes on K by

$$L_k(K) := \left\{ \boldsymbol{x} = \Sigma_{l=1}^{n+1} \lambda_l \boldsymbol{a}_l;\ \Sigma_{l=1}^{n+1} \lambda_l = 1, \lambda_l \in \{0, 1/k, \ldots, k/k\}, 1 \leq l \leq n+1 \right\}.$$

Lemma 5.1.2. *Let K be a simplex, $k \in \mathbb{N}$ and $p \in \mathcal{P}^k$. Then p is uniquely identified by its value at the Lagrange nodes $L_k(K)$.*

Proof. See [Cia78] page 49. □

Theorem 5.1.3. *Let $k \in \mathbb{N}$, $\mathcal{T}$ be a triangulation of Ω by simplices, and*

$$\mathbb{X}_h^k(\Omega) = \{\phi_h \in C^0(\bar{\Omega}) : \phi_h|_K \in \mathcal{P}^k(K), K \in \mathcal{T}\}$$

Define

$$L_k = \bigcup_{K \in \mathcal{T}} L_k(K)$$

as the global Lagrange nodes of order k for the triangulation $\mathcal{T}$. Then a function $\phi_h \in \mathbb{X}_h^k(\Omega)$ is uniquely determined by the values in the nodes $N \in L_k$.

The Taylor-Hood elements were first proposed in [TH73].

Definition 5.1.6 (Taylor-Hood Element)**.** Let $k \in \mathbb{N}$. Define

$$\mathbb{V}_h^k = (\mathbb{X}_h^{k+1})^{d+1}$$

$$\mathbb{Q}_h^k = \mathbb{X}_h^k,$$

as the Taylor-Hood element space of order k.

The MINI element was first propose in [ABF84] and consists of continuous piecewise linear elements enriched with bubble functions, also see [GR86, BF91].

Definition 5.1.7 (MINI Element)**.** Let $k \in \mathbb{N}$. Define $\mathbb{B}^k = \{v : v|_K = \alpha(K)\lambda_1 \dots \lambda_{d+1}\}$ then the MINI element is given by

$$\mathbb{V}_h = (\mathbb{X}_h^1)^{d+1} \oplus (\mathbb{B}^{d+2})^{d+1}$$

$$\mathbb{Q}_h = \mathbb{X}_h^1.$$

The most significant property of the Taylor-Hood and MINI element is that they satisfies a discrete LBB condition for the Stokes problem.

5.2 Finite Elements for Surfaces

Throughout this section Γ denotes a compact, oriented, smooth d-hypersurface in $\mathbb{R}^{d+1}$ with or without boundary (see Section 3.1).

Definition 5.2.1 (Polyhedral surface)**.** A pair $(\tilde{\Gamma}, \tilde{\mathcal{T}})$ is a polyhedral surface if $\tilde{\Gamma} \subset \mathbb{R}^{d+1}$ and $\tilde{\mathcal{T}}$ is a finite family of closed, non degenerate, d-simplices in $\mathbb{R}^{d+1}$ such that:

- the intersection of two simplices in the family is either empty or a $(d-k)$-dimensional sub-simplex of both simplices with $k = 1, \ldots, d$, and
- $\tilde{\Gamma} = \bigcup_{\tilde{K} \in \tilde{\mathcal{T}}} \tilde{K}$.

Remark 5.2.1 (Surface triangulation). The previous definition of $\tilde{\mathcal{T}}$ is the right extension of definition 5.1.1 to define a surface triangulation.

Definition 5.2.2 (Polyhedral Approximation. Lift)**.** A polyhedral surface $(\tilde{\Gamma}, \tilde{\mathcal{T}})$ is an approximation to Γ if there exists a continuous bijection $\boldsymbol{l} : \tilde{\Gamma} \to \Gamma$ such that $\boldsymbol{l}|_{\tilde{K}} : \tilde{K} \to \Gamma$ is smooth for all $\tilde{K} \in \tilde{\mathcal{T}}$. Then a polyhedral approximation to Γ is the triplet $(\tilde{\Gamma}, \tilde{\mathcal{T}}, \boldsymbol{l})$. The function $\boldsymbol{l}$ is called the lift.

Observe for example that with these definitions the two polygonal approximations shown in figure 5.1 are different even though $\tilde{\Gamma}$ is the same.

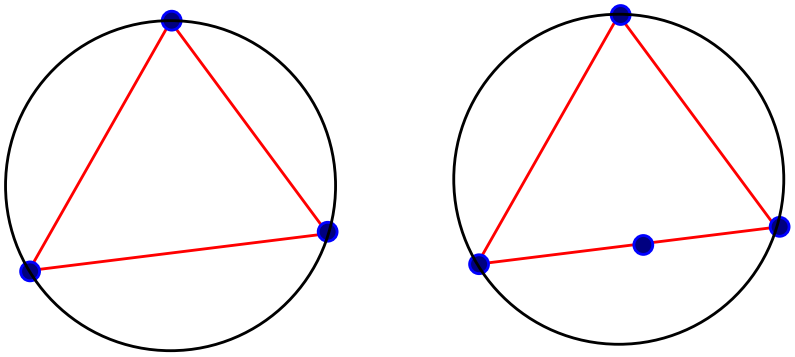


Figure 5.1: In this picture the hyper-surface Γ is the circle, a polygonal $\tilde{\Gamma}$ is the triangle. These approximations are different as $\tilde{\mathcal{T}}$ on the left has three elements and $\tilde{\mathcal{T}}$ on the right has four.

If for example the polyhedral surface $(\tilde{\Gamma}, \tilde{\mathcal{T}})$ is within a small tubular neighborhood of Γ where the distance function is smooth then a polyhedral approximation is

defined by taking $\boldsymbol{l}(\boldsymbol{x}) = \boldsymbol{x} + d(x)\boldsymbol{\nu}(\boldsymbol{y})$ where $\boldsymbol{y} = \boldsymbol{y}(\boldsymbol{x})$ is the orthogonal projection of $\boldsymbol{x}$ onto Γ.

Remark 5.2.2. Using the polyhedral approximation approach in the sense of Definition 5.2.2 is more general than using the distance function. In particular there are instances, both theoretical and computational, when the distance function is not available. See for instance the work of Mekchay [Mek05].

Having the polyhedral approximation $(\tilde{\Gamma}_h, \tilde{\mathcal{T}}, \boldsymbol{l})$ we can define higher order piecewise polynomial approximations to Γ in the following isoparametric way (refer to figure 5.2). Let $\hat{K} \subset \mathbb{R}^d$ be the reference element spanned by the canonical

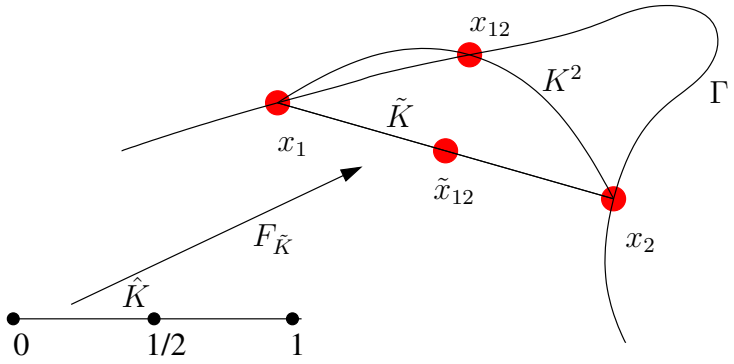


Figure 5.2: Relation among Γ, Γ_h^1 and Γ_h^2. The picture shows an illustrative situation showing the reference simplex $\hat{K}$ the affine simplex $\tilde{K} = K^1$ and the "quadratic simplex" K^2.

basis $\{\boldsymbol{e}_1, \ldots, \boldsymbol{e}_d\}$ and $F_{\tilde{K}} : \hat{K} \to \tilde{K}$ be the unique injective affine map such that the vertices of $\hat{K}$ are mapped to the vertices of $\tilde{K}$. Given a positive integer γ we define $F_{K^\gamma} : \hat{K} \to \mathbb{R}^{d+1}$ as the unique $\mathbb{R}^{d+1}$-valued polynomial of degree γ such that $F_{K^\gamma}(\hat{x}_i) = \boldsymbol{l}(F_{\tilde{K}}(\hat{x}_i))$ for all Lagrange nodes $\hat{x}_i$ of degree γ (definition 5.1.5). This defines for each γ a piecewise polynomial surface Γ_h^γ with the associated curved triangulation $\mathcal{T}^\gamma$ with the curved elements K^γ. A finite element space on Γ_h^γ can

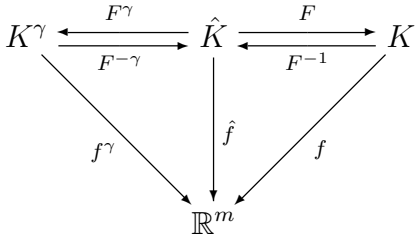
then be defined as follows

$$\mathbb{S}_h^{\gamma,k} = \{\phi \in C^0(\Gamma_h^\gamma) : \phi \circ F_{K^\gamma} \in \mathbb{P}^k(\hat{K})\}, \tag{5.7}$$

where $\mathbb{P}^k(\hat{K})$ is the set of polynomial of degree less or equal k in d variables. Observe that we have a degree γ for the mesh Γ_h^γ and a degree k for the polynomials, we will say that Γ_h^γ is a mesh of degree γ.

The polyhedral approximation $(\tilde{\Gamma}_h, \tilde{\mathcal{T}}, \boldsymbol{l})$ also provides a local parametric representation of both Γ_h^γ and Γ. More precisely, the parametric representation is given by the family $\{F_{K^\gamma} : \hat{K} \to K^\gamma : K^\gamma \in \mathcal{T}^\gamma\}$. With the convention that γ can be "nil", and $F_K := \boldsymbol{l} \circ F_{\tilde{K}}$ the previous family includes a parametrization of Γ as a particular case.

Next we obtain formulas to relate differentials over different domains. They are the main tool to work with finite elements on surface as they allow one to transform the problem to a flat reference element. Before proceeding further recall the definitions of the geometric quantities $\mathbf{T}$, $\mathbf{G}$, $\mathbf{D}$ and q given in Section 3.1. Let $f : K \to \mathbb{R}^m$ be a given function, then $\hat{f}$ and f^γ will denote the corresponding functions defined by the commutative diagram (5.8). Note that to simplify notation below we use $F^{-\gamma}$ for $(F^\gamma)^{-1}$.



(5.8)

From Remark 3.1.5

$$\nabla_{K^\gamma} f^\gamma = (\mathbf{D}^\gamma \nabla_{\hat{K}} \hat{f}) \circ F^{-\gamma} \tag{5.9}$$

and inverting this equation through (3.1) we get

$$\nabla_{\hat{K}} \hat{f} = (\mathbf{T}^\gamma)^\mathsf{T} (\nabla_{K^\gamma} f^\lambda \circ F^\gamma). \tag{5.10}$$

Combining the last two equations we obtain

$$\nabla_K f = \left(\mathbf{D} (\mathbf{T}^\gamma)^\mathsf{T} \nabla_{K^\gamma} f^\lambda \right) \circ \boldsymbol{l}^{-\gamma} \quad \text{and} \quad \nabla_{K^\gamma} f^\lambda = \left(\mathbf{D}^\gamma (\mathbf{T})^\mathsf{T} \nabla_K f \right) \circ \boldsymbol{l}^\gamma. \tag{5.11}$$

where $\boldsymbol{l}^\gamma = F \circ F^{-\gamma}$. Also the integration change of variables is given by

$$\int_{K^\gamma} f^\gamma = \int_{\hat{K}} \hat{f} q^\gamma \qquad \int_K f = \int_{K^\gamma} f^\gamma \delta^\gamma \tag{5.12}$$

where $\delta^\gamma = \frac{q}{q^\gamma} \circ \boldsymbol{F}^{-\gamma}$ and recalling definitions of Section 3.1 $q^\gamma = \sqrt{\det(\mathbf{G}^\gamma)}$.

5.3 Interpolation Results for Surfaces

In this section we extend Theorem 5.1.1 to a surface finite element. Then the result is applied to the surface approximation, This will be used to define the estimator for refining and coarsening in Section 7.2.

Similarly to the flat case, we can say that a surface finite element $(K, \mathcal{P}, \mathcal{N})$ is affine equivalent to a flat finite element $(\hat{K}, \hat{\mathcal{P}}, \hat{\mathcal{N}})$ if the function F of Definition 5.1.4 satisfies that $\mathbf{B}^\mathsf{T}\mathbf{B}$ is invertible. Using the equivalence of tangential derivatives for rigid deformations of Lemma 3.1.8 we can extend Theorem 5.1.1 to a surface finite element. It all consists in changing the partial derivatives by tangential ones.

Corollary 5.3.1. *Let $(\hat{K}, \hat{\mathcal{P}}, \hat{\mathcal{N}})$ be a finite element satisfying the following inclusions for some integers $m \geq 0$ and $k \geq 0$ and for some numbers $p, q \in [1, \infty]$,*

1. $W_p^{k+1}(\hat{K}) \subset C^0(\hat{K})$,

2. $W_p^{k+1}(\hat{K}) \subset W_q^m(\hat{K})$,

3. $\mathcal{P}^k(\hat{K}) \subset \hat{\mathcal{P}} \subset W_q^m(\hat{K})$.

Then there exists a constant $C(\hat{K}, \hat{\mathcal{P}}, \hat{\mathcal{N}})$ *such that, for all affine equivalent finite elements* $(K, \mathcal{P}, \mathcal{N})$ *and all functions* $v \in W^{k+1,p}(K)$

$$|v - \mathcal{I}_K v|_{m,q,K} \leq C(\hat{K}, \hat{\mathcal{P}}, \hat{\mathcal{N}}) \operatorname{meas}(K)^{(1/q)-(1/p)} \frac{h_K^{k+1}}{\rho_K^m} |v|_{k+1,p,K}. \tag{5.13}$$

Proof. Use Lemma 3.1.8 and Theorem 5.1.1. □

Lemma 5.3.2 (Surface Approximation)**.** *Let* $(\tilde{\Gamma}_h, \tilde{\mathcal{T}}, \boldsymbol{l})$ *be a polyhedral approximation. Then*

$$|\boldsymbol{l} - \boldsymbol{l}^\gamma|_{m,q,K} \leq C(\hat{K}, \hat{\mathcal{P}}, \hat{\mathcal{N}}) \operatorname{meas}(K)^{(1/q)-(1/p)} \frac{h_K^{k+1}}{\rho_K^m} |\boldsymbol{l}|_{k+1,p,K}.$$

5.4 Discrete Curvature Computations

From equations (3.15) and (3.16) we obtain discrete formulas for curvatures. Let $\boldsymbol{h}_h$ and $\boldsymbol{\Pi}_h^j$ in $(\mathbb{S}_h^{\gamma,k})^{d+1}$ be functions that satisfy

$$\int_{\Gamma_h} \boldsymbol{h}_h \cdot \boldsymbol{w}_h = \int_{\Gamma_h} \nabla_{\Gamma_h} \boldsymbol{x}_h : \nabla_{\Gamma_h} \boldsymbol{w}_h, \tag{5.14}$$

$$\int_{\Gamma_h^\gamma} \boldsymbol{\Pi}_h^j \cdot \boldsymbol{w}_h = -\int_{\Gamma_h^\gamma} \boldsymbol{\nu}^j \nabla_{\Gamma_h} \cdot \boldsymbol{w}_h, - \int_{\Gamma_h^\gamma} \boldsymbol{\nu}^j \boldsymbol{h}_h \cdot \boldsymbol{w}_h, \tag{5.15}$$

for all $\boldsymbol{w}_h \in (\mathbb{S}_h^{\gamma,\gamma})^{d+1}$, then the following error formulas [Hei] are obtained.

Theorem 5.4.1. *Let* Γ *be a hypersurface of class* C^m*,* $m > 3$*, without boundary and let* $(\tilde{\Gamma}, \tilde{\mathcal{T}}, \boldsymbol{l})$ *be a polyhedral approximation to* Γ*, such that* $\tilde{\mathcal{T}}$ *is a quasi uniform*

triangulation, Then for $1 \leq \gamma < m$*, we have*

$$\|\boldsymbol{h} - \boldsymbol{h}_h\|_{L^2(\Gamma_h^\gamma)} \leq ch^{\gamma-1}; \tag{5.16}$$

$$\|\boldsymbol{\Pi} - \boldsymbol{\Pi}_h\|_{L^2(\Gamma_h^\gamma)} \leq ch^{\gamma-1}. \tag{5.17}$$

where $c = c(\sigma, \Gamma, \gamma)$*.*

Proof. See [Hei]. □

Remark 5.4.1. Note that in the one dimensional case the nodal interpolant is equivalent to the elliptic projection and ensures the convergence of the curvature even with piecewise linear element; compare to (5.16). This is not the case in higher dimension.

5.5 Finite Elements for the Laplace-Beltrami Equation

Here we study the Laplace-Beltrami Equation. The proofs presented are the key for proofs of geometric refinement in Section 7.2.2. Given a closed smooth surface Γ and $f \in L^2(\Gamma)$ with integral zero, $u : \Gamma \to \mathbb{R}$ is a solution to the Laplace-Beltrami equation if

$$\int_\Gamma \nabla_\Gamma u \nabla_\Gamma \phi = \int_\Gamma f\phi \qquad \forall \phi \in H^1(\Gamma), \tag{5.18}$$

Given a scalar α it can be shown [Aub98] that the previous equation has a unique solution u with mean value α.

If $\tilde{\Gamma}$ is a polyhedral approximation to Γ in the sense of Section 5.2, then given γ and k, a finite element approximation to equation (5.18) is given by

$$\int_{\Gamma^\gamma} \nabla_{\Gamma^\gamma} {u_h}^\gamma \nabla_{\Gamma^\gamma} \phi^\gamma = \int_{\Gamma^\gamma} {f_h}^\gamma \phi^\gamma \qquad \forall \phi^\gamma \in \mathbb{S}_h^{\gamma,k}(\Gamma^\gamma) \tag{5.19}$$

where ${f_h}^\gamma$ is some approximation to f^γ. For simplicity of notation let us consider the bilinear form $B_{\Gamma^\gamma} : H^1(\Gamma^\gamma) \times H^1(\Gamma^\gamma) \to \mathbb{R}$ defined by $B_{\Gamma^\gamma}(u^\gamma, w^\gamma) = \int_{\Gamma^\gamma} \nabla_{\Gamma^\gamma} u^\gamma \nabla_{\Gamma^\gamma} w^\gamma$, and the restricted B_{K^γ} obtained by replacing Γ^γ by K^γ in the previous definition.

Lemma 5.5.1 (Quasi Galerkin Orthogonality of the Error)**.** *Let u and ${u_h}^\gamma$ be the solutions to* (5.18) *and* (5.19) *respectively, then*

$$B_{\Gamma^\gamma}(u^\gamma - {u_h}^\gamma, {w_h}^\gamma) = \int_{\Gamma^\gamma} (f^\gamma \delta^\gamma - {f_h}^\gamma){w_h}^\gamma + \int_{\Gamma^\gamma} \nabla_{K^\gamma} {u^\gamma}^\mathsf{T} \mathbf{A}^\gamma \nabla_{K^\gamma} {w_h}^\gamma, \tag{5.20}$$

for all ${w_h}^\gamma$ in $\mathbb{S}_h^{\gamma,k}(\Gamma^\gamma)$, where $A^\gamma = \left(\frac{1}{q^\gamma}\mathbf{T}^\gamma(q^\gamma \mathbf{G}^{\gamma -1} - q\mathbf{G}^{-1})\mathbf{T}^{\gamma\mathsf{T}}\right) \circ \boldsymbol{F}^{-\gamma}$.

Proof. By linearity

$$B_{K^\gamma}(u^\gamma - {u_h}^\gamma, {w_h}^\gamma) = \int_{K^\gamma} \nabla_{K^\gamma} u^\gamma \nabla_{K^\gamma} w^\gamma - \int_{K^\gamma} f_h^\gamma {w_h}^\gamma. \tag{5.21}$$

Also observe that

$$\begin{aligned}\int_{K^\gamma} f^\gamma {w_h}^\gamma \frac{q}{q^\gamma} = \int_K f w_h &= \int_K \nabla_K u^\mathsf{T} \nabla_K w_h \\ &= \int_{K^\gamma} \nabla_{K^\gamma} {u^\gamma}^\mathsf{T} \left(\mathbf{T}^\gamma \mathbf{G}^{-1} \mathbf{T}^{\gamma\mathsf{T}} \frac{q}{q^\gamma}\right) \circ \boldsymbol{F}^{-\gamma} \nabla_{K^\gamma} {w_h}^\gamma.\end{aligned}$$

Then adding and subtracting the first and last terms to equation (5.21) we get

$$\begin{aligned}B_{K^\gamma}(u^\gamma - {u_h}^\gamma, {w_h}^\gamma) = &\int_{K^\gamma} f^\gamma {w_h}^\gamma \delta^\gamma - {f_h}^\gamma {w_h}^\gamma + \int_{K^\gamma} \nabla_\Gamma u^\gamma \nabla_\Gamma w^\gamma \\ &- \int_{K^\gamma} \nabla_{K^\gamma} {u^\gamma}^\mathsf{T} \left(\mathbf{T}^\gamma \mathbf{G}^{-1} \mathbf{T}^{\gamma\mathsf{T}} \frac{q}{q^\gamma}\right) \circ \boldsymbol{F}^{-\gamma} \nabla_{K^\gamma} {w_h}^\gamma,\end{aligned}$$

whence

$$\begin{aligned}B_{K^\gamma}(u^\gamma - u_h^\gamma, {w_h}^\gamma) = &\int_{K^\gamma} (f^\gamma \delta^\gamma - f_h)^\gamma {w_h}^\gamma \\ &+ \int_{K^\gamma} \nabla_{K^\gamma} {u^\gamma}^\mathsf{T} \left(\frac{1}{q^\gamma}\mathbf{T}^\gamma(q^\gamma \mathbf{G}^{-\gamma} - q\mathbf{G}^{-1})\mathbf{T}^{\gamma\mathsf{T}}\right) \circ \boldsymbol{F}^{-\gamma} \nabla_{K^\gamma} {w_h}^\gamma,\end{aligned}$$

because $\mathbf{T}^\gamma \mathbf{G}^{-\gamma} \mathbf{T}^{\gamma\mathsf{T}} = \mathbf{D}^{-\gamma} \mathbf{T}^{\gamma\mathsf{T}} = \mathbf{I} - \boldsymbol{\nu}^\gamma \otimes \boldsymbol{\nu}^\gamma$ according to (3.1). This is the desired result. □

Theorem 5.5.2 (Strang's Lemma)**.** *Let $\tilde{\Gamma}$ be a polyhedral approximation to Γ, u and ${u_h}^\gamma$ be the solutions to* (5.18) *and* (5.19) *respectively. If ${e_h}^\gamma = u^\gamma - {u_h}^\gamma$, then*

$$\begin{aligned}\frac{1}{4}|e_h^\gamma|^2_{H^1(\Gamma^\gamma)} \leq |u^\gamma|^2_{H^1(\Gamma^\gamma)} \|\mathbf{A}^\gamma\|^2_{L^\infty(\Gamma^\gamma)} + C|(f^\gamma \delta^\gamma - {f_h}^\gamma)|^2_{L^2(\Gamma^\gamma)} \\ + \inf_{{w_h}^\gamma \in \mathbb{S}_h^{\gamma,k}(\Gamma^\gamma)} \Big\{ |u^\gamma - {w_h}^\gamma|^2_{H^1(\Gamma^\gamma)} + |u^\gamma|_{H^1(\Gamma^\gamma)} \|\mathbf{A}^\gamma\|_{L^\infty(\Gamma^\gamma)} |u^\gamma - {w_h}^\gamma|_{H^1(\Gamma^\gamma)} \\ + |(f^\gamma \delta^\gamma - {f_h}^\gamma)|_{L^2(\Gamma^\gamma)} |u^\gamma - {w_h}^\gamma|_{L^2(\Gamma^\gamma)} \Big\}. \quad (5.22)\end{aligned}$$

Proof. Rewriting ${e_h}^\gamma = u^\gamma - {w_h}^\gamma + {w_h}^\gamma - {u_h}^\gamma$ for all ${w_h}^\gamma \in \mathbb{S}_h^{\gamma,k}(\Gamma^\gamma)$ we readily obtain

$$B_{\Gamma^\gamma}({e_h}^\gamma, {e_h}^\gamma) = \underbrace{B_{\Gamma^\gamma}({e_h}^\gamma, u^\gamma - {w_h}^\gamma)}_{\boxed{1}} + \underbrace{B_{\Gamma^\gamma}({e_h}^\gamma, {w_h}^\gamma - {u_h}^\gamma)}_{\boxed{2}}. \quad (5.23)$$

Cauchy-Schwarz inequality yields for $\boxed{1}$

$$|\boxed{1}| \leq |{e_h}^\gamma|_{H^1(\Gamma^\gamma)} |u^\gamma - {w_h}^\gamma|_{H^1(\Gamma^\gamma)} \leq \frac{1}{4}|{e_h}^\gamma|^2_{H^1(\Gamma^\gamma)} + |u^\gamma - {w_h}^\gamma|^2_{H^1(\Gamma^\gamma)}.$$

Invoking Lemma 5.5.1 we obtain for $\boxed{2}$

$$|\boxed{2}| \leq \underbrace{\left| \int_{\Gamma^\gamma} (f^\gamma \delta^\gamma - {f_h}^\gamma)({w_h}^\gamma - {u_h}^\gamma) \right|}_{\boxed{3}} + \underbrace{\left| \int_{\Gamma^\gamma} \nabla_{\Gamma^\gamma} {u^\gamma}^\mathsf{T} \mathbf{A}^\gamma \nabla_{\Gamma^\gamma} ({w_h}^\gamma - {u_h}^\gamma) \right|}_{\boxed{4}}.$$

Applying Cauchy-Schwarz inequality and adding and subtracting u^γ leads to

$$\boxed{3} \leq |(f^\gamma \delta^\gamma - {f_h}^\gamma)|_{L^2(\Gamma^\gamma)} \left(|u^\gamma - {w_h}^\gamma|_{L^2(\Gamma^\gamma)} + |{e_h}^\gamma|_{L^2(\Gamma^\gamma)} \right). \quad (5.24)$$

Employing Poincare's inequality $|{e_h}^\gamma|_{L^2(\Gamma^\gamma)} \leq C|{e_h}^\gamma|_{H^1(\Gamma^\gamma)}$ implies

$$\boxed{3} \leq |(f^\gamma \delta^\gamma - {f_h}^\gamma)|_{L^2(\Gamma^\gamma)} |u^\gamma - {w_h}^\gamma|_{L^2(\Gamma^\gamma)} + \frac{1}{4}|{e_h}^\gamma|^2_{H^1(\Gamma^\gamma)} + C|(f^\gamma \delta^\gamma - {f_h}^\gamma)|^2_{L^2(\Gamma^\gamma)}. \quad (5.25)$$

Similarly for $\boxed{4}$

$$\begin{aligned}
\boxed{4} &\leq \|\mathbf{A}^\gamma\|_{L^\infty(\Gamma^\gamma)}|u^\gamma|_{H^1(\Gamma^\gamma)}|{w_h}^\gamma - {u_h}^\gamma|_{H^1(\Gamma^\gamma)} \\
&\leq |u^\gamma|_{H^1(\Gamma^\gamma)}\|\mathbf{A}^\gamma\|_{L^\infty(\Gamma^\gamma)}|u^\gamma - {w_h}^\gamma|_{H^1(\Gamma^\gamma)} + |u^\gamma|_{H^1(\Gamma^\gamma)}\|\mathbf{A}^\gamma\|_{L^\infty(\Gamma^\gamma)}|e_h|_{H^1(\Gamma^\gamma)}) \\
&\leq |u^\gamma|_{H^1(\Gamma^\gamma)}\|\mathbf{A}^\gamma\|_{L^\infty(\Gamma^\gamma)}|u^\gamma - {w_h}^\gamma|_{H^1(\Gamma^\gamma)} + \frac{1}{4}|{e_h}^\gamma|^2_{H^1(\Gamma^\gamma)} + |u^\gamma|^2_{H^1(\Gamma^\gamma)}\|\mathbf{A}^\gamma\|^2_{L^\infty(\Gamma^\gamma)}.
\end{aligned} \tag{5.26}$$

Finally, gathering the above estimates an recalling the coercitivity $|{e_h}^\gamma|^2_{H^1(\Gamma^\gamma)} = B_{\Gamma^\gamma}({e_h}^\gamma, {e_h}^\gamma)$ we get

$$\begin{aligned}
\frac{1}{4}|{e_h}^\gamma|^2_{H^1(\Gamma^\gamma)} \leq & |u^\gamma - w_h^\gamma|^2_{H^1(\Gamma^\gamma)} + C|(f^\gamma\delta^\gamma - {f_h}^\gamma)|^2_{L^2(\Gamma^\gamma)} \\
& + |u^\gamma|_{H^1(\Gamma^\gamma)}\|\mathbf{A}^\gamma\|_{L^\infty(\Gamma^\gamma)}|u^\gamma - {w_h}^\gamma|_{H^1(\Gamma^\gamma)} + |u^\gamma|^2_{H^1(\Gamma^\gamma)}\|\mathbf{A}^\gamma\|^2_{L^\infty(\Gamma^\gamma)} \\
& + |(f^\gamma\delta^\gamma - {f_h}^\gamma)|_{L^2(\Gamma^\gamma)}|u^\gamma - {w_h}^\gamma|_{L^2(\Gamma^\gamma)}.
\end{aligned}$$

The desired result follows by taking the infimum over all possible ${w_h}^\gamma \in \mathbb{S}_h^{\gamma,k}(\Gamma^\gamma)$. □

5.6 Quadratic Isoparametric Elements

Some of the computational methods of Chapter 7 use a hierarchical approach to quadratics by working with its affine base. In this section we present a result that gives a geometric condition ensuring that the hybrid approach will work. The result is based on an idea by Ciarlet and Raviart [Cia78, Theorem 4.3.3] for flat elements. Here we extend it to surfaces.

Let $\tilde{K}$ be a d-simplex in $\mathbb{R}^{d+1}$ then definitions 5.1 make sense. Let $\hat{K} \subset \mathbb{R}^d$ be the reference element spanned by the canonical basis $\{\boldsymbol{e}_1, \ldots, \boldsymbol{e}_d\}$ and $F_{\tilde{K}} : \hat{K} \to \tilde{K}$ be the unique injective affine map such that the vertices of $\hat{K}$ are mapped to the vertices of $\tilde{K}$. Then $h_{\tilde{K}}/\sigma_{\tilde{K}} \approx \sqrt{(DF_{\tilde{K}})^\mathsf{T}(DF_{\tilde{K}})} = \tilde{q}$. Here we are interested in

surface quadratic isoparametric elements. For simplicity of notation we let $K = K^2$ and $F : \hat{K} \to K$. We denote the quadratic midnodes by x_{ij} and $\tilde{x}_{ij}$, see Figure 5.3. By definition of isoparametric element we have

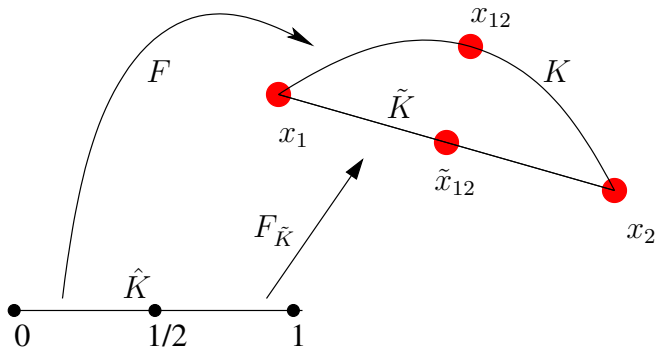


Figure 5.3: The picture shows the elements involve in the approximation result for quadratic isoparametric element: K is the quadratic elements, $\tilde{K}$ its affine base and $\hat{K}$ the master element.

$$F(\hat{\boldsymbol{x}}) = \mathbf{B}\hat{\boldsymbol{x}} + \boldsymbol{b} + \Sigma \hat{p}_{ij}(\hat{\boldsymbol{x}})(x_{ij} - \tilde{x}_{ij}),$$

where $\hat{p}_{ij}$ is the reference nodal Lagrange basis function of degree 2.

Theorem 5.6.1 (Quadratic Approximation)**.** *Let $(\tilde{\Gamma}, \tilde{\mathcal{T}}, \boldsymbol{l})$ be a polyhedral approximation with shape regularity parameter σ such that*

$$|x_{ij} - \tilde{x}_{ij}| \leq O(h_{\tilde{K}}^2). \tag{5.27}$$

Then for small h_K there are constants C_0 and C_1 depending on σ such that

$$C_0 |\tilde{q}|_{\tilde{K}} \leq |q|_{\infty,K} \leq C_1 |\tilde{q}|_{\tilde{K}},$$

where $q = \sqrt{\det((DF)^{\mathsf{T}}(DF))}$ is the area element.

Proof. From the definition of F

$$DF(\hat{\boldsymbol{x}}) = \mathbf{B}(\hat{\boldsymbol{x}}) + \mathbf{E}(\hat{\boldsymbol{x}}),$$

where $\mathbf{E}(\hat{\boldsymbol{x}}) = \Sigma D\hat{p}_{ij}(\hat{\boldsymbol{x}})(x_{ij} - \tilde{x}_{ij})$. Since the basis function $\hat{p}_{ij}$ are independent of K and using Assumption (5.27), we find that

$$\sup_{\hat{K}} |\mathbf{E}(\hat{\boldsymbol{x}})| \leq Ch^2.$$

Since the first fundamental form $\mathbf{G} = (DF)^{\mathsf{T}}(DF) = \mathbf{B}^{\mathsf{T}}\mathbf{B} + \mathbf{B}^{\mathsf{T}}\mathbf{E} + \mathbf{E}^{\mathsf{T}}\mathbf{B} + \mathbf{E}^{\mathsf{T}}\mathbf{E}$, it follows that $\mathbf{G} = \mathbf{B}^{\mathsf{T}}\mathbf{B} + O(h^2)$ and $\det(\mathbf{G}) = \det(\mathbf{B}^{\mathsf{T}}\mathbf{B}) + O(h^4)$. This implies $q^2 = \tilde{q}^2 + O(h^4)$ and ,since $\tilde{q} \approx h_K/\sigma_K$, the result follows for h_K small. □

5.7 FEM for Geometric Evolution Equations

To define an approximate solution of a geometric evolution equation (Section 4.2) $\boldsymbol{v} = -\delta E$ a Petrov-Galerkin finite element method for the space variable is solved at each time. Here $E(\boldsymbol{x})$ stands for some energy, δE for its functional derivative and $\boldsymbol{v}$ for the velocity prescribing the surface Γ. In order to obtain a practical method we still have to discretize in time. Consider the time partition $t_0 = 0 < \cdots < t_{M+1} = T$ of $[0, T]$, with time step $\tau_n := t_{n+1} - t_n$, $n = 0, \ldots, M$. Let $\mathbb{S}_h := \mathbb{S}_h^{\gamma,k}$ and $\boldsymbol{x}_h^{n+1} \in \mathbb{S}_h(\Gamma_h^n)$ be a parametric approximation to $\mathbf{Id}_{\Gamma(t_{n+1})}$. Let Γ_h^0 be a polyhedral approximation to $\Gamma(0)$ of degree γ. Then a sequence $\{\boldsymbol{x}_h^n\}$ of approximations to $\{\mathbf{Id}_{\Gamma(t_n)}\}$ is defined recursively by

$$\boldsymbol{x}_h^{n+1} \in \mathbb{S}_h(\Gamma_h^n): \quad \int_{\Gamma_h^n} \frac{(\boldsymbol{x}_h^{n+1} - \mathbf{Id}_{\Gamma_h^n})}{\tau_n} \cdot \boldsymbol{\phi}_h = -\int_{\Gamma_h^n} \delta E \boldsymbol{\phi}_h \quad \forall \boldsymbol{\phi}_h \in \mathbb{S}_h(\Gamma_h^n). \quad (5.28)$$

Then Γ_h^{n+1} is obtained from $\boldsymbol{x}_h^{n+1}$, preserving the connectivity of Γ_h^n. The specific schemes obtained by applying this method to the different flows (i.e. different energies E) of Chapter 4 will be described in detail in Chapter 6.

5.8 Gradient Recovery

Recall the finite element method for the Poisson equation on a domain $\Omega \subset \mathbb{R}^2$

$$a(u,v) := \int_\Omega \nabla u \nabla v = \int_\Omega fv \quad \forall v \in \mathbb{V}.$$

Let u_h be the finite element solution in $\mathbb{X}_h^1(\Omega)$, i.e. u_h is a piecewise linear function. Up to a log and a constant the following error estimates are satisfied

$$\begin{aligned} \|u - u_h\|_{L^\infty} &\lesssim h^2 \|D^2 u\|_{L^\infty}, \\ \|\nabla u - \nabla u_h\|_{L^\infty} &\lesssim h \|D^2 u\|_{L^\infty}. \end{aligned} \tag{5.29}$$

Sometimes one wishes to build and approximation to $D^2 u$ from u_h. This is the goal of the gradient recovery technique (see for example [HSWW01]). The idea is to use some sort of average of ∇u_h to gain information on the second derivatives. Below we discuss the idea of the method applied to the Poisson's problem. From the discussion we obtain the size of the patch in terms of h. Let $r := \partial_i u$ and $r_h := \partial_i u_h$ with $i = 1, 2$. Let

$$\nabla_h r_h = G r_h := \int \nabla \delta(z - x) r_h(x) dx,$$

where for a given number $H > 0$, δ is a function such that:

1. $\text{supp}(\delta) \subset B(z, H)$,
2. $\int_B \nabla \delta \lesssim 1/H$,
3. $\int_{B(z,H)} \delta = 1$.

Then for $B = B(z, H) \subset \Omega$ we have

$$\begin{aligned} Gr_h(z) - \nabla r(z) &= \int_B \nabla\delta(z-x) r_h(x) dx - \int_B \delta(z-x)\nabla r(z) dx \\ &= -\int_B \nabla\delta(z-x) r(x) dx + \int_B \nabla\delta(z-x)(r_h(x) - r(x)) dx \\ &\quad - \int_B \delta(z-x)\nabla r(z) dx \\ &= \int_B \delta(z-x)\nabla r(x) dx + \int_B \nabla\delta(z-x)(r_h(x) - r(x)) dx \\ &\quad - \int_B \delta(z-x)\nabla r(z) dx \\ &= \int_B \nabla\delta(z-x)(r_h(x) - r(x)) dx + \int_B \delta(z-x)(\nabla r(z) - \nabla r(x)) dx. \end{aligned}$$

From here and the finite element error estimates (5.29) we get

$$|Gr_h(z) - \nabla r(z)| \lesssim \frac{h}{H}\|D^2 u\|_{\infty,B} + H\|D^2 r\|_{\infty,B} = \frac{h}{H}C_1 + HC_2,$$

provided $u \in W^3_\infty$. The inequality is optimal when $H = \sqrt{C_1 h / C_2}$, and gives

$$|Gr_h(z) - \nabla r(z)| \le C\sqrt{h}.$$

This calculation extends to the Laplace-Beltrami operator on close and smooth surfaces.

Chapter 6

Numerical Schemes

In this chapter we present space and time discretizations of the continuous problems described in chapter 4. A Petrov-Galerkin finite element method for the space variable is solved at each time. First we treat the geometric case. Doing this provides helpful insight as to how to deal with the fluid-membrane schemes treated later in this chapter.

6.1 Geometric Evolution Equations Schemes

First we describe a discrete weak formulation and then the matrix formulation of the problems. This is because of notational convenience as many matrices and equations are shared by the different problems. The discrete schemes in this section are obtain by applying the general method proposed in section 5.7 to the continuous problems of section 4.2. Continuous piecewise polynomial finite element spaces will be used for the space discretization of all the variables. The use of a mixed finite element method for the fourth order problems allows us to compute the curvature vector even when the representation of the surface is only continuous [Rus05, DD07, DDE05]. In particular, finite elements globally C^1 are not needed in this context. Other methods to avoid global C^1 elements for surfaces are described in [COS00] and the references therein.

Recalling the notation of section 5.7 where $\boldsymbol{x}_h^{n+1} \in \mathbb{S}_h(\Gamma_h^n)$ is a parametric approximation to $\mathbf{Id}_{\Gamma(t_{n+1})}$, we can define $\boldsymbol{v}_h^{n+1} \in \mathbb{S}_h(\Gamma_h^n)$ an approximation to the velocity through the equation

$$\boldsymbol{x}_h^{n+1} = \mathbf{Id}_{\Gamma_h^n} + \tau_n \boldsymbol{v}_h^{n+1}. \tag{6.1}$$

The following remarks explain general choices taken in the discretizations. In the corresponding Subsections the specific choices are described.

Remark 6.1.1 (Velocity and Position). At the discrete level it is equivalent to work either with the velocity or the position, it's just a change of variable. Working with the velocity helps make the exposition more coherent as this is the natural variable when the membrane is coupled with the fluid. So we will work all our discrete schemes using $\boldsymbol{v}_h^{n+1}$ instead of $\boldsymbol{x}_h^{n+1}$.

Remark 6.1.2 (Geometric Built-in Linearization). The surface gradients as well as all the integrals are computed in the previous domain. This linearizes the "geometrical" nonlinearity coming from the free surface.

Remark 6.1.3 (Time Discretization). As in Section 4.2 let $\boldsymbol{u}(\cdot, t) = \mathbf{Id}_{\Gamma(t)}$. For the time discretization we use an implicit Euler scheme obtained by the velocity approximation $\dot{\boldsymbol{u}}(t^{n+1}) \approx \boldsymbol{v}_h^{n+1}$ and using equation (6.1) the position approximation $\mathbf{Id}_{\Gamma(t_{n+1})} \approx \boldsymbol{x}_h^{n+1} = \mathbf{Id}_{\Gamma_h^n} + \tau_n \boldsymbol{v}_h^{n+1}$.

6.1.1 Discrete Weak Formulations

We assume that the initial surface Γ_0 is known and Γ_h^0 is a polyhedral approximation to Γ_0 (Definition 5.2.2). To avoid notational complications Γ_h^0 could also denote a

piecewise polynomial surface without explicit use of the superscript γ. Also we use $\mathbb{S}_h(\Gamma_h^n)$ to denote the finite element space $\mathbb{S}_h^{\gamma,k}$ with $k = \gamma$.

6.1.1.1 Mean Curvature Flow

A discrete scheme to approximate Problem 4.2.3 follows. Given the initial polyhedral approximation Γ_h^0 and a time partition $t_0 = 0 < \cdots < t_{M+1} = T$ of $[0, T]$, with time-step $\tau_n := t_{n+1} - t_n$, $n = 0, \ldots, M$, find the velocity $\boldsymbol{v}_h^{n+1} \in \mathbb{S}_h(\Gamma_h^n)$ such that

$$\int_{\Gamma_h^n} \boldsymbol{v}_h^{n+1} \cdot \boldsymbol{\phi}_h + \tau \int_{\Gamma_h^n} \nabla_{\Gamma_h^n} \boldsymbol{v}_h^{n+1} : \nabla_{\Gamma_h^n} \boldsymbol{\phi}_h = -\int_{\Gamma_h^n} \nabla_{\Gamma_h^n} \mathbf{Id}_{\Gamma_h^n} : \nabla_{\Gamma_h^n} \boldsymbol{\phi}_h \qquad \forall \boldsymbol{\phi}_h \in \mathbb{S}_h(\Gamma_h^n), \tag{6.2}$$

is satisfied. And Γ_h^{n+1} is obtained from $\boldsymbol{v}_h^{n+1}$ through equation (6.1) keeping the connectivity of Γ_h^n. This scheme is due to Dziuk [Dzi91].

Similarly, for the constraint Problem 4.2.4 we have: find the velocity $\boldsymbol{v}_h^{n+1} \in \mathbb{S}_h(\Gamma_h^n)$ and the scalars π^{n+1} such that

$$\begin{aligned} \int_{\Gamma_h^n} \boldsymbol{v}_h^{n+1} \cdot \boldsymbol{\phi}_h + \tau \int_{\Gamma_h^n} \nabla_{\Gamma_h^n} \boldsymbol{v}_h^{n+1} : \nabla_{\Gamma_h^n} \boldsymbol{\phi}_h & \\ = -\int_{\Gamma_h^n} \nabla_{\Gamma_h^n} \mathbf{Id}_{\Gamma_h^n} : \nabla_{\Gamma_h^n} \boldsymbol{\phi}_h + \pi^{n+1} \int_{\Gamma_h^n} \boldsymbol{\phi}_h \cdot \boldsymbol{\nu}_h^n \qquad & \forall \boldsymbol{\phi}_h \in \mathbb{S}_h(\Gamma_h^n), \end{aligned} \tag{6.3}$$

$$\operatorname{meas}(\Omega_h^{n+1}) = \operatorname{meas}(\Omega_h^n). \tag{6.4}$$

And again Γ_h^{n+1} is obtained from $\boldsymbol{v}_h^{n+1}$ through equation (6.1) keeping the connectivity of Γ_h^n. The reason to take the term involving the multiplier explicit is due to the solving method described in Section 7.3.

6.1.1.2 Willmore Flow

A discrete scheme to approximate Problem 4.2.6 is the following. Given an initial approximation Γ_h^0, an initial approximation to curvature $\boldsymbol{h}_h^0 \in \mathbb{S}_h(\Gamma_h^0)$ and a time partition $t_0 = 0 < \cdots < t_{M+1} = T$ of $[0,T]$, with time-step $\tau_n := t_{n+1} - t_n$, $n = 0, \ldots, M$, find the velocity $\boldsymbol{v}_h^{n+1} \in \mathbb{S}_h(\Gamma_h^n)$ and the curvature $\boldsymbol{h}_h^{n+1} \in \mathbb{S}_h(\Gamma_h^n)$ such that

$$\int_{\Gamma_h^n} \boldsymbol{v}_h^{n+1} \cdot \boldsymbol{\phi}_h = -\int_{\Gamma_h^n} \delta W_h^{n+1} \cdot \boldsymbol{\phi}_h, \qquad \forall \boldsymbol{\phi}_h \in \mathbb{S}_h(\Gamma_h^n), \tag{6.5}$$

and

$$\int_{\Gamma_h^n} \boldsymbol{h}_h^{n+1} \cdot \boldsymbol{\psi}_h - \tau \int_{\Gamma_h^n} \nabla_{\Gamma_h^n} \boldsymbol{v}_h^{n+1} : \nabla_{\Gamma_h^n} \boldsymbol{\psi}_h = \tau \int_{\Gamma_h^n} \nabla_{\Gamma_h^n} \mathbf{Id}_h^n : \nabla_{\Gamma_h^n} \boldsymbol{\psi}_h, \quad \forall \boldsymbol{\psi}_h \in \mathbb{S}_h(\Gamma_h^n), \tag{6.6}$$

are satisfied. Here $\delta W_h^{n+1} := \delta W_h^{n+1}(\Gamma_h^n, \boldsymbol{h}_h^n, \boldsymbol{v}_h^{n+1}, \boldsymbol{h}_h^{n+1})$ is an approximation of δW, equation (6.5) is an approximation of (4.26) and equation (6.6) is an approximation of the mean curvature equation consistent with (6.2). Recall that δW is given by equation (4.19), and satisfies (from equation (4.23))

$$\int_\Gamma \delta W \cdot \boldsymbol{\phi} = -\int_\Gamma \left[(\mathbf{I} - 2\boldsymbol{\nu} \otimes \boldsymbol{\nu}) \nabla_\Gamma \boldsymbol{h} \right] : \nabla_\Gamma \boldsymbol{\phi} + \frac{1}{2} \int_\Gamma |\boldsymbol{h}|^2 \nabla_\Gamma \mathbf{Id} : \nabla_\Gamma \boldsymbol{\phi} \qquad \forall \boldsymbol{\phi}. \tag{6.7}$$

The following choices have been made concerning the time integration of the above relation:

- The surface gradients, the outer unit normal as well as all the integral are computed using the previous domain in order to remove the "geometrical" nonlinearity coming from the free surface (cf. Remark 6.1.2);

- the curvature in the nonlinear term

$$\int_\Gamma |\boldsymbol{h}|^2 \nabla_\Gamma \mathbf{Id} : \nabla_\Gamma \boldsymbol{\phi}$$

  is treated explicitly, which removes the "algebraic" nonlinearity;

- the term $-\int_\Gamma \left[(\mathbf{I} - 2\boldsymbol{\nu}\otimes\boldsymbol{\nu})\,\nabla_\Gamma \boldsymbol{h}\right] : \nabla_\Gamma \boldsymbol{\phi}$ leads to a system with positive and negative eigenvalues difficult to solve numerically (see Figure 6.1). Therefore, bearing in mind the use of an iterative method to solve positive definite systems, following [CDD+04], the above term is split into two parts

$$\int_\Gamma \nabla_\Gamma \boldsymbol{h} \cdot \nabla_\Gamma \boldsymbol{\phi} - 2\int_\Gamma (\mathbf{I} - \boldsymbol{\nu}\otimes\boldsymbol{\nu}) \nabla_\Gamma \boldsymbol{h} \cdot \nabla_\Gamma \boldsymbol{\phi}$$

  and the former is treated implicitly whilst the latter needs to be treated explicitly. Note that the above splitting does not solve the issue of zero eigenvalues.

Under these considerations the chosen approximation of δW reads

$$\begin{aligned}
\int_{\Gamma_h} \delta W_h^n \cdot \boldsymbol{\phi}_h = &- \int_{\Gamma_h^n} \nabla_{\Gamma_h^n} \boldsymbol{h}_h^{n+1} : \nabla_{\Gamma_h^n} \boldsymbol{\phi}_h \\
&- \frac{\tau}{2} \int_{\Gamma_h^n} |\tilde{\boldsymbol{h}}_h^n|^2 \nabla_{\Gamma_h^n} \boldsymbol{v}_h^{n+1} : \nabla_{\Gamma_h^n} \boldsymbol{\phi}_h \\
&+ 2\int_{\Gamma_h^n} \left[(\mathbf{I} - \boldsymbol{\nu}\otimes\boldsymbol{\nu}) \nabla_{\Gamma_h^n} \bar{\boldsymbol{h}}_h^n\right] : \nabla_{\Gamma_h^n} \boldsymbol{\phi}_h \\
&+ \frac{1}{2} \int_{\Gamma_h^n} |\tilde{\boldsymbol{h}}_h^n|^2 \nabla_{\Gamma_h^n} \mathbf{Id}_h^n : \nabla_{\Gamma_h^n} \boldsymbol{\phi}_h \qquad \forall \boldsymbol{\phi}_h.
\end{aligned} \tag{6.8}$$

Remark 6.1.4 (Explicit Treatment of $\boldsymbol{h}$ and $\boldsymbol{\nu}$). In (6.8) there are several possibilities for what is meant by $\bar{\boldsymbol{h}}_h^n$, $\tilde{\boldsymbol{h}}_h^n$ and $\boldsymbol{\nu}$. In [CDD+04], they use the element normal for $\boldsymbol{\nu}$ and the transported $\boldsymbol{h}_h^n$ from Γ_h^{n-1} to Γ_h^n for $\tilde{\boldsymbol{h}}_h^n$. This is because they use piecewise linear elements and then the element curvature is 0. Instead for $\bar{\boldsymbol{h}}_h^n$ they decide to

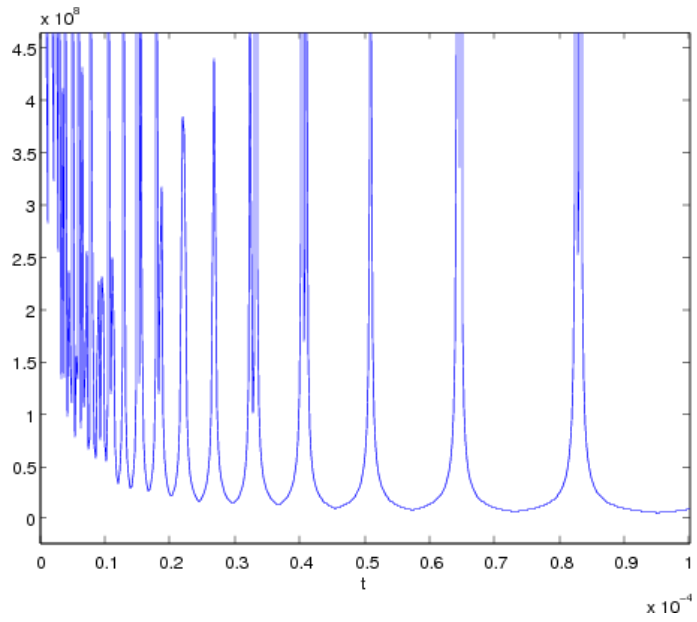


Figure 6.1: Condition number of the matrix $[M + \tau A_h, -A_{\boldsymbol{\nu}}; -\tau A, M]$ in terms of the time-step. This shows that for some τ the system has zero eigenvalues. The matrix corresponds to a 2x1 aspect ratio ellipse in $\mathbb{R}^2$.

use an explicit computation of curvature on Γ_h^n. When using quadratic isoparametric elements other possible choice would be to use the element curvature for $\tilde{\boldsymbol{h}}_h^n$. We use the approach of [CDD+04] for both our piecewise linear and piecewise quadratic computations. We plan to investigate how the other choices compare in the near future.

Remark 6.1.5 (Initial Curvature). In practice the initial mean curvature may be difficult to provide. And the Willmore flow highly depends on having a good approximation to curvature. Figures 6.2 and 6.3 show that using piecewise linear and a explicit computation of curvature can be very stiff for the flow. Observe in Figure 6.3 that it takes about 5 iterations for the linear flow to get into a reasonable regime. This is due to the bad initial approximation to curvature that can be seen from Figure 6.2. This one is a rather simple and nice initial mesh so the flow gets

started, but there are other instances that the flow can never get started. One possible solution is to use higher order interpolation to compute the initial curvature. Another one is to apply a gradient recovery method to the explicit computation of curvature.

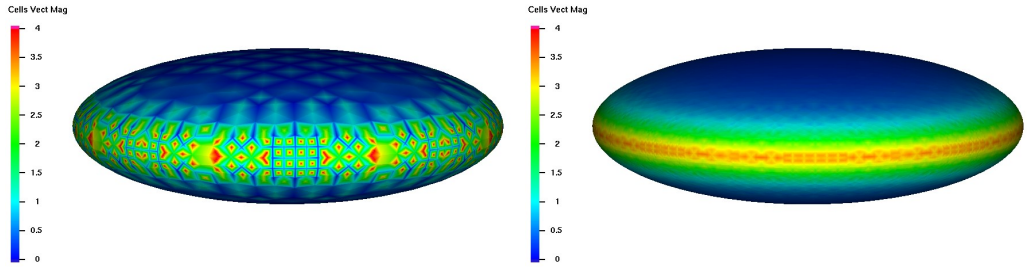


Figure 6.2: The figure shows the initial curvature obtained for the Willmore flow using piecewise linear and piecewise quadratic elements. As can be seen from the picture using piecewise linears the approximation of curvatrure is rather bad. This is reflected in the energy graph of Figure 6.3

Remark 6.1.6 (Gradient Recovery). For small h, the surface approximation looks locally as a flat domain. So it makes sense to apply the gradient recovery method of Section 5.8, using $\boldsymbol{h}$ in place of u.

6.1.1.3 Biomembranes

Recalling section 4.2.3, the geometric model for biomembranes is given by the Willmore flow when it is subject to surface area and enclosed volume constraints. Then using equation (6.8), and considering the area and volume multipliers, the chosen

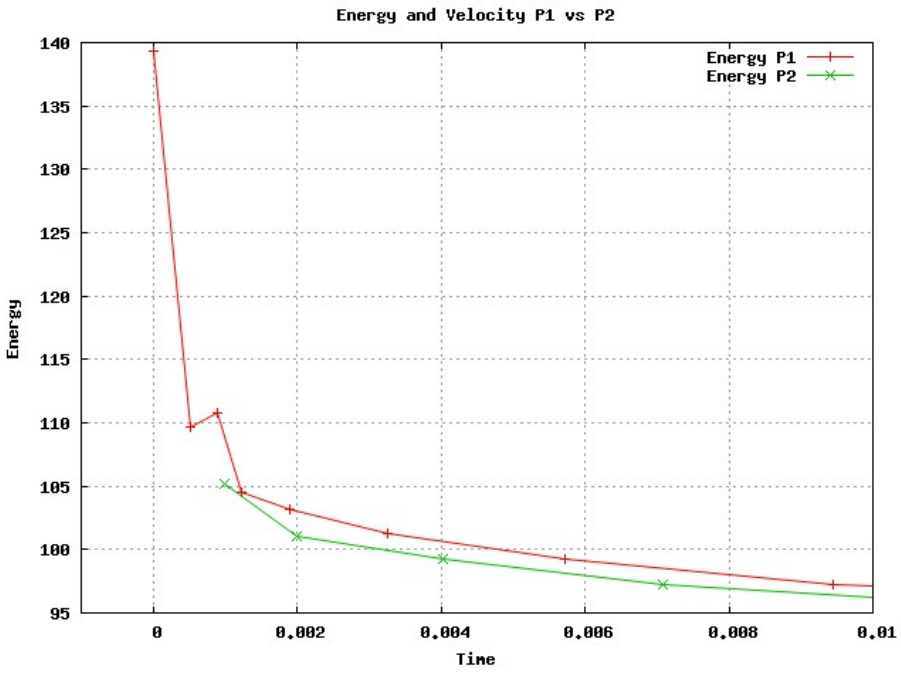


Figure 6.3: Energy graph for the first few iterations of Willmore flow. The initial shape is the 3x3x1 ellipsoid of Figure 6.2. The graph compares the behavior using piecewise linears and piecewise quadratics. Observe that it takes about 5 iterations for the linear flow to get into a reasonable regime. This is due to the bad initial approximation to curvature. This one is a rather simple and nice initial mesh so the flow gets started, but there are other instances that the flow can never get started.

approximation of δE reads

$$\begin{aligned}\int_{\Gamma_h} \delta E_h^n \cdot \boldsymbol{\phi}_h = &- \int_{\Gamma_h^n} \nabla_{\Gamma_h^n} \boldsymbol{h}_h^{n+1} : \nabla_{\Gamma_h^n} \boldsymbol{\phi}_h \\ &- \frac{\tau}{2} \int_{\Gamma_h^n} |\tilde{\boldsymbol{h}}_h^n|^2 \nabla_{\Gamma_h^n} \boldsymbol{v}_h^{n+1} : \nabla_{\Gamma_h^n} \boldsymbol{\phi}_h \\ &+ 2 \int_{\Gamma_h^n} \left[(\mathbf{I} - \boldsymbol{\nu} \otimes \boldsymbol{\nu}) \nabla_{\Gamma_h^n} \bar{\boldsymbol{h}}_h^n \right] : \nabla_{\Gamma_h^n} \boldsymbol{\phi}_h \\ &+ \frac{1}{2} \int_{\Gamma_h^n} |\tilde{\boldsymbol{h}}_h^n|^2 \nabla_{\Gamma_h^n} \mathbf{Id}_h^n : \nabla_{\Gamma_h^n} \boldsymbol{\phi}_h \\ &+ \lambda^{n+1} \int_{\Gamma_h^n} \tilde{\boldsymbol{h}}_h^n \cdot \boldsymbol{\phi}_h + \pi^{n+1} \int_{\Gamma_h^n} \boldsymbol{\phi}_h \cdot \boldsymbol{\nu} \qquad \forall \boldsymbol{\phi}_h. \end{aligned} \tag{6.9}$$

So that the fully discrete scheme for the geometric biomembrane model is as follows:

given an initial approximation Γ_h^0, an initial approximation to curvature $\boldsymbol{h}_h^0 \in \mathbb{S}_h(\Gamma_h^0)$

and a time partition $t_0 = 0 < \cdots < t_{M+1} = T$ of $[0,T]$, with time-step $\tau_n := t_{n+1} - t_n$, $n = 0, \ldots, M$, find the velocity $\boldsymbol{v}_h^{n+1} \in \mathbb{S}_h(\Gamma_h^n)$, the curvature $\boldsymbol{h}_h^{n+1} \in \mathbb{S}_h(\Gamma_h^n)$ and the scalars π^{n+1} and λ^{n+1}, such that equation (6.5) with δW_h^n given by (6.9), equation (6.6) and

$$\text{meas}(\Omega_h^{n+1}) = \text{meas}(\Omega_h^n), \qquad \text{meas}(\Gamma_h^{n+1}) = \text{meas}(\Gamma_h^n), \tag{6.10}$$

are satisfied. The actual computation of the Lagrange multipliers λ^{n+1} and π^{n+1} will be explained in Section 7.3.

6.1.2 Matrix Formulation

In this section we turn our attention to equivalent matrix formulations of the fully discrete problems of the previous Subsection. We comment about the solution method for the resulting linear systems. Also definitions from this section will be used in section 6.2.2.

Let $\{\phi_i\}_{i=1}^N$ be the finite element basis of $\mathbb{S}_h$. Let $\{\boldsymbol{e}_k\}_{k=1}^{d+1}$ be the canonical basis of $\mathbb{R}^{d+1}$. Then by definition $\boldsymbol{\phi}_{i,k} = \phi_i \boldsymbol{e}_k$ is the finite element basis of $(\mathbb{S}_h)^{d+1}$. Let $(\mathbf{M})_{ij}$ indicate the (i,j)-th component of matrix $\mathbf{M}$, and let $\vec{\mathbf{M}}$ be a matrix whose components $(\vec{\mathbf{M}})_{ij}$ are $d+1$ square matrices. We then define

- $(\mathbf{M})_{ij} = \int_{\Gamma_h} \phi_i \phi_j, \qquad (\vec{\mathbf{M}})_{ij} = (\mathbf{M})_{ij} \mathbf{I}_{d+1},$
- $(\mathbf{A})_{ij} = \int_{\Gamma_h} \nabla_{\Gamma_h} \phi_i \cdot \nabla_{\Gamma_h} \phi_j, \qquad (\vec{\mathbf{A}})_{ij} = (\mathbf{A})_{ij} \mathbf{I}_{d+1},$
- $(\mathbf{A_h})_{ij} = \int_{\Gamma_h} |\boldsymbol{h}|^2 \nabla_{\Gamma_h} \phi_i \cdot \nabla_{\Gamma_h} \phi_j, \qquad (\vec{\mathbf{A_h}})_{ij} = (\mathbf{A_h})_{ij} \mathbf{I}_{d+1},$
- $(\vec{\mathbf{A}_{\boldsymbol{\nu}}})_{ij} = \int_{\Gamma_h} (\mathbf{I} - \boldsymbol{\nu}_h \otimes \boldsymbol{\nu}_h) \nabla_{\Gamma_h} \phi_i \cdot \nabla_{\Gamma_h} \phi_j.$

Given a function $\boldsymbol{v}_h \in (\mathbb{S}_h)^{d+1}$ it follows that $\boldsymbol{v}_h = \Sigma_{i,k} V_{i,k} \boldsymbol{\phi}_{i,k}$. We denote its nodal values by $\vec{\boldsymbol{V}}$. Similarly we use $\vec{\boldsymbol{X}}$ to denote the nodal values of $\mathbf{Id}_{\Gamma_h}$.

To go from a discrete weak formulation to an equivalent matrix system, we have to compute the entries that the weak formulation produces for a pair of basis functions. To illustrate the process assume the weak formulation has a bilinear form $\mathcal{B}(\boldsymbol{\phi}, \boldsymbol{\psi}) = \int_\Gamma \mathbf{B} \nabla \boldsymbol{\phi} : \nabla \boldsymbol{\psi}$, where $\mathbf{B}$ is a $d+1$ square matrix. The corresponding finite element matrix using our notation would be $\vec{\mathbf{B}}$. Let $\mathbf{D} := (\vec{\mathbf{B}})_{i,j}$, then using the product rules (3.27) we get

$$\begin{aligned}
(\mathbf{D})_{k,l} &= \int_\Gamma \mathbf{B} \nabla \boldsymbol{\phi}_{i,k} : \nabla \boldsymbol{\phi}_{j,l} \\
&= \int_\Gamma (\mathbf{B}\boldsymbol{e}_k \otimes \nabla \phi_i) : (\boldsymbol{e}_l \otimes \nabla \phi_j) = \int_\Gamma ((\mathbf{B}\boldsymbol{e}_k) \otimes \nabla \phi_i) : (\boldsymbol{e}_l \otimes \nabla \phi_j) \qquad (6.11) \\
&= \int_\Gamma (\mathbf{B}\boldsymbol{e}_k \cdot \boldsymbol{e}_l)(\nabla \phi_i \cdot \nabla \phi_j) = \int_\Gamma b_{kl} \nabla \phi_i \cdot \nabla \phi_j,
\end{aligned}$$

which implies that $\mathbf{B} = \mathbf{D}$. This gives a justification for all previously defined matrices and is the proof of the following equivalence lemmas.

6.1.2.1 Matrix System for Mean Curvature Flow

Lemma 6.1.1 (Mean Curvature Flow)**.** *Using the definitions at the beginning of Section 6.1.2, let* $\vec{\mathbf{C}}_n = \vec{\mathbf{M}}_n + \tau_n \vec{\mathbf{A}}_n$ *and* $\vec{\boldsymbol{F}}_n = -\vec{\mathbf{M}}_n \vec{\boldsymbol{X}}_n$*. Then equation* (6.2) *is equivalent to*

$$\vec{\mathbf{C}}_n \vec{\boldsymbol{V}}_{n+1} = \vec{\boldsymbol{F}}_n, \qquad (6.12)$$

and equation (6.3) *is equivalent to*

$$\vec{\mathbf{C}}_n \vec{\boldsymbol{V}}_{n+1} = \vec{\boldsymbol{F}}_n + \pi^{n+1} \vec{\boldsymbol{F}}_n^{\pi}, \qquad (6.13)$$

where $(\vec{\boldsymbol{F}}_n^{\pi})_{i,k} = \int_{\Gamma_h^n} \boldsymbol{\phi}_{i,k} \cdot \boldsymbol{\nu}$.

Remark 6.1.7. The matrix $\vec{\mathbf{C}}_n$ is symmetric and positive definite. So the preconditioned conjugate gradient method (PCG) can be use to solve equation (6.12).

6.1.2.2 Matrix System for Willmore and Bending Flow

Lemma 6.1.2 (Geometric Willmore)**.** *Using the definitions at the beginning of Section 6.1.2, equations* (6.5)*,* (6.6) *and* (6.7) *are equivalent to the matrix system*

$$\begin{aligned} \vec{\mathbf{M}}_n \vec{\boldsymbol{V}}_{n+1} + \vec{\mathbf{A}}_n \vec{\boldsymbol{H}}_{n+1} + \tau_n \vec{\mathbf{A}}_{\mathbf{h}n} \vec{\boldsymbol{V}}_{n+1} &= 2\vec{\mathbf{A}}_{\boldsymbol{\nu}n} \vec{\boldsymbol{H}}_n - \vec{\mathbf{A}}_{\mathbf{h}n} \vec{\boldsymbol{X}}_n, \\ \vec{\mathbf{M}}_n \vec{\boldsymbol{H}}_{n+1} - \tau_n \vec{\mathbf{A}}_n \vec{\boldsymbol{V}}_{n+1} &= \vec{\mathbf{A}}_n \vec{\boldsymbol{X}}_n. \end{aligned} \tag{6.14}$$

Here $\vec{\boldsymbol{H}}_n$ *is computed according to the decision taken in Remark 6.1.4.*

The next two remarks describe two different ways in which the previous system can be solved.

Remark 6.1.8 (Full System). Multiplying the second equation of (6.14) by $-1/\tau_n$ the system matrix can be written in a symmetric way as

$$\begin{bmatrix} \vec{\mathbf{M}}_n + \tau_n \vec{\mathbf{A}}_{\mathbf{h}n} & \vec{\mathbf{A}}_n \\ \vec{\mathbf{A}}_n & -\frac{1}{\tau_n} \vec{\mathbf{M}}_n \end{bmatrix}. \tag{6.15}$$

Since this matrix is not positive definite the CG method does not work. We can use instead either of the iterative solvers MINRES or GMRES.

Remark 6.1.9 (Schur Complement). As the mass matrix is invertible we can solve the second equation of (6.14) for $\vec{\boldsymbol{H}}_{n+1}$:

$$\vec{\boldsymbol{H}}_{n+1} = \vec{\mathbf{M}}_n^{-1} \vec{\mathbf{A}}_n (\vec{\boldsymbol{X}}_n + \tau_n \vec{\boldsymbol{V}}_n).$$

We next plug this into the first equation in (6.14) to get:

$$\vec{\mathbf{M}}_n \vec{\boldsymbol{V}}_{n+1} + \vec{\mathbf{A}}_n \vec{\mathbf{M}}_n^{-1} \vec{\mathbf{A}}_n \vec{\boldsymbol{X}}_n + \tau_n \vec{\mathbf{A}}_n \vec{\mathbf{M}}_n^{-1} \vec{\boldsymbol{V}}_n + \tau_n \vec{\mathbf{A}}_{\mathbf{h}n} \vec{\boldsymbol{V}}_{n+1} = 2\vec{\mathbf{A}}_{\boldsymbol{\nu}n} \vec{\boldsymbol{H}}_n - \vec{\mathbf{A}}_{\mathbf{h}n} \vec{\boldsymbol{X}}_n.$$

Collecting terms, and using that according to Remark 6.1.4 $\vec{\boldsymbol{H}}_n$ is computed explicitly by $\vec{\boldsymbol{H}}_n = \vec{\mathbf{M}}_n^{-1}\vec{\mathbf{A}}_n\vec{\boldsymbol{X}}_n$, we get

$$\left(\vec{\mathbf{M}}_n + \tau_n\left(\vec{\mathbf{A}_{\mathbf{h}}}_n + \vec{\mathbf{A}}_n\vec{\mathbf{M}}_n^{-1}\vec{\mathbf{A}}_n\right)\right)\vec{\boldsymbol{V}}_{n+1} = \left(2\vec{\mathbf{A}_{\boldsymbol{\nu}}}_n\vec{\mathbf{M}}_n^{-1}\vec{\mathbf{A}}_n - \vec{\mathbf{A}_{\mathbf{h}}}_n\right)\vec{\boldsymbol{X}}_n \tag{6.16}$$

If we define

$$\mathbf{S} := \vec{\mathbf{M}}_n + \tau_n\left(\vec{\mathbf{A}_{\mathbf{h}}}_n + \vec{\mathbf{A}}_n\vec{\mathbf{M}}_n^{-1}\vec{\mathbf{A}}_n\right) \text{ and } \vec{\boldsymbol{F}} := \left(2\vec{\mathbf{A}_{\boldsymbol{\nu}}}_n\vec{\mathbf{M}}_n^{-1}\vec{\mathbf{A}}_n - \vec{\mathbf{A}_{\mathbf{h}}}_n\right)\vec{\boldsymbol{X}}_n,$$

then we can rewrite equation (6.16) as

$$\mathbf{S}\vec{\boldsymbol{V}}_{n+1} = \vec{\boldsymbol{F}}. \tag{6.17}$$

The matrix $\mathbf{S}$ is symmetric and positive definite so PCG can be used to solve this system.

Similarly, according to (6.9), the previous Lemma yields

Lemma 6.1.3 (Geometric Biomembrane)**.** *Using the definitions at the beginning of Section 6.1.2, the matrix system for the biomembrane model is given by*

$$\begin{aligned}\vec{\mathbf{M}}_n\vec{\boldsymbol{V}}_{n+1} + \vec{\mathbf{A}}_n\vec{\boldsymbol{H}}_{n+1} + \tau_n\vec{\mathbf{A}_{\mathbf{h}}}_n\vec{\boldsymbol{V}}_{n+1} &= 2\vec{\mathbf{A}_{\boldsymbol{\nu}}}_n\vec{\boldsymbol{H}}_n - \vec{\mathbf{A}_{\mathbf{h}}}_n\vec{\boldsymbol{X}}_n + \lambda^{n+1}\vec{\boldsymbol{F}}_n^{\lambda} + \pi^{n+1}\vec{\boldsymbol{F}}_n^{\pi},\\ \vec{\mathbf{M}}_n\vec{\boldsymbol{H}}_{n+1} - \tau_n\vec{\mathbf{A}}_n\vec{\boldsymbol{V}}_{n+1} &= \vec{\mathbf{A}}_n\vec{\boldsymbol{X}}_n,\end{aligned} \tag{6.18}$$

where $(\vec{\boldsymbol{F}}_n^{\pi})_{i,k} = \int_{\Gamma_h^n}\boldsymbol{\phi}_{i,k}\cdot\boldsymbol{\nu}$ *and* $\vec{\boldsymbol{F}}_n^{\lambda} = \vec{\mathbf{M}}_n\vec{\boldsymbol{H}}_n$.

6.2 Fluid-Membrane Schemes

The discussion for the discretization of geometric evolution equations (section 6.1 gives useful guidelines as to how to proceed in certain aspects of the discretization

of the fluid-membrane problems. One example is what terms of the boundary force to make explicit when solving the system. We expect the solution of this problem to be *more regular* than the corresponding geometric model: in fact, before we equate the velocity of the membrane to a force (or variational derivative of energy), whereas now we equate the latter with the acceleration (which involves one more derivative).

6.2.1 Discrete Weak Formulation

The space time discretization of the Stokes system is standard in the finite element community, see for instance the monographs [Tem84, GR86, Glo03] and the references therein. Among all the stable finite element pairs, depending on the degree used to parametrize the boundary we have chosen either

- the "mini" element if the boundary is piecewise linear. This element consists of continuous piecewise linears enriched with cubic bubble functions for velocity and continuous piecewise linears for pressure.

- The Taylor-Hood element of order 1 if the boundary is piecewise quadratic. This element consists of continuous piecewise quadratics for velocity and continuous piecewise linears for pressure.

See Section 5.1.1 for more details on the definitions and properties of these elements. Therefore, the trace of the velocity $\boldsymbol{v}_h$ is in the same space as the finite element describing the interface, i.e it is continuous piecewise linear or quadratic on the boundary (see coupling equation (4.35)). A discussion of the importance of the finite element choices for moving free boundaries is reported in Section 7.1. $\mathbb{V}(\Omega_h^n)$

denotes the finite element space for the velocity and $\mathbb{Q}(\Omega_h^n)$ the one for the pressure. Also $\mathbb{V}(\Gamma_h^n)$ is the finite element space obtained as the trace of $\mathbb{V}(\Omega_h^n)$.

From the weak formulation (4.45) we obtain the following scheme: given an initial domain approximation Ω_h^0, a time partition $t_0 = 0 < \cdots < t_{M+1} = T$ of $[0, T]$, with time-step $\tau_n := t_{n+1} - t_n$, $n = 0, \ldots, M$, and an initial velocity $\boldsymbol{v}_0$, let $\boldsymbol{v}_h^0 \in \mathbb{V}(\Omega_h^0)$ be defined by

$$\int_{\Omega_h^0} \boldsymbol{v}_h^0 \cdot \boldsymbol{\phi}_h = \int_{\Omega_h^0} \boldsymbol{v}_0 \cdot \boldsymbol{\phi}_h, \qquad \forall \boldsymbol{\phi}_h \in \mathbb{V}(\Omega_h^0).$$

Find for $n \geq 0$, the velocity $\boldsymbol{v}_h^{n+1} \in \mathbb{V}(\Omega_h^n)$ and the pressure $p_h^{n+1} \in \mathbb{Q}(\Omega_h^n)$ such that

$$\begin{aligned} &\int_{\Omega_h^n} \boldsymbol{v}_h^{n+1} \cdot \boldsymbol{\phi}_h + \frac{\tau}{Re} \int_{\Omega_h^n} \left(\nabla \boldsymbol{v}_h^{n+1} + (\nabla \boldsymbol{v}_h^{n+1})^T \right) : \nabla \boldsymbol{\phi}_h - \tau \int_{\Omega_h^n} p_h^{n+1} \ \nabla \cdot \boldsymbol{\phi}_h \\ &\quad = \int_{\Omega_h^n} \boldsymbol{v}_h^n \cdot \boldsymbol{\phi}_h + \tau \int_{\Omega_h^n} \boldsymbol{b}(t^{n+1}) \cdot \boldsymbol{\phi}_h + \frac{\tau}{\alpha Re} \int_{\Gamma_h^n} \delta E_h^{n+1} \cdot \boldsymbol{\phi}_h, \qquad \forall \boldsymbol{\phi}_h \in \mathbb{V}(\Omega_h^n), \end{aligned} \tag{6.19}$$

and

$$\int_{\Omega_h^n} \nabla \cdot \boldsymbol{v}_h^{n+1} \ q_h = 0 \qquad \forall q_h \in \mathbb{Q}(\Omega_h^n), \tag{6.20}$$

are satisfied. And the new domain Ω_h^{n+1} is obtained from Ω_h^n and $\boldsymbol{v}_h^{n+1}$, by the parametrization

$$\boldsymbol{x}_h^{n+1} = \mathbf{Id}_{\Omega_h^n} + \tau_n \boldsymbol{v}_h^{n+1}, \tag{6.21}$$

maintaining the connectivity of Ω_h^n. This choice also updates the membrane with the velocity of the fluid. The quantity $\int_{\Gamma_h^n} \delta E_h^{n+1} \cdot \boldsymbol{\phi}_h$ in (6.19) will be specified below for the different problems. The number α can be either the capillary number Ca or the bending number Be (see Section 4.3.3).

Remark 6.2.1 (Time Discretization of Material Derivative). In equation (6.19) we have use the approximation $\dot{\boldsymbol{v}} \approx \frac{\boldsymbol{v}_h^{n+1}-\boldsymbol{v}_h^n}{\tau_n}$. This is justify because of equation (6.21) all mesh nodes are moved with the velocity field. And can be considered to be a particle advected with the fluid.

6.2.1.1 Capillarity

From the discussion on the discrete mean curvature flow scheme in section 6.1.1.1, we obtain (see equation (6.2))

$$\int_{\Gamma_h^n} \delta E_h^{n+1} \cdot \boldsymbol{\phi}_h = -\int_{\Gamma_h^n} \nabla_{\Gamma_h^n} \mathbf{Id}_{\Gamma_h^n} : \nabla_{\Gamma_h^n} \boldsymbol{\phi}_h - \tau \int_{\Gamma_h^n} \nabla_{\Gamma_h^n} \boldsymbol{v}_h^{n+1} : \nabla_{\Gamma_h^n} \boldsymbol{\phi}_h. \tag{6.22}$$

Plugging this in (6.19) and using $\alpha = Ca$ gives the discrete scheme for capillarity:

$$\begin{aligned}\int_{\Omega_h^n} \boldsymbol{v}_h^{n+1} \cdot \boldsymbol{\phi}_h + \frac{\tau}{Re} \int_{\Omega_h^n} \left(\nabla \boldsymbol{v}_h^{n+1} + (\nabla \boldsymbol{v}_h^{n+1})^T\right) : \nabla \boldsymbol{\phi}_h - \tau \int_{\Omega_h^n} p_h^{n+1} \ \nabla \cdot \boldsymbol{\phi}_h \\ = \int_{\Omega_h^n} \boldsymbol{v}_h^n \cdot \boldsymbol{\phi}_h + \tau \int_{\Omega_h^n} \boldsymbol{b}(t^{n+1}) \cdot \boldsymbol{\phi}_h \\ + \frac{\tau}{CaRe} \Bigg(- \int_{\Gamma_h^n} \nabla_{\Gamma_h^n} \mathbf{Id}_{\Gamma_h^n} : \nabla_{\Gamma_h^n} \boldsymbol{\phi}_h \\ - \tau \int_{\Gamma_h^n} \nabla_{\Gamma_h^n} \boldsymbol{v}_h^{n+1} : \nabla_{\Gamma_h^n} \boldsymbol{\phi}_h \Bigg) \qquad \forall \boldsymbol{\phi}_h \in \mathbb{V}(\Omega_h^n),\end{aligned} \tag{6.23}$$

and

$$\int_{\Omega_h^n} \nabla \cdot \boldsymbol{v}_h^{n+1} \ q_h = 0 \qquad \forall q_h \in \mathbb{Q}(\Omega_h^n). \tag{6.24}$$

The previous scheme is basically the one reported by Bänsch [Bän01].

6.2.1.2 Willmore

Here again from the discussion of the Willmore flow in section 6.1.1.2, equation (6.8) gives

$$\begin{aligned}\int_{\Gamma_h^n} \delta E_h^{n+1} \cdot \boldsymbol{\phi}_h = &- \int_{\Gamma_h^n} \nabla_{\Gamma_h^n} \boldsymbol{h}_h^{n+1} : \nabla_{\Gamma_h^n} \boldsymbol{\phi}_h \\ &- \frac{\tau}{2} \int_{\Gamma_h^n} |\tilde{\boldsymbol{h}}_h^n|^2 \nabla_{\Gamma_h^n} \boldsymbol{v}_h^{n+1} : \nabla_{\Gamma_h^n} \boldsymbol{\phi}_h \\ &+ 2 \int_{\Gamma_h^n} \left[(\mathbf{I} - \boldsymbol{\nu} \otimes \boldsymbol{\nu}) \nabla_{\Gamma_h^n} \bar{\boldsymbol{h}}_h^n \right] : \nabla_{\Gamma_h^n} \boldsymbol{\phi}_h \\ &+ \frac{1}{2} \int_{\Gamma_h^n} |\tilde{\boldsymbol{h}}_h^n|^2 \nabla_{\Gamma_h^n} \mathbf{Id}_h^n : \nabla_{\Gamma_h^n} \boldsymbol{\phi}_h \qquad \forall \boldsymbol{\phi}_h \in \mathbb{V}(\Gamma_h^n).\end{aligned} \tag{6.25}$$

Plugging this in (6.19) and using $\alpha = \rho V L^2$ and adding (6.6) gives the discrete scheme for the coupled Willmore problem:

$$\begin{aligned}\int_{\Omega_h^n} \boldsymbol{v}_h^{n+1} \cdot \boldsymbol{\phi}_h + &\frac{\tau}{Re} \int_{\Omega_h^n} \left(\nabla \boldsymbol{v}_h^{n+1} + (\nabla \boldsymbol{v}_h^{n+1})^T \right) : \nabla \boldsymbol{\phi}_h - \tau \int_{\Omega_h^n} p_h^{n+1} \ \nabla \cdot \boldsymbol{\phi}_h \\ &= \int_{\Omega_h^n} \boldsymbol{v}_h^n \cdot \boldsymbol{\phi}_h + \tau \int_{\Omega_h^n} \boldsymbol{b}(t^{n+1}) \cdot \boldsymbol{\phi}_h \\ &+ \frac{\tau}{\alpha Re} \Bigg(- \int_{\Gamma_h^n} \nabla_{\Gamma_h^n} \boldsymbol{h}_h^{n+1} : \nabla_{\Gamma_h^n} \boldsymbol{\phi}_h \\ &- \frac{\tau}{2} \int_{\Gamma_h^n} |\tilde{\boldsymbol{h}}_h^n|^2 \nabla_{\Gamma_h^n} \boldsymbol{v}_h^{n+1} : \nabla_{\Gamma_h^n} \boldsymbol{\phi}_h \\ &+ 2 \int_{\Gamma_h^n} \left[(\mathbf{I} - \boldsymbol{\nu} \otimes \boldsymbol{\nu}) \nabla_{\Gamma_h^n} \bar{\boldsymbol{h}}_h^n \right] : \nabla_{\Gamma_h^n} \boldsymbol{\phi}_h \\ &+ \frac{1}{2} \int_{\Gamma_h^n} |\tilde{\boldsymbol{h}}_h^n|^2 \nabla_{\Gamma_h^n} \mathbf{Id}_h^n : \nabla_{\Gamma_h^n} \boldsymbol{\phi}_h \Bigg) \qquad \forall \boldsymbol{\phi}_h \in \mathbb{V}(\Omega_h^n),\end{aligned} \tag{6.26}$$

and

$$\int_{\Omega_h^n} \nabla \cdot \boldsymbol{v}_h^{n+1} \ q_h = 0 \qquad \forall q_h \in \mathbb{Q}(\Omega_h^n), \tag{6.27}$$

6.2.1.3 Bending

This is the same as the scheme of section 6.2.1.2 upon adding the area constraint and taking $\alpha = Be$ in (6.19).

Remark 6.2.2. (Volume Constraint) The volume conservation is ensured (up to consistency) by the incompressibility condition (6.20). Therefore, the Lagrange multiplier π used in Section 6.1.1.3, is not needed anymore and can be "hidden" in the pressure. This is the option chosen in practice.

6.2.2 Matrix Formulation

Here we follow the notation of Section 6.1.2, the main addition is that now we have matrices both in the volume Ω and on the surface Γ. Recall that $\mathbb{V}(\Omega)$ is the finite element space for the velocity, isoparametric mesh and curvature; and $\mathbb{Q}(\Omega)$ is the space for the pressure. Also $\mathbb{V}(\Gamma)$ is the finite element space obtained as the trace of $\mathbb{V}(\Omega)$. Now we enlarge the matrix list defined in Section 6.1.2 with the volume matrices.

- $(\mathbf{M}_{\Omega_h})_{ij} = \int_{\Omega_h} \phi_i \phi_j, \qquad (\vec{\mathbf{M}}_{\Omega_h})_{ij} = (\mathbf{M})_{ij} \mathbf{I}_{d+1},$
- $(\mathbf{A}_{\Omega_h})_{ij} = \int_{\Omega_h} \nabla\phi_i \cdot \nabla\phi_j, \qquad (\vec{\mathbf{A}}_{\Omega_h})_{ij} = (\mathbf{A})_{ij} \mathbf{I}_{d+1},$
- $(\mathbf{B}_{\Omega_h})^m_{ij} = \int_{\Omega_h} \frac{\partial \phi_i}{\partial x_m} \psi_j.$

6.2.2.1 Matrix System for Capillarity

Lemma 6.2.1 (Capillarity)**.** *Using the definitions at the beginning of Section 6.1.2 and 6.2.2, let*

$$\vec{\mathbf{C}}_{\Omega_h^n} = \vec{\mathbf{M}}_{\Omega_h^n} + \frac{\tau_n}{Re}\vec{\mathbf{A}}_{\Omega_h^n} + \frac{\tau_n^2}{CaRe}\vec{\mathbf{A}}_{\Gamma_h^n}$$

and $(\vec{\boldsymbol{F}}_n)_{i,k} = \int_{\Gamma_h^n} \boldsymbol{\phi}_{i,k} \cdot \boldsymbol{b}(t^{n+1})$*. Then equation* (6.23) *is equivalent to*

$$\begin{bmatrix} \vec{\mathbf{C}}_{\Omega_h^n} & -\tau_n \mathbf{B}_{\Omega_h^n} \\ \mathbf{B}_{\Omega_h^n}{}^{\mathsf{T}} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \vec{\boldsymbol{V}}_{n+1} \\ P_{n+1} \end{bmatrix} = \begin{bmatrix} \vec{\mathbf{M}}_{\Omega_h^n}\vec{\boldsymbol{V}}_n + \tau_n \vec{\boldsymbol{F}}_n - \frac{\tau_n}{CaRe}\vec{\mathbf{A}}_{\Gamma_h^n}\vec{\boldsymbol{X}}_n \\ \mathbf{0} \end{bmatrix}. \tag{6.28}$$

Here two different methods to solve system (6.28) can be used. The first one is to solve the full system (6.28) either with a direct solver or a preconditioned GMRES. The second one is the Uzawa Method as explained in the following remark.

Remark 6.2.3. (Uzawa Method) At each time step, a Conjugate-Gradient Uzawa method is used to decouple the computation of the pressure p_h^{n+1} from the velocity $\boldsymbol{v}_h^{n+1}$. Refer to [Glo03] for more precision on the CG-Uzawa algorithm. As an inner loop, given the pressure p_h^{n+1}, equation (6.19) is solved using the same techniques as presented in Subsection 6.1.2.1.

6.2.2.2 Matrix System for Fluid-Biomembranes

Lemma 6.2.2 (Fluid Willmore)**.** *Using the definitions at the beginning of Section 6.1.2 and 6.2.2, let*

$$\vec{\mathbf{C}}_{\Omega_h^n} = \vec{\mathbf{M}}_{\Omega_h^n} + \frac{\tau_n}{Re}\vec{\mathbf{A}}_{\Omega_h^n} + \frac{\tau_n^2}{2BeRe}\vec{\mathbf{A}}_{\mathbf{h}\Gamma_h^n},$$

$$\vec{\boldsymbol{G}}_n = -\frac{\tau_n}{BeRe}\left(2\vec{\mathbf{A}}_{\boldsymbol{\nu}\Gamma_h^n}\vec{\boldsymbol{H}}_n + \frac{1}{2}\vec{\mathbf{A}}_{\mathbf{h}\Gamma_h^n}\vec{\boldsymbol{X}}_n\right),$$

and $(\vec{\boldsymbol{F}}_n)_{i,k} = \int_{\Gamma_h^n} \boldsymbol{\phi}_{i,k} \cdot \boldsymbol{b}(t^{n+1})$. *Then equation* (6.26) *is equivalent to*

$$\begin{bmatrix} \vec{\mathbf{C}}_{\Omega_h^n} & \frac{\tau_n}{2BeRe}\vec{\mathbf{A}}_{\Gamma_h^n} & -\tau_n \mathbf{B}_{\Omega_h^n} \\ \frac{\tau_n}{2BeRe}\vec{\mathbf{A}}_{\Gamma_h^n} & -BeRe\vec{\mathbf{M}}_{\Gamma_h^n} & \mathbf{0} \\ \mathbf{B}_{\Omega_h^n}{}^{\mathsf{T}} & \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \vec{\boldsymbol{V}}_{n+1} \\ \vec{\boldsymbol{H}}_{n+1} \\ P_{n+1} \end{bmatrix} = \begin{bmatrix} \vec{\mathbf{M}}_{\Omega_h^n}\vec{\boldsymbol{V}}_n + \tau_n \vec{\boldsymbol{F}}_n + \vec{\boldsymbol{G}}_n \\ \vec{\mathbf{A}}_{\Gamma_h^n}\vec{\boldsymbol{X}}_n \\ \mathbf{0} \end{bmatrix} \tag{6.29}$$

Remark 6.2.4. (Uzawa Method) Similarly to Remark 6.2.3 at each time step, a Conjugate-Gradient Uzawa method is used to decouple the computation of the pressure p_h^{n+1} from the velocity-curvature $(\boldsymbol{v}_h^{n+1}, \boldsymbol{h}_h^{n+1})$. Refer to [Glo03] for more precision on the CG-Uzawa algorithm. As an inner loop, given the pressure p_h^{n+1}, equation (6.19) is solved using the same techniques as presented in Subsection 6.1.2.2.

Chapter 7

Implementation of a Parametric AFEM

In this chapter we address the computational issues related to the implementation of the parametric AFEM for geometric evolution equations and coupled fluid-membrane problems. A set of computational tools including space and time adaptivity, mesh enhancement and discrete constraints implementation, is presented. These tools are crucial to successfully use the parametric FEM. It is important to mention the synergic nature of the tools. Even though individually each tool provides its contribution, the effect is multiplied when they collaborate with each other. For this to happen they need to be applied in the proper order.

In Section 7.1 we discuss the counterintuitive effect that a mismatch of the finite element spaces may have on problems involving curvature. In a personal communication with Kunibert Siebert it was mentioned the loss of half an order for capillary problems. We have seen the loss of a whole order in the case of mean curvature flow. We discover that this is associated to the violation of a geometric condition that reappears in the setting of refinements and coarsenings.

In Section 7.2 we propose a suitable remedy. Also here we deal with the issue of geometric adaptivity as means of describing the surface accurately with the minimal number of degrees of freedom. First we propose a geometric estimator based on the pointwise error. Then we define a geometric compatibility condition that is key for

the adaptivity not to deteriorate the flow. Based on this condition we provide a novel refinement procedure together with a theorem showing the benefits of it.

In Section 7.3 we present a novel method to compute the solutions of discrete systems with isoperimetric constraints. Some of its features are: the preservation of constraints to machine precision; same computational effort as the problem without constraints; and a more predictable and less oscillating behavior than the penalization method.

In Section 7.4 we deal with the issue of mesh improvement. When a parametric FEM is used to discretize a geometric evolution equation it will create a discrete flow of the mesh. Even if the initial mesh has a perfect quality, as it moves with the flow it will get distorted: the larger the overall domain deformation the more the mesh deteriorates. We present an optimization method novel in many aspects that improves the mesh quality, preserves the shape of its boundary, maintains the local meshsize, and produces negligible changes to the finite element functions defined on the mesh. Different cases are analyzed depending on the type of domain and the mesh degree.

In Section 7.5 we describe a novel hybrid affine-quadratic approach to the surface/boundary isoparametric elements. The idea is to keep the quadratic element not far from its affine support, but still allow it to have the characteristic rounded shape coming from the quadratic bubble. Then the affine techniques for mesh improvement and time-step adaptivity can be used on quadratic meshes.

In Section 7.6 a geometric timestep control is discussed. In general nonlinear time dependent fourth order problems present a highly varying time scale along its

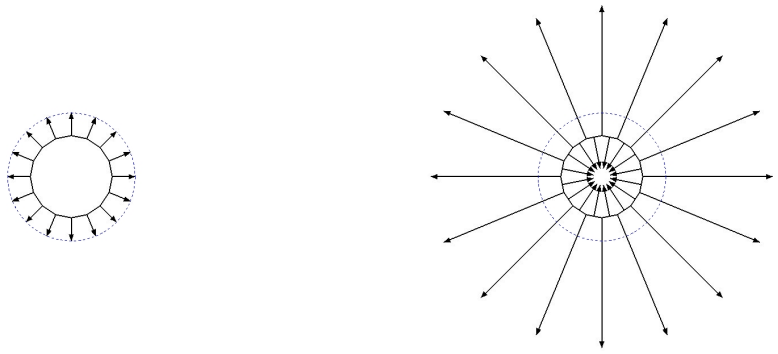
evolution. The timestep control presented here serves two geometric purposes. The first one is to ensure that it is not too big as to produce node crossing. Secondly, it guarantees that if the velocity is too small the timestep is big enough so that it does not take too many timesteps to produce a negligible evolution. The treatment is different from the one in [BMN05] in the sense that we use the element quality in the time selection method.

Finally in Section 7.7 we present the general full Algorithm where the order in which the tools previously described should be applied to potentiate themselves in a synergic way.

7.1 Effect of the Different Finite Element Spaces

Because of the nature of the parametric FEM approach (Section 5.7), the selection of the finite element spaces involved may have counterintuitive behavior depending on how the surface is approximated by the mesh. For instance, consider an affinely triangulated approximation of a smooth surface. The dynamics of the surface coordinate corresponding to a mean curvature flow (6.2) can be approximated with any polynomial degree. Formally, by analogy with the flat case, one would expect a better order of convergence when using higher polynomial degree for the flow. But this does not occur and in fact it could get even worse. This situation is due to a mismatch between the finite element spaces used for the flow and for the representation of the surface. What is unexpected is not the lack of improvement but rather the actual deterioration of the approximation when for example linear ele-

ments are replace by quadratics. Some light can be shed as to what happens with the following simple example. Suppose we approximate the unit circle with a mesh formed by line segments (affine elements). Let us use formula (5.14) to compute the curvature taking first the coordinates of the flow and the corresponding curvature in $\mathbb{S}_h^{1,1}$ (linear FEM) and then in $\mathbb{S}_h^{1,2}$ (quadratic FEM). The results are shown in Figure 7.1, the exact answer being radial arrows of length one pointing outwards. When using the degree pair $(1,1)$ a good approximation is observed, whereas employing the pair $(1,2)$ yields arrows with alternating directions and different lengths. An intuitive explanation can be grabbed from Figure 7.2. We see that when using

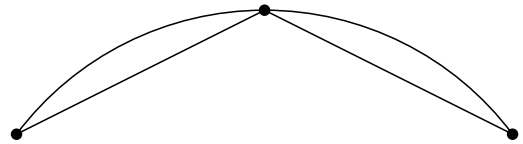


(a) Using piecewise linear elements. (b) Using piecewise quadratic elements.

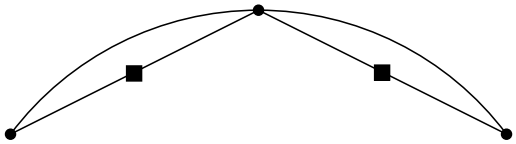
Figure 7.1: Vector curvature of the unit circle computed on a uniform mesh of degree one made of 16 segments using formula (5.14) with piecewise linear (left) and piecewise quadratic elements (right). The exact answer is given by radial arrows of unit length pointing outwards, indicated in the figure by the blue dashed circle. A good approximation is observed in 7.1(a) whereas 7.1(b) shows alternating directions with different arrow lengths. The arrows in the left figure have length 1, and both figures are plotted with the same scale.

piecewise quadratic functions for the coordinates of the flow but piecewise affine elements to represent the circle, we add degrees of freedom (the midpoint of the segments), which do not lie on the circle. This reinforces a mismatch between the

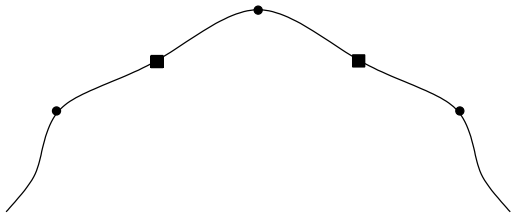
computational domain and the exact one. We have more degrees of freedom but we do not exploit them correctly as is reflected in the discrete curvature formula for example. The mismatch behavior observed in this one dimensional example also



(a) Piecewise linear representation of the circle with $\mathcal{P}^1$ finite elements.



(b) Piecewise linear representation of the circle with $\mathcal{P}^2$ finite elements. The midpoints lie outside the circle.



(c) Smooth curve that the weak formula for curvature assumes from the position of the nodes in Figure 7.2(b).

Figure 7.2: Zoom on piece of circle of Figure 7.1. If we use piecewise quadratic elements for the spaces where the curvature and coordinates belong but linear elements to represent the circle, we add degrees of freedom (the midpoint of the segments represented with a box). This degree of freedom is not on the circle, thereby providing misleading geometric information to the curvature formula which assumes somehow that the nodes describe another curve (shown in 7.2(c))in the best way they can. A similar situation also occurs when using piecewise linear elements and refining them (see Section 7.2).

happens for surfaces in three dimensions. Not surprisingly, the optimal order of convergence for piecewise quadratics elements is recovered immediately if we use quadratic isoparametric elements to represent the surface.

One may think at this point that it is extremely logical to use the same degree for the coordinate function describing the flow as for the elements approximating the surface. Even though this point is correct, what we point out is the unexpected

effect that happens if we do not. Using quadratic elements for the flow coordinates provides more degrees of freedom but does not change the polygonal shape of the surface. Then one would not expect a better order of approximation but would not expect a worse order either. Suppose that now we match the polynomial degree of the mesh and the coordinate function and we perform some refinement. A similar deterioration of the rate of convergence reappears again in this setting. In Section 7.2.2 we identify the source of conflict as the violation of some *discrete geometric condition*, that happens to be exactly the same as the one violated by the previous example, and we propose a suitable remedy.

7.2 Space Adaptivity

We would like to describe a surface accurately while keeping the number of degrees of freedom minimal, and so computationally affordable. A simple strategy for this is to equidistribute the pointwise error as proposed in [BMN05]. This is motivated by error estimates of geometric problems like mean curvature flow [LN05] and Laplace-Beltrami equation [DD07, Mek05, MMN]. In our case the surface is unknown, but it becomes crucial to be able to modify locally its resolution. This is achieved by means of refinements and coarsenings. The geometric estimator for this purpose (Section 7.2.1) is an approximation to the upper bound of the pointwise error of the domain approximation. Once it is known where to refine, say where the local indicators are relatively large, a decision has to be made as to where to place the newly created nodes (recall the surface is unknown). A new paradigm appears as for

example naive linear interpolation introduces a numerical artifact (Section 7.2.2). We also resolve the paradigm in Section 7.2.2.

7.2.1 Estimator

Recalling the interpolation result for surfaces of Lemma 5.3.2 when $p = q = \infty$ for the surface element K, we get

$$|\boldsymbol{l} - \boldsymbol{l}^{\gamma}|_{L^{\infty}(\tilde{K})} \leq C h_{\tilde{K}}^{\gamma+1} |\boldsymbol{l}|_{W_{\infty}^{k+1}(\tilde{K})}, \tag{7.1}$$

where C depends on the shape regularity σ of the triangulation. For the case of a polyhedral approximation ($\gamma = 1$) the right hand side of (7.1) can be approximated by the second fundamental form $\nabla_{\Gamma}\boldsymbol{\nu}$, as stated in the following Lemma.

Lemma 7.2.1 (Second Derivatives Approximation)**.** *Given $\epsilon > 0$ there is h and $\boldsymbol{l}$ such that if $h_{\tilde{K}} < h$ then*

$$|\boldsymbol{l}|_{W_{\infty}^{k+1}(\tilde{K})} \leq (1+\epsilon) |\nabla_{\Gamma}\boldsymbol{\nu}|_{L^{\infty}(K)}.$$

Sketch of proof. The idea is that a smooth surface is locally like a graph then following [GT83, Section 14.6] and taking $\boldsymbol{l}$ to be the one defined by the graph it follows that the second derivatives of $\boldsymbol{l}$ are the second fundamental form. □

The estimator we use is a computable approximation to $\eta(K) = h_K^2 |\nabla_{\Gamma}\boldsymbol{\nu}|_{\infty,K}$. For a shape regular polyhedral surface this quantity can be approximated by

$$\eta^*(K) = \int_K |\nabla_K \boldsymbol{\nu}|, \tag{7.2}$$

where $|.|$ denotes any matrix norm. At this point it is crucial to make a remark on how $|\nabla_K \boldsymbol{\nu}|$ is computed. Since the surface is approximated by affine elements

($\gamma = 1$) the normal is only piecewise constant, to get a meaningful estimator we use a gradient recovery technique [HSWW01]. More precisely, let N_i be a triangulation node and $\omega_i = \{K \in \mathcal{T} : N_i \in K\}$ be the star center at N_i. Now we construct a piecewise linear normal by defining its nodal value at N_i by $\Sigma_{K\in\omega_i} \boldsymbol{\nu}_K \,\mathrm{meas}(K)$. Then its gradient which will be piecewise constant can be computed and used in (7.2). If the mesh is of degree $\gamma \geq 2$ one could use the previous estimator and in

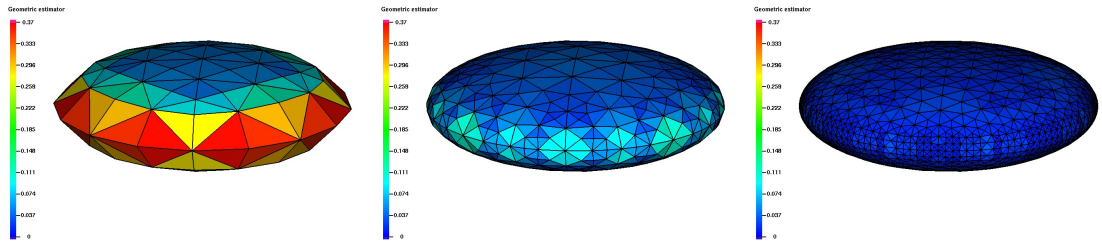


Figure 7.3: Geometric estimator for piecewise ellipsoid. The estimator (7.2) is used to refine a 3x3x1 ellipsoid. The nodes are projected after the refinement. Observe how the areas of high curvature are the ones refined more and also how the color becomes uniform showing the equidistribution of the estimator.

fact averaging is not necessary as one could use equation (5.15) which is convergent in view of Theorem 5.4.1. But the main drawback is that in the light of (7.1) this estimator is not sharp. For example for quadratic elements, the power of h_K as well as the number of derivative should be 3. In this case an alternative is to use

$$h_S \|\nu_+ - \nu_-\|_{L^\infty(S)}$$

where S is any side and ν_+, ν_- are the unit normals of the adjacent elements K_+ and K_- to S. The heuristics behind is as follows:

$$h_K^{\gamma+1} \|D^{\gamma+1} z\|_{L^\infty}$$

is the correct quantity on the right-hand side of (7.1), where γ is the polynomial degree and z represents the surface as a graph, provided $z \in C^{\gamma+1}$. Now

$$\|\frac{\nu_+ - \nu_-}{h_K}\|_{L^\infty(\partial K)} \approx \|D^2(z - Iz)\|_{L^\infty(K)}$$

and the right-hand side exhibits the correct order of convergence regardless of the regularity of z (because it is an interpolation error). If $z \in C^{\gamma+1}$, then $\|D^2(z - Iz)\|_{L^\infty(K)} \leq Ch_K^{\gamma-1}$. Therefore, the replacement for the right hand side of (7.1) reads

$$h_K^2\|D^2(z - Iz)\|_{L^\infty(K)} \approx h_K\|\nu_+ - \nu_-\|_{L^\infty(\partial K)}.$$

The refinement algorithm consists of the bisection of simplices. In two dimensions we use the newest vertex bisection and in three dimensions the bisection procedure of Kossaczky [Kos94]. For the mesh refinement strategy we use the maximum strategy [SS05], described in Algorithm 7.2.1.

7.2.2 Geometrically Consistent Refinement

As it was mentioned in the introduction and commented at the end of Section 7.1, providing a new degree of freedom to represent a surface that happens not to lie exactly on the surface may lead to a mismatch that manifests as a numerical artifact. This is also the case when we want to refine an element on a surface and we need to specify the position of the new nodes. A numerical artifact appears for example if we use affine elements to approximate a surface and place the new nodes by linear interpolation. The problem seemingly does not appear if we use quadratic isoparametric elements and quadratic interpolation to place the new node. But this

Algorithm 7.2.1 Mesh Adaptation. Maximum Strategy

1: **procedure** ADAPT MESH($\mathcal{T}$)
2: Let $\gamma, \gamma_c \in (0,1)$ with $\gamma > \gamma_c$
3: Compute $\eta(K)$ for each $\eta(K) \in \mathcal{T}$
4: $\eta = \max(\eta(K), \eta(K) \in \mathcal{T}$
5: **for** $\eta(K) \in \mathcal{T}$ **do**
6: **if** $\eta(K) > \gamma\eta$ **then**
7: Mark K for refinement
8: **end if**
9: **end for**
10: Refine
11: **for** $\eta(K) \in \mathcal{T}$ **do**
12: **if** $\eta(K) < \gamma_c\eta$ **then**
13: Mark K for coarsening
14: **end if**
15: **end for**
16: Coarsen
17: **end procedure**

is not the case as a careful look reveals the loose of an order in the convergence rate. In this section we identify the new nodes mismatch with the violation of a geometric relation. Building on this relation we present a refinement algorithm for piecewise polynomial surfaces of any degree which approximate a smooth free surface and neither creates numerical artifacts nor reduces the optimal order of convergence.

The key idea is the use of extra geometric information that is available to define the refinement. More precisely, we use the approximation to vector mean curvature $\boldsymbol{h}$. From formula (3.14), the mean curvature $\boldsymbol{h}$ of a surface Γ is related to the position $\mathbf{Id}_\Gamma$ by the equation

$$\boldsymbol{h} = -\Delta_\Gamma \mathbf{Id}_\Gamma.$$

In the discretizations presented in chapter 6, this equation is satisfied approximately by requiring that

$$\boldsymbol{h}_h^{n+1} = -\Delta_{\Gamma_h^n} \boldsymbol{x}_h^{n+1} \tag{7.3}$$

is satisfied in a weak form. Upon refinement of Γ_h^n, both $\boldsymbol{x}_h^{n+1}$ and $\boldsymbol{h}_h^{n+1}$ are enlarge with new degrees of freedom. If an upper bar denotes refinement by linear interpolation then the equation $\overline{\boldsymbol{h}_h^{n+1}} = -\Delta_{\overline{\Gamma_h^n}} \overline{\boldsymbol{x}_h^{n+1}}$ may be violated. We refer to equation (7.3) as the *geometric consistent condition*.

By examining the mean curvature flow scheme (6.2), one may be tempted to think that, as $\boldsymbol{h}_h$ does not enter the scheme, just doing linear interpolation for $\boldsymbol{x}_h$ should work fine. However, in that case, the approximation to the velocity deteriorates. This deterioration may eventually be compensated with the pass of time due to the smoothing effect of the mean curvature flow. The point is that we introduce a non negligible numerical artifact upon doing a geometrically inconsistent refinement that would not have appeared otherwise. As a numerical example, Figure 7.4 shows the effect of linear refinement for the motion of the unit circle by mean curvature. From here one could infer how disastrous this effect could be for more complicated flows such as the Willmore flow, which use point values of curvature in

the scheme (equation (6.8)). Below we propose a novel refinement procedure, and

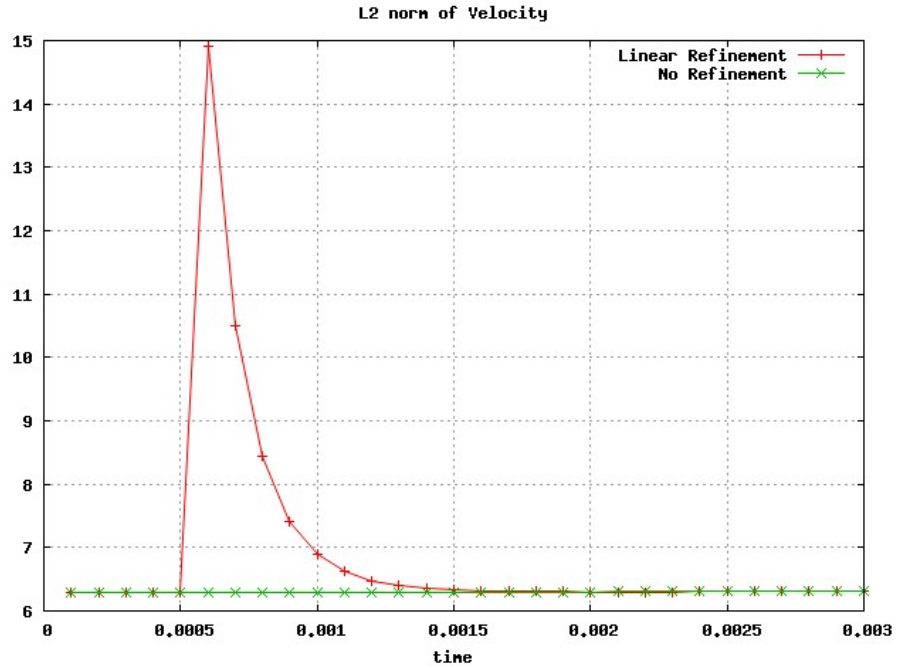


Figure 7.4: Mean Curvature Flow for the unit circle with a fixed time step of 1.0e-4. An almost uniform polygonal approximation to the unit circle with 64 nodes is subject to the discrete flow of equation (6.2) and after 5 iterations is globally refined, thereby giving 128 nodes. The picture shows the L^2 norm of the velocity with and without refinement: the dramatic spike due to refinement is damped by the regularizing effects of the flow.

we prove that with this method $\overline{\boldsymbol{h}_h} = -\Delta_{\overline{\Gamma_h}}\overline{\boldsymbol{x}_h}$, and the approximations to $\boldsymbol{x}$ and $\boldsymbol{h}$ are as good as before introducing the new degrees of freedom. It is important to realize that on adding degrees of freedom there is no hope to expect $\overline{\boldsymbol{x}_h}$ to be a better approximation than $\boldsymbol{x}_h$. Only after a solve we expect the former to be better.

7.2.2.1 The Method

Let Γ_h^n be a piecewise polynomial approximation to Γ_{t_n}. Let $\boldsymbol{h}_h^{n+1}$ and $\boldsymbol{x}_h^{n+1}$ in $\mathbb{S}_h^\gamma(\Gamma_h^n)$ be approximations to mean curvature and position respectively. Also assume

the following geometric consistent condition $\boldsymbol{h}_h^{n+1} = -\Delta_{(\Gamma_h^\gamma)^n}\boldsymbol{x}_h^{n+1}$ is satisfied. To simplify the notations in this Section we use $(\Gamma_h^\gamma)^n = \Gamma_h^n$ as γ is fixed. Observe that in general $\boldsymbol{x}_h^{n+1}$ is not Γ_h^n but rather Γ_h^{n+1}. We also assume that some criteria is available to decide where more or less resolution would be beneficial for the flow, which in our case means having a marking decision on where to perform refinements and coarsenings (cf. Section 7.2). The issue treated here is how to perform refinement if our only knowledge of the domain is the current approximate surface Γ_h^n that we are about to refine. Algorithm 7.2.2 gives a method to add more resolution to Γ_h^n that satisfies the geometric consistent condition after the refinement and that does not change the approximation error that we had before the refinement for both position and curvature. Finally we define rigorously what we mean by a marking decision: If $\mathcal{M} = \{(K, j) : K \in \mathcal{T}, j \in \mathbb{N}_0\}$ denotes the marked set, then element K is to be bisected at least j times. For example if the pair $(K, 2) \in \mathcal{M}$ then the element $K \in \mathcal{T}$ will be refined twice. $\mathcal{M}$ is obtained by using some estimator and a marking strategy. In line 2 of Algorithm 7.2.2, $\overline{\Gamma_h^n}$ is obtained by refining Γ_h^n by isoparametric

Algorithm 7.2.2 Surface Refinement Algorithm

1: **procedure** $\left(\overline{\Gamma_h^n}, \overline{\boldsymbol{x}_h^{n+1}}, \overline{\boldsymbol{h}_h^{n+1}}\right) =$ SURF REF$(\Gamma_h^n, \boldsymbol{h}_h^{n+1}, \boldsymbol{x}_h^{n+1}, \mathcal{M})$

2: $\quad \overline{\Gamma_h^n} =$ isoparametric refinement(Γ_h^n)

3: $\quad \overline{\boldsymbol{h}_h^{n+1}} =$ Interpolation$(\boldsymbol{h}_h^{n+1})$

4: $\quad$ Compute $\overline{\boldsymbol{x}_h^{n+1}}$ the solution to $\overline{\boldsymbol{h}_h^{n+1}} = -\Delta_{\overline{\Gamma_h^n}}\overline{\boldsymbol{x}_h^{n+1}}$

5: **end procedure**

interpolation as explain in the next remark.

Remark 7.2.1 (Isoparametric interpolation and refinement). An isoparametric element is by definition given by the deformation of the master element $\hat{K}$ through the mapping $F(\hat{\boldsymbol{x}}) = \Sigma \boldsymbol{x}_i \hat{p}_i(\hat{\boldsymbol{x}})$. Where $\hat{p}_i$ are the Lagrange basis function on $\hat{K}$ and $\{\boldsymbol{x}_i\}$ the Lagrange node in K. Isoparametric interpolation means given a function $f : K \to \mathbb{R}$ the interpolant is given by $\mathcal{I}f = \Sigma f(\boldsymbol{x}_i)\hat{p}_i \circ F^{-1}$. Refinement is done by bisection. First the bisection edge is bisected in the master element $\hat{K}$ this creates the corresponding new Lagrange nodes $\hat{x}_j^*$, and by isoparametric interpolation we obtain the new nodes that bisect K, i.e., $\boldsymbol{x}_j^* = \mathcal{I}F(\hat{\boldsymbol{x}}_j^*)$.

This in particular implies that $\overline{\Gamma_h^n} = \Gamma_h^n$, i.e. they are the same surface. But $\text{card}(\overline{\mathcal{T}}) > \text{card}(\mathcal{T})$. Before the refinement $(\Gamma_h^n, \mathcal{T})$ is a piecewise polygonal approximation to $\Gamma(t_n)$ meaning that there is lift $\boldsymbol{l}$. By construction given $\bar{K} \in \overline{\mathcal{T}}$ there exists a unique $K \in \mathcal{T}$ such that $\bar{K} \subset K$. Then we can define the lift $\bar{\boldsymbol{l}}$ by $\bar{\boldsymbol{l}}|\bar{K} = \boldsymbol{l}|K$ for $\bar{K} \in \overline{\mathcal{T}}$. And given a parametrization $(\hat{K}, F_K : \hat{K} \to K))$ of Γ_h^n we can define the associated parameterization $(\hat{\bar{K}}, \bar{F}_{\bar{K}})$ of $\overline{\Gamma_h^n}$ by $\hat{\bar{K}} = (F_K)^{-1}(\bar{K})$ and $\bar{F}_{\bar{K}} = (F_K)|_{\hat{\bar{K}}}$. Now we are in a position to prove that the Algorithm 7.2.2 does what it promises.

Theorem 7.2.2 (Geometrically Consistent Refinement)**.** *With the definitions and hypothesis assumed in Algorithm 7.2.2 the following is satisfied*

1. $\|\boldsymbol{h} - \overline{\boldsymbol{h}_h^{n+1}}\|_{L^2(\overline{\Gamma_h^n})} = \|\boldsymbol{h} - \boldsymbol{h}_h^{n+1}\|_{L^2(\Gamma_h^n)}$;
2. $|\mathbf{Id}_{\Gamma(t_{n+1})} - \overline{\boldsymbol{x}_h^{n+1}}|_{H^1(\overline{\Gamma_h^n})} \leq \mathcal{A}$; *and*
3. $\overline{\boldsymbol{h}_h^{n+1}} = -\Delta_{\overline{\Gamma_h^n}} \overline{\boldsymbol{x}_h^{n+1}}$.

where $\mathcal{A}$ *is the optimal upper bound from Theorem 5.5.2 for* $|\mathbf{Id}_{\Gamma(t_{n+1})} - \boldsymbol{x}_h^{n+1}|_{H^1(\Gamma_h^n)}$.

Proof. Lines 2 and 3 in Algorithm 7.2.2 imply that $\boldsymbol{h}_h^{n+1}$ and $\overline{\boldsymbol{h}_h^{n+1}}$ are the same functions defined on the same domain, then statement 1 is satisfied. By line 4 in Algorithm 7.2.2 statement 3 is true. To prove statement 2 observe that $\mathbf{Id}_{\Gamma(t_n)}$ is the solution of the Laplace Beltrami equation with $\boldsymbol{h}$ as the right hand side. Both $\boldsymbol{x}_h^{n+1}$ and $\overline{\boldsymbol{x}_h^{n+1}}$ are approximations to $\mathbf{Id}_{\Gamma(t_n)}$. Then invoking Theorem 5.5.2 and observing that $\|\mathbf{A}\|_{L^\infty(\Gamma_h^n)} = \|\bar{\mathbf{A}}\|_{L^\infty(\overline{\Gamma_h^n})}$ and $\mathbb{S}_h(\Gamma_h^n) \subset \mathbb{S}_h(\overline{\Gamma_h^n})$). □

Remark 7.2.2 (Implementation Trick). In practice Algorithm 7.2.2 works very fast and reliably as far as our simulations have shown. It is also possible to fix the old nodes (as a Dirichlet condition in line 4) and only solve for the position of the newly created nodes. In our simulation two or three iterations of the conjugate gradient method were enough to solve line 4, making the refinement method quite efficient computationally.

Remark 7.2.3 (Effect of Coarsening). So far we have been mostly talking about refinement. To a smaller extend the problem also appears when doing coarsening.

7.3 Constraints

In Section 4.1 we described how to impose constraints via Lagrange multipliers. Now we present a method to implement these isoperimetric type of constraints at the discrete level, that preserves them to machine precision. First we describe the method as it applies to a flow with volume and area constraints (for example see Problem 4.2.7 and the discrete scheme of Section 6.1.1.3). Then we generalize it for the discrete version of the general system (4.4). Consider the continuous system

(4.3). In chapter 6 we considered discrete schemes for different energies E. The first step is to take the terms involving the multipliers explicitly in the discretization. At the matrix level the discrete system can be written as

$$\mathcal{E}^n(V_h^{n+1}, \dots) = F_w + \lambda^{n+1} F_a + \pi^{n+1} F_v. \tag{7.4}$$

The idea then is to solve the above system by solving a similar system for three different right hand sides and find the multipliers using the area and volume conservation relations

$$\text{Area}(\Gamma_h^{n+1}) = \text{Area}(\Gamma_h^n) \quad \text{and} \quad \text{Vol}(\Gamma_h^{n+1}) = \text{Vol}(\Gamma_h^n). \tag{7.5}$$

More precisely, let $\boldsymbol{V}_{wh}{}^{n+1}$, $\boldsymbol{V}_{ah}{}^{n+1}$ and $\boldsymbol{V}_{vh}{}^{n+1}$ be the solutions of

$$\mathcal{E}^n(\boldsymbol{V}_{wh}{}^{n+1}) = F_w,$$

$$\mathcal{E}^n(\boldsymbol{V}_{ah}{}^{n+1}) = F_a,$$

$$\mathcal{E}^n(\boldsymbol{V}_{vh}{}^{n+1}) = F_v,$$

respectively. Using the linearity of system (7.4) we have

$$\boldsymbol{V}_h^{n+1} = \boldsymbol{V}_{wh}{}^{n+1} + \lambda^{n+1} \boldsymbol{V}_{ah}{}^{n+1} + \pi^{n+1} \boldsymbol{V}_{vh}{}^{n+1}$$

where λ^{n+1} and π^{n+1} are determined so that (7.5) are satisfied. To this end, observe that given V_{wh}^{n+1}, V_{ah}^{n+1} and V_{vh}^{n+1}, the couple $(\lambda^{n+1}, \pi^{n+1})$ is a root of $f : \mathbb{R}^2 \to \mathbb{R}$ defined by

$$f(\lambda, \pi) := \begin{bmatrix} \text{Area}(\Gamma_h(\lambda, \pi)) - \text{Area}(\Gamma_h^n) \\ \\ \text{Vol}(\Gamma_h(\lambda, \pi)) - \text{Vol}(\Gamma_h^n) \end{bmatrix}, \tag{7.6}$$

where

$$\Gamma_h(\lambda, \pi) := \Gamma_h^n + \tau \left(V_{wh}^{n+1} + \lambda V_{ah}^{n+1} + \pi V_{vh}^{n+1} \right).$$

A Newton method can now be used to find the roots λ^{n+1} and π^{n+1} of f. The derivatives of f can be obtained from the rules of shape differential calculus.

Lemma 7.3.1 (Derivatives of f)**.** *The derivatives of the function f of equation* (7.6) *is given by*

$$Df = \begin{bmatrix} \int_{\Gamma_h^n} \nabla_{\Gamma_h} \cdot \boldsymbol{V}_{ah} & \int_{\Gamma_h^n} \nabla_{\Gamma_h} \cdot \boldsymbol{V}_{vh}) \\ \int_{\Gamma_h^n} \boldsymbol{\nu} \cdot \boldsymbol{V}_{ah} & \int_{\Gamma_h^n} \boldsymbol{\nu} \cdot \boldsymbol{V}_{vh}) \end{bmatrix} \tag{7.7}$$

Proof. Using the velocity method of the shape differential calculus of Section 3.2 with t replaced by λ or π and Theorem 4.1.1 the result follows. □

Remark 7.3.1 (Initial Guess). A good computable initial guess for the Newton method can be obtained from the formulas

$$\begin{aligned} \text{Area}(\lambda, \pi) &= \frac{1}{d} \int_\Gamma \nabla_\Gamma \cdot \mathbf{Id}, \\ \text{Vol}(\lambda, \pi) &= \frac{1}{d+1} \int_\Omega \nabla \cdot \mathbf{Id} = \frac{1}{d+1} \int_\Gamma \mathbf{Id} \cdot \boldsymbol{\nu}. \end{aligned}$$

If we use the approximations $\Gamma(\lambda, \pi) \approx \Gamma^n$ and $\mathbf{Id}_\Gamma \approx \mathbf{Id}_{\Gamma^n} + \tau \boldsymbol{V}_h^{n+1}$, then

$$\text{Area}(\lambda, \pi) \approx \frac{\tau}{d}\left(\int_{\Gamma^n} \nabla_{\Gamma_h^n} \cdot \boldsymbol{V}_w + \lambda \int_{\Gamma^n} \nabla_{\Gamma_h^n} \cdot \boldsymbol{V}_a + \pi \int_{\Gamma^n} \nabla_{\Gamma_h^n} \cdot \boldsymbol{V}_v\right) + \text{Area}(\Gamma) \tag{7.8}$$

$$\text{Vol}(\Gamma(\lambda, p)) \approx \frac{\tau}{d+1}\left(\int_{\Gamma^n} \boldsymbol{n} \cdot \boldsymbol{V}_w + \lambda \int_{\Gamma^n} \boldsymbol{\nu} \cdot \boldsymbol{V}_a + \pi \int_{\Gamma^n} \boldsymbol{\nu} \cdot \boldsymbol{V}_v\right) + \text{Vol}(\Gamma) \tag{7.9}$$

Let $\alpha_i = \int_\Gamma \nabla_\Gamma \cdot \boldsymbol{V}_i$ and $\beta_i = \int_\Gamma \boldsymbol{\nu} \cdot \boldsymbol{V}_i$ with $i \in \{w, a, v\}$, then we can solve the following system for λ and p

$$\begin{bmatrix} \alpha_a & \alpha_v \\ \beta_a & \beta_v \end{bmatrix} \begin{bmatrix} \lambda \\ \pi \end{bmatrix} = \begin{bmatrix} -\alpha_w \\ -\beta_w \end{bmatrix}, \tag{7.10}$$

to get an initial guess.

In Figures 7.5, 7.6 and 7.7 we show the features of the method for a 3x3x1 ellipsoid evolved with a Willmore flow with surface area and enclosed volume constraints. Similarly, for the general case (4.4) of N isoperimetric constraints of the

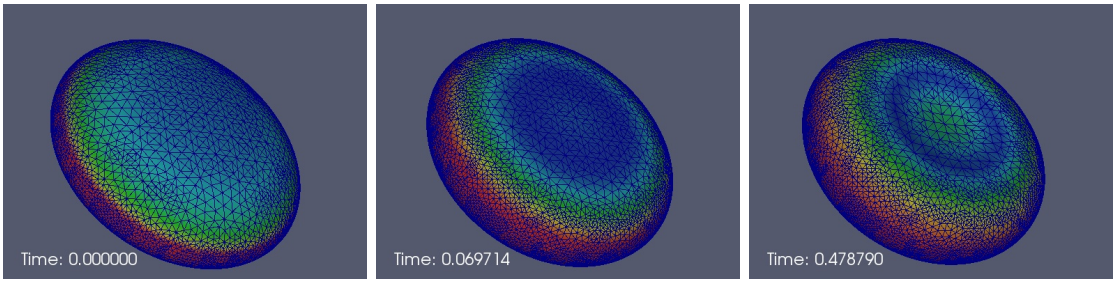


Figure 7.5: This sequence shows the evolution of an initial ellipsoid of aspect ratio 3x3x1 under the bending flow of Problem 4.2.7 subject to surface area and enclose volume constraints. Figures 7.6 and 7.7 show how the constraints are preserved exactly together with the behavior of the multipliers.

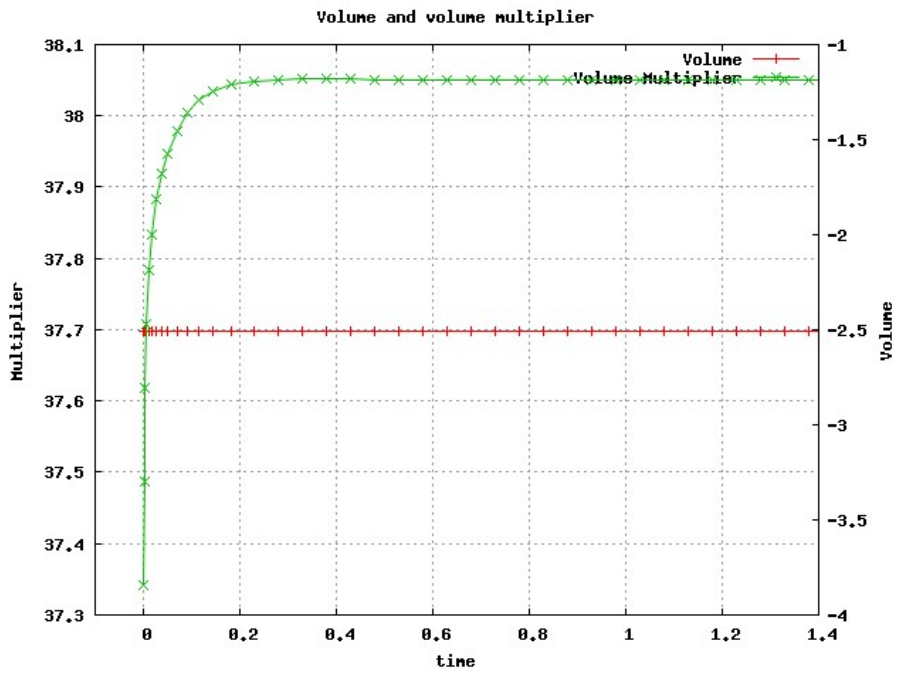


Figure 7.6: Graph for the volume and volume multiplier corresponding to simulation depicted in Figure 7.5. Observe how the constraint is preserve to machine precision and that at equilibrium there is no oscillation of the multiplier. A penalization methods usually exhibits and oscillatory behavior when reaching equilibrium.

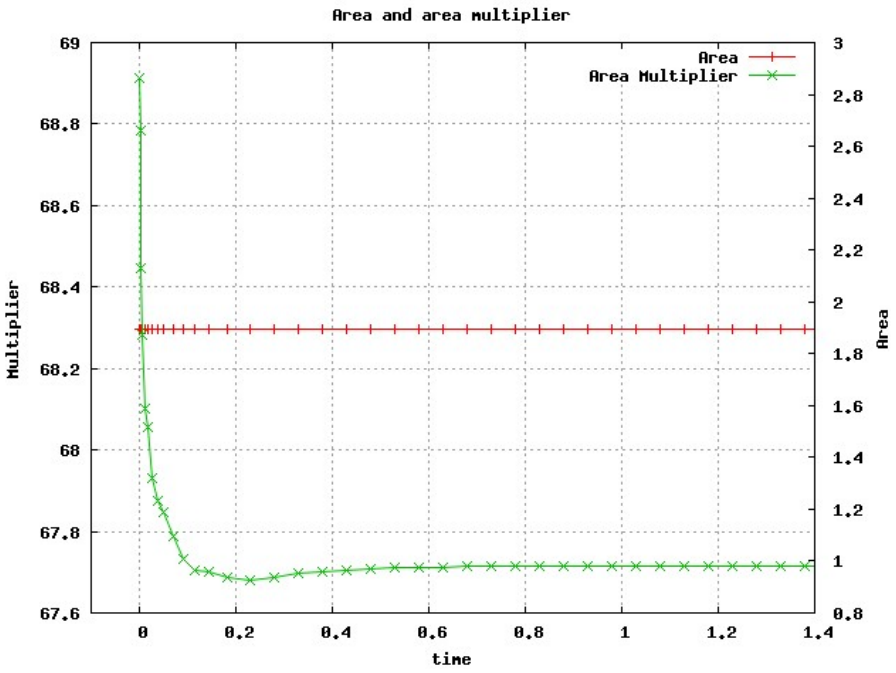


Figure 7.7: Graph for the area and area multiplier corresponding to simulation depicted in Figure 7.5. Observe how the constraint is preserve to machine precision and that at equilibrium there is no oscillation of the multiplier. A penalization methods usually exhibits and oscillatory behavior when reaching equilibrium.

form $F_i(\Gamma) = \int_\Gamma f_i$, the previous discussion leads to a matrix level discrete system that can be written as

$$\mathcal{E}^n(V_h^{n+1}, \dots) = F_0 + \sum_{i=1}^{N} \lambda_i^{n+1} \, F_i, \tag{7.11}$$

together with N scalar constraints

$$F_i(\Gamma_t) = F_i(\Gamma_0) \quad i = 1, \dots, N. \tag{7.12}$$

The method consists in solving $N + 1$ matrix systems

$$\mathcal{E}^n({\boldsymbol{V}_{ih}}^{n+1}) = F_i,$$

and find $\boldsymbol{\lambda} \in \mathbb{R}^N$ a root of $f : \mathbb{R}^N \to \mathbb{R}$ give by

$$f(\boldsymbol{\lambda}) = \begin{bmatrix} F_1(\Gamma_h(\boldsymbol{\lambda}) - F_1(\Gamma_0) \\ \vdots \\ F_N(\Gamma_h(\boldsymbol{\lambda}) - F_N(\Gamma_0) \end{bmatrix}. \tag{7.13}$$

Remark 7.3.2 (Implementation Issues). If a direct solver is used to solve the system, then it is only necessary to do the factorization once and use it to solve all the systems. This implies that the computational cost of using this method to impose the constraints is basically the same as not having the constraints. On the other hand if an iterative solver is used then the system has to be solved N number of times, but each system is independent from the others so a parallel implementation is very straightforward.

Remark 7.3.3 (Comparison with Penalization Method). The previous method exhibits a couple of advantages over penalization: penalization is sensitive to the penalization parameter and the solution oscillates when reaching equilibrium. These drawbacks do not happen with the previous method.

7.4 Mesh Improvement

In the finite element community mesh generation is assumed as a given. The main effort goes to design a method to compute discrete approximations to solutions of PDEs and study their convergence when the meshsize goes to zero.

Mesh generation is a community by itself. They tend to think in the following way: given a mesh, is it of sufficient quality to be passed to the consumer? The

consumer can be a finite element code, a video game designer or the movie industry. This way of thinking differs from the PDE rooted thinking of the finite element community [FG00].

If a parametric FEM is used to discretize a geometric evolution equation it will create a discrete flow of the mesh. Even if the initial mesh has a perfect quality, as it moves with the flow it will get distorted: the larger the overall domain deformation the larger mesh deterioration. Thus, it is crucial to include something in the method to preserve a good quality mesh along the deformation. Observe that for us mesh quality control is part of the design of a robust finite element method.

We start the section with a discussion of mesh improvement methods in particular optimization and smoothing. Then we survey the concepts of mesh quality and objective functions. In Section 7.4.3 we present the geometric optimization algorithm that we specialize to the different scenarios of application.

7.4.1 Optimization and Smoothing Techniques

Techniques for mesh improvement such as smoothing, optimization and edge swapping can be classified as either maintaining the connectivity or acting on the connectivity of the mesh. To allow refinements and coarsenings (Section 7.2) the mesh is implemented using a binary tree structure. This inhibits the application of the second type of improvements.

The classical example of a global smoothing method is Laplacian smoothing (see [FG00] Section 18.4.1). For moving domains, the theory of elasticity can be

used to smooth the mesh after moving the boundary. One example is the harmonic extension method [Gas01, Bän01, Kos06]. These methods are ad hoc and do not necessarily improve the mesh quality. Their advantage relies on being easy to implement and computationally cheap. But interpolation of finite element functions is extremely costly as it involves a global mesh search for each node.

Optimization techniques use classical optimization strategies. The idea is to define a smooth cost function over the set of all vertices [LAF97], and find a minimizer. This problem is almost impossible to solve globally so in practice it is approximated by defining local subproblems on stars. Then iterations of the Gauss-Seidel type are performed, each optimizing one local subproblem. This means that the iterations are performed in a sequential order and the previous iterations affect the latter. In particular the output will depend on the ordering of the iterations.

We propose a mesh optimization which consists in a reallocation of the nodes (tangential in the case of surfaces) such that:

- improves the mesh quality,
- preserves the shape of its boundary,
- maintains the local mesh size, and
- produces negligible changes to the finite element functions defined on the mesh.

The last point is desirable to minimize the effect that the reallocation transfers to the flow. It is attained by interpolating the finite element functions on the old mesh for the new mesh position. One reason why a local optimization method is preferred

to global smoothing is because as each iteration is performed locally on a star, a finite element function can be cheaply updated by interpolation over the old star. The smoothing algorithm will differ depending on:

- the degree of the mesh,
- whether it is a bulk or a surface mesh,
- whether it is an interior or a boundary node,
- whether the domain is known or not.

7.4.2 Quality Metrics and Objective Functions

There is a big number of element quality metrics in the literature. Simply put one expects the quality of a triangle to be one if equilateral, zero if degenerate and negative if inverted.

In 2001, Knupp [Knu01] worked on a theory to define quality metrics. It is based on the Jacobian matrix of a map from an ideal reference element. Building on an algebraic framework that uses the matrix norm, trace and determinant he classifies what quantities are meaningful and what are redundant in the definition of a quality metric.

Let $\bar{K}$ be the perfect quality element and let $F : \bar{K} \to K$, be the unique affine mapping $F = \mathbf{S}\bar{\boldsymbol{x}} + \boldsymbol{s}$ that sends vertices of $\bar{K}$ to vertices of K. Then $\mathbf{S}$ is the Jacobian matrix of F and let $\sigma := \det(\mathbf{S})$. Under the previous framework it can be

shown that

$$q_2(K) = \frac{2\sigma}{\mathrm{tr}(\mathbf{S}^\mathsf{T}\mathbf{S})} \quad \text{and} \quad q_3(K) = \frac{3\sigma^{(2/3)}}{\mathrm{tr}(\mathbf{S}^\mathsf{T}\mathbf{S})} \tag{7.14}$$

define good quality metrics in two and three dimensions respectively (see [Knu01]). Quality metrics (7.14) and slight modification of them have been the choice in our computations. It is important to remark that the theory explained here works for any reasonable quality metric.

Consider the mesh $(\mathcal{N}, \mathcal{T})$. Let $N_i \in \mathcal{N}$ be a node of coordinates $\boldsymbol{x}$ and $\omega_i = \{K \in \mathcal{T} : N_i \in K\}$ be the star center at N_i. An optimization function can be derived from an element quality metrics as the p-norm of $\eta(K) := \frac{1}{q(K)}$, the reciprocal of the element quality. More precisely, given $p \geq 1$

$$\Upsilon_i(\boldsymbol{x}) := \left(\sum_{K \in \omega_i} \eta(K)^p \right)^{1/p}, \tag{7.15}$$

defines a star objective function to be minimized. Figure 7.8 shows the level set of the function Υ for a two dimensional star of formed by three triangles. As can be seen from the picture if the singular barrier at the exterior edges is crossed or if the mesh is tangled the optimization will not converge. In this case a variant of η to deal with tangling can be found in [ERM$^+$03].

For surfaces we develop a similar quantity η_S that we proceed to describe. Recall from (7.14) that

$$q_2(K) = \frac{2\sigma}{\mathrm{tr}(\mathbf{S}^\mathsf{T}\mathbf{S})},$$

observe that $\sigma^2 = \det(\mathbf{S}^\mathsf{T}\mathbf{S})$ so up to a sign $\sigma = \sqrt{\det(\mathbf{S}^\mathsf{T}\mathbf{S})}$. In the case of a surface element K, the function F becomes $F : \bar{K} \subset \mathbb{R}^2 \to K \subset \mathbb{R}^3$ and $\mathbf{S}$ cannot have a determinant, but $\mathbf{S}^\mathsf{T}\mathbf{S}$ is a 2 by 2 invertible matrix. Then we can make sense of

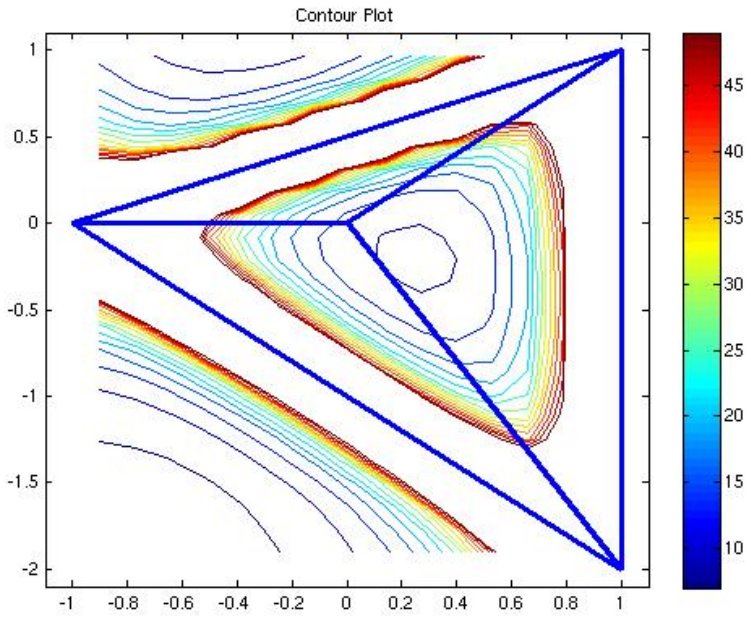


Figure 7.8: Contour plot of objective function Υ for a two dimensional star of formed by three triangles with $p = 2$. The argument of the function is the position of the star center. From the picture is can be infer that a center at approximately $(0.25, -0.2)$ minimizes Υ. Also observe that when the center approaches the star boundary the function goes to infinity (refer to as the singular boundary barrier). If the optimization tool crosses the edge then it will not converge.

(7.14) in the following way,

$$q_S(K) = \frac{2\sqrt{\det(\mathbf{S}^\mathsf{T}\mathbf{S})}}{\operatorname{tr}(\mathbf{S}^\mathsf{T}\mathbf{S})}. \tag{7.16}$$

and the corresponding

$$\eta_S(K) = \frac{\operatorname{tr}(\mathbf{S}^\mathsf{T}\mathbf{S})}{2\sqrt{\det(\mathbf{S}^\mathsf{T}\mathbf{S})}}. \tag{7.17}$$

For optimization routines it is important to be able to evaluate the derivatives of η.

Lemma 7.4.1 (Derivatives of η)**.** *Let $\alpha \in \{x_1, \ldots, x_{d+1}\}$ and η the function defined in* (7.14)*. Then for $d = 1$*

$$\partial_\alpha \eta = 2\eta \left[\frac{\partial_\alpha \mathbf{S} : \mathbf{S}}{\mathbf{S} : \mathbf{S}} - \frac{\partial_\alpha \sigma}{3\sigma} \right], \tag{7.18}$$

and for $d = 2$

$$\partial_\alpha \eta = 2\eta \left[\frac{2\partial_\alpha \mathbf{S} : \mathbf{S}}{\mathbf{S} : \mathbf{S}} - \frac{\partial_\alpha \sigma}{\sigma} \right]. \tag{7.19}$$

Proof. The proof is done for $d = 2$ as the case $d = 1$ is similar. Using the quotient and chain rules in (7.19) we get

$$\begin{aligned} \partial_\alpha \eta &= \frac{1}{3} \left[\frac{2\ \partial_\alpha \mathbf{S} : \mathbf{S}\ \sigma^{2/3} - \mathbf{S} : \mathbf{S}\ \frac{2}{3}\partial_\alpha \sigma}{\sigma^{4/3}} \right] \\ &= \frac{2\ \mathbf{S} : \mathbf{S}}{3\sigma^{2/3}} \left[\frac{2\partial_\alpha \mathbf{S} : \mathbf{S}}{\mathbf{S} : \mathbf{S}} - \frac{\partial_\alpha \sigma}{\sigma} \right] \end{aligned}$$

And the result follows. □

Remark 7.4.1 (Derivative of σ). Recall that $\sigma = \det(\mathbf{S})$, using the chain rule and the derivative of the determinant from Lemma 3.3.1 we get

$$\partial_\alpha \sigma = \det(\mathbf{S})\, \mathrm{tr}((\partial_\alpha \mathbf{S})\mathbf{S}^{-1}).$$

7.4.3 Geometric Optimization Algorithm

The general process for mesh improvement using local geometric optimization is described in Algorithm 7.4.1 . Given the star ω_i the optimization process (line 4) will find new coordinates for its center such that Υ_i is minimize and thus the star quality is improved. Some remarks about the algorithm are pertinent. They will be explained in details later in this section.

Remark 7.4.2 (Quadratic Meshes). Algorithm 7.4.1 is stated for affine elements. Still we will make use of it when working with quadratics. This is possible because we propose the use of a hierarchical approach to quadratics based on Theorem 5.6.1. This means that to optimize a quadratic star, first we optimize the affine base using

Algorithm 7.4.1 General mesh optimization algorithm.

1: **procedure** IMPROVE MESH(Mesh)
2: Build list of affine stars $\mathcal{L} = \{\omega_i\}$
3: **for** $\omega_i \in \mathcal{L}$ **do**
4: Minimize Υ_i
5: Interpolate FE functions at the new center.
6: **end for**
7: **end procedure**

Algorithm 7.4.1 and then we use a proper handling of the quadratic part. The details which are not trivial are the subject of Section 7.5.

Remark 7.4.3 (Interpolation). Even though for a pedagogical better understanding line 5 says "new center", interpolation has to be performed at all interior nodes. If the function to be interpolated is piecewise linear then there is only one interior point (the center), but if it is piecewise quadratic the interior nodes also include the interior edges midpoints.

Before describing the details involve in the implementation Algorithm 7.4.1 Figure 7.9 shows mesh smoothing in action. The full simulation is presented in Section 8.5.1. In the following Subsubsection we discuss lines 4 and 5 of Algorithm 7.4.1 in details. The treatment of these lines is different depending on the dimension or codimension of the mesh.

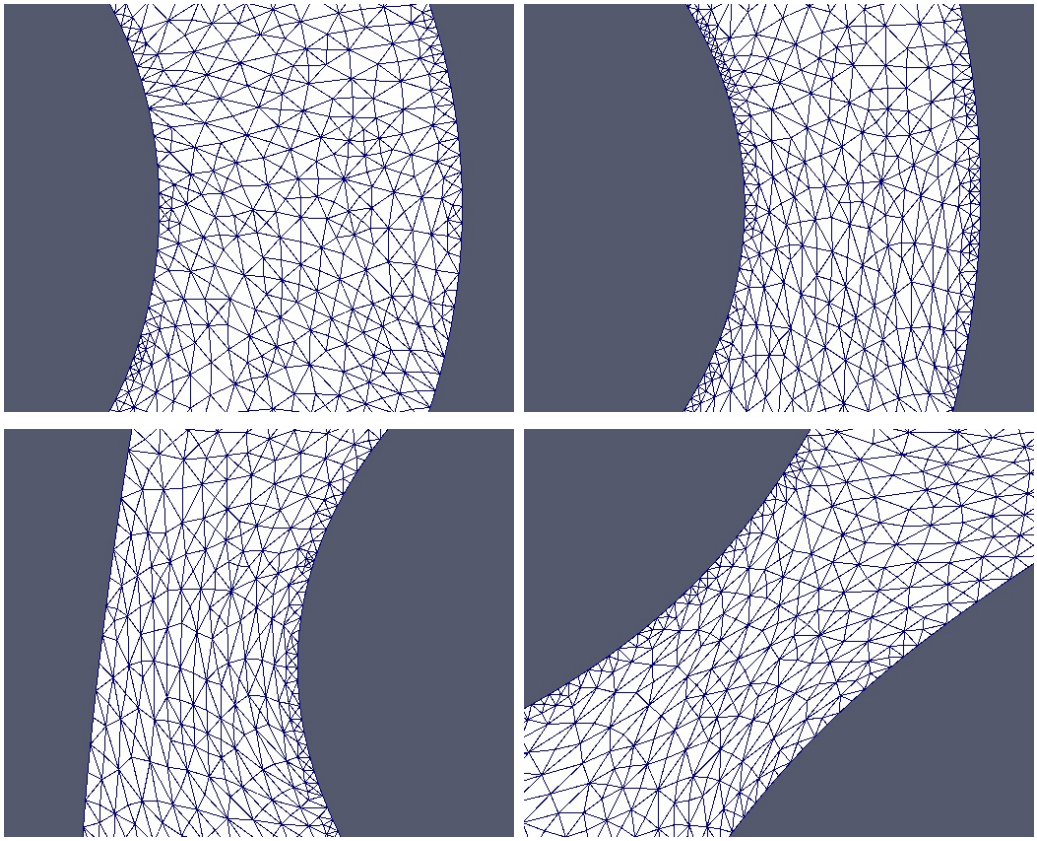


Figure 7.9: This is a zoom on the mesh for the simulation presented in full in Section 8.5.1. In this simulation the domain suffers quite a dramatic motion. Not only it evolves with a constrained bending surface force but it is also subject to an external sheering force that cause it to rotate faster and faster in time. Even though all the nodes are moving with the flow the mesh preserves its quality due to the smoothing described in Algorithm 7.4.1 being applied in each time iteration. Zoom on meshes for iterations 0, 20, 50 and 135 are shown.

7.4.3.1 Interior Volume Star

This is the case of a $(d+1)$-dimensional affine mesh in $\mathbb{R}^{d+1}$ with the center of a star being an interior node. For the minimization of line 4 the objective function (7.15) is used. Below we describe line 5, i.e, how we perform the finite element function interpolation after a new center x^* has been found. It is helpful at this point to read the explanation in Figure 7.10 before and while reading Algorithm 7.4.2. In line

Algorithm 7.4.2 Interpolation on a Star

1: **procedure** INTERPOLATE FUNCTION(f, ω, x^*)

2: Let $\mathcal{D} = \{\bar{N}_j : \bar{N}_j \text{ is a dof of } \mathbb{X}_f \text{ interior to } \omega\}$

3: **for** $\bar{N}_j \in \mathcal{D}$ **do**

4: Let x_j^* new coordinate of $\bar{N}_j$.

5: Find $K_j \in \omega$ such that $x_j^* \in K_j$

6: $\boldsymbol{\lambda}_j = \boldsymbol{\lambda}(x_j^*)$:

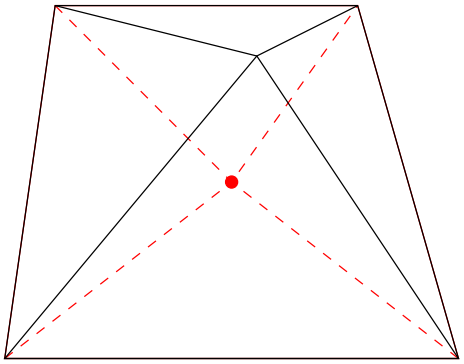
7: $f(x_j^*) = \text{interpolate}(f, \boldsymbol{\lambda}_j, K_j)$

8: **end for**

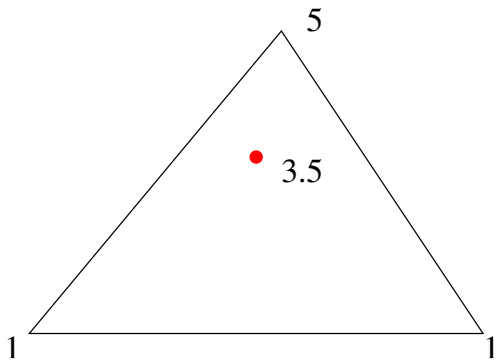
9: **end procedure**

4 of Algorithm 7.4.1, for a given star ω_i a new coordinate x_i^* for its center node N_i has being found. We assume there is a list of finite element functions define on the mesh $(\{N_i\}, \mathcal{T})$ that have to be updated to reflect the change in the mesh. In line 2 of Algorithm 7.4.2, given a function f in the finite element space $\mathbb{X}_f$ we collect in the set $\mathcal{D}$ the degrees of freedom of $\mathbb{X}_f$ that are interior points of the star ω_i. For each $\bar{N}_j \in \mathcal{D}$ the new coordinate x^* of the center will induce new coordinates for $\bar{N}_j$ that we call x_j^* (line 4). In lines 5 and 6 we search to which element $K \in \omega$ the point x_j^* belong to and what is its barycentric coordinate $\boldsymbol{\lambda}_j = \boldsymbol{\lambda}(x_j^*)$ in this element. Finally, in line 7 , knowing the barycentric coordinates of x_j^* the actual interpolation is computed.

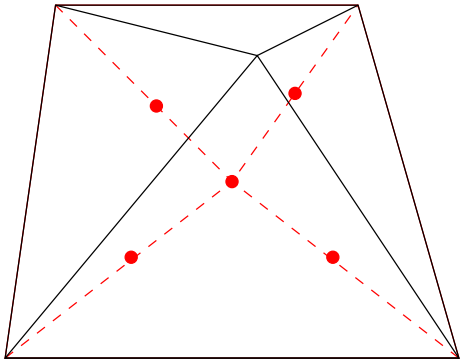
Remark 7.4.4 (Order of improvement iterations). Algorithm 7.4.1 is not invariant under the order in which the stars are optimized.In general this is random, but it seems possible that defining a smart ordering would lead to faster improvement.



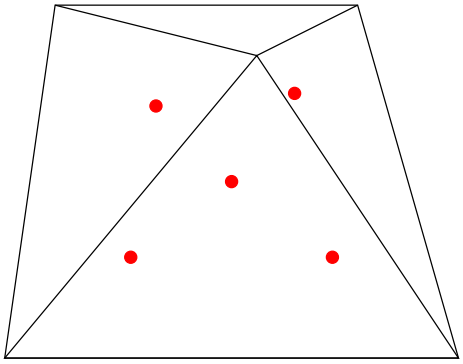
(a) Original and improved star.



(b) Interpolation of FE function at new point



(c) New quadratic interior nodes.



(d) Search elements for each red node

Figure 7.10: Optimization quality improvement and finite element interpolation on a 2D affine star. The old star where mesh optimization has been performed is in black. The new center and corresponding new star are depicted in red. Figure 7.10(b) illustrates the process of interpolation for a piecewise linear function. First the element holding the new center and its barycentric coordinates with respect to this element have to be found among all the elements of the original star. Then interpolation is performed on the element. Figures 7.10(c) and 7.10(d) illustrate the case of piecewise quadratic interpolation. In this case the search and interpolation process described before has to be repeated for each new interior node.

Remark 7.4.5 (Number of Interior dofs). Suppose $\mathbb{X}_f$ is the Lagrange finite element space of degree 2, and $d+1 = 2$. If ω is a star with n elements, then $\text{card}(\mathcal{D}) = n+1$.

To illustrate Algorithms 7.4.1 and 7.4.2 for a particular example consider the

initial unit circle of Figure 7.11. We applied a sequence of geometric mesh opti-

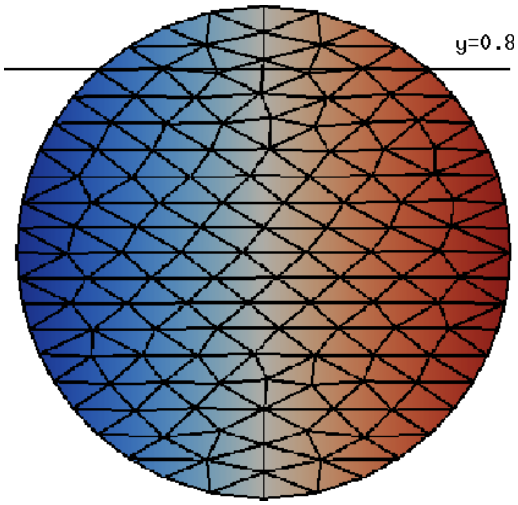


Figure 7.11: Initial affine mesh of unit circle. The color represents a linear scalar function defined on it. A sequence of geometric mesh optimization is going to be applied.

mization as described in Algorithms 7.4.1 and 7.4.2. Figure 7.12 shows the meshes at some of this iterations. The maximum, mean and minimum element quality as a function of the number of iteration is depicted in Figure 7.13. It can be seen from the picture how the minimum and mean quality is improved and the maximum is not changed much. In this example, the final mean quality is close to 0.95, which is an equilateral triangle for the eye. Also the minimum is of very high quality.

The other matter is the interpolation of functions defined on the mesh after the optimization has changed it. For illustration this linear function is defined on the initial circle mesh. Algorithm 7.4.1 is applied with and without the interpolation step. Without the interpolation step the nodal values of the function are transported with the flow. As the function is linear interpolation preserves it exactly through the smoothing. Figure 7.14 shows a comparison of the function values along the line

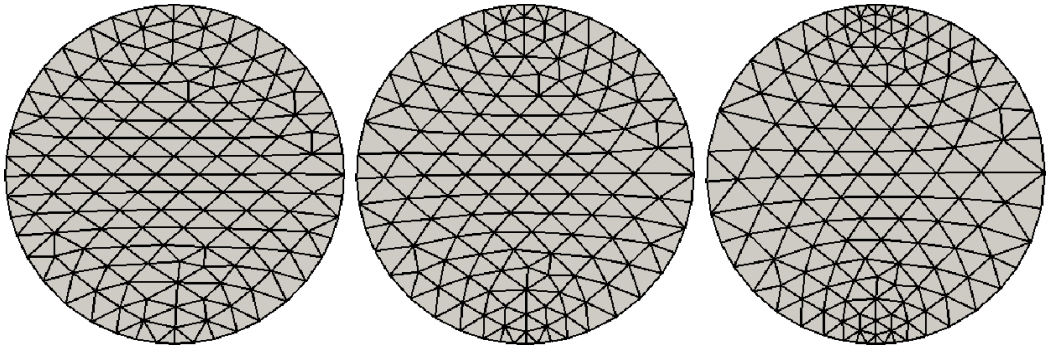


Figure 7.12: Subsequent meshes for iterations 3, 10 and 20, when the smoothing method of Algorithm 7.4.1 has being applied to the unit circle of Figure 7.11.

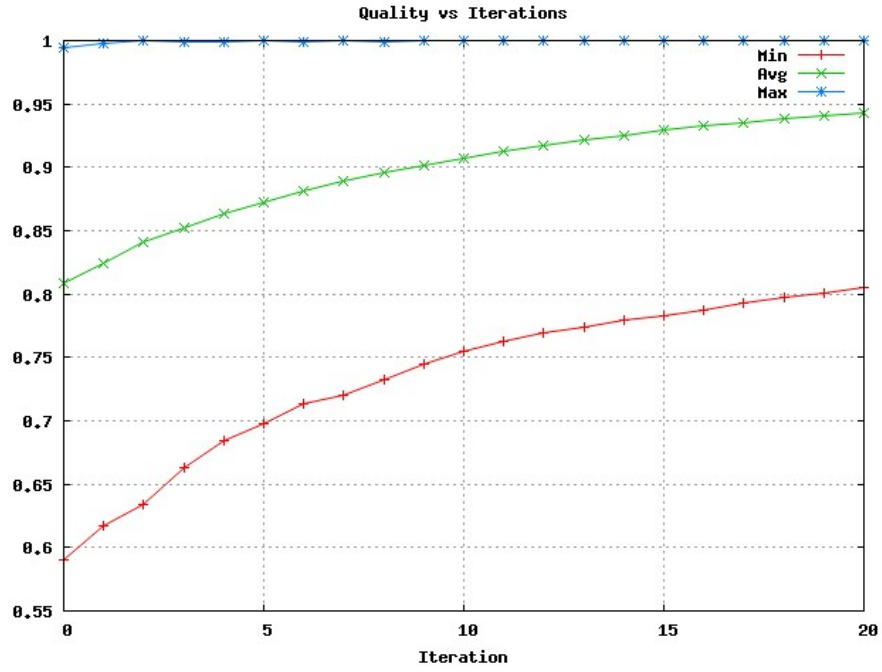


Figure 7.13: Plot of the mesh quality (minimum, mean and maximum element quality) as a function of the number of iterations for the smoothing described in figure 7.12. It can be seen from the picture how the minimum and mean quality is improved and the maximum is not changed much. In this example, the final mean quality is close to 0.95, which is an equilateral triangle for the eye. Also the minimum is of very high quality.

$y = 0.8$ with and without the interpolation step.

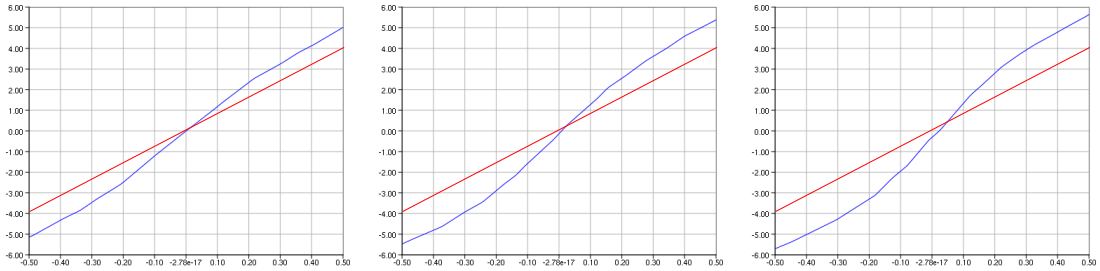


Figure 7.14: A linear function is defined on the initial circle mesh of Figure 7.11. Optimization Algorithm 7.4.1 is applied with and without the interpolation step. Without the interpolation step the nodal values of the function are transported with the flow. As the function is linear interpolation preserves it exactly through the smoothing. The graphs show the function values with and without interpolation step over the probe line $y = 0.8$ for iterations 3,10 and 20. The value should remain constant and it does when the interpolation step is applied, otherwise observe that when it is not applied the function looses it linearity.

7.4.3.2 Boundary Volume Star

This is the case of a $(d+1)$-affine mesh in $\mathbb{R}^{d+1}$ with the center of the star being a boundary node. For the minimization of line 4 the objective function (7.15) is used subject to the constraint that the minimizer is on the boundary. This constraint brings some additional complications. The first one is related to the knowledge of the boundary. For mesh generation (Section 9) the boundary is known and the extra constraint equation is available to be added to the optimization code. In the case of a free boundary problem (as the ones of chapter 6) we only know the boundary by its approximation, that by the way, it is the one that we are trying to enhance. So a decision has to be made as to what the boundary is. In this case two options are possible: either use some smooth reconstruction of the boundary (e.g. splines) and pass it to the optimization code as a constraint, or replace the boundary part of the star by an approximating flat d-dimensional star and do the smoothing here as

explained in Section 7.4.3.3. The second complication is the interpolation of finite element functions. As the new node may be located where there was no mesh before Algorithm 7.4.2 may fail. Again in this case our approach is to use the surface flat star and a lift as described in Section 7.4.3.3.

7.4.3.3 Surface Star

This is the case of a d-dimensional affine mesh in $\mathbb{R}^{d+1}$. The center of the star has to be reallocated to a new position that minimizes an objective function subject to the constraint of remaining in a position that preserves the shape of the discrete boundary. The objective function to optimize in this case is the one obtained in equation (7.17). Two different scenarios are possible, either the boundary is known (as it is the case of mesh generation), or all we know about the boundary is the current mesh (as it is the case for free boundary problems). In the first case the smooth function defining the surface is passed to the optimization code as a constraint. A particular case of this scenario when the surface is described as a deformation of simple domains is studied in Section 9.

We proceed to explain a smoothing method used when the domain is unknown. The key idea is to work on a projection plane Π so to reduce the problem to the previous discussion. Consider a surface affine star ω_i, we want to find a plane Π and an injection $P : \omega_i \to \Pi$ as close as possible to the identity function (see Figure 7.15). The idea is that we want the quality of the projected star $\tilde{\omega}_i = P(\omega_i)$ to be as close as possible to the quality of ω_i. At this point optimization is performed on

$\tilde{\omega}_i$ using Algorithm 7.4.1. In this case line 4 generates the new center $\tilde{x}^*$. All that remains to do is to map the new $\tilde{\omega}_i^*$ to $\tilde{\omega}_i$ using the lift $l = P^{-1}$.

For the interpolation step, line 5 of Algorithm 7.4.1, Algorithm 7.4.2 can be used without change provided that P is linear (e.g. the orthogonal projection to Π). The reason is that P being linear implies that the barycentric coordinates are invariant. More precisely, let $K \in \omega_i$, $\tilde{K} = P(K)$, $\boldsymbol{x} \in K$ and $\tilde{\boldsymbol{x}} = P(\boldsymbol{x})$, then $\boldsymbol{\lambda}_K(\boldsymbol{x}) = \boldsymbol{\lambda}_{\tilde{K}}(\tilde{\boldsymbol{x}})$. As a particular case of interpolation, if we pass the coordinates as one of the functions f to be interpolated, we obtain the new surface star ω_i^*. The previous discussion is summarized in Algorithm 7.4.3.

Algorithm 7.4.3 Optimization of a Surface Star

1: **procedure** SURF STAR OPTIM(ω_i)
2: Find Π and $P : \omega_i \to \Pi$
3: Let $\tilde{\omega}_i = P(\omega_i)$
4: Minimize $\tilde{\Upsilon}_i$
5: Use Alg. 7.4.2 on $(\tilde{\omega}_i, \tilde{x}^*, f)$
6: **end procedure**

Remark 7.4.6 (Computation of Π). One computationally convenient way to find the plane Π is by defining it to be the unique plane perpendicular to the star normal through the star center of mass. The normal has to be understood in the average sense discussed in Section 7.2.1.

Suppose that with the affine surface star ω we pass a quadratic vector function to be interpolated to Algorithm 7.4.3. Assume now that this quadratic function is the one defining a quadratic isoparametric element. Figure 7.16 shows what happens

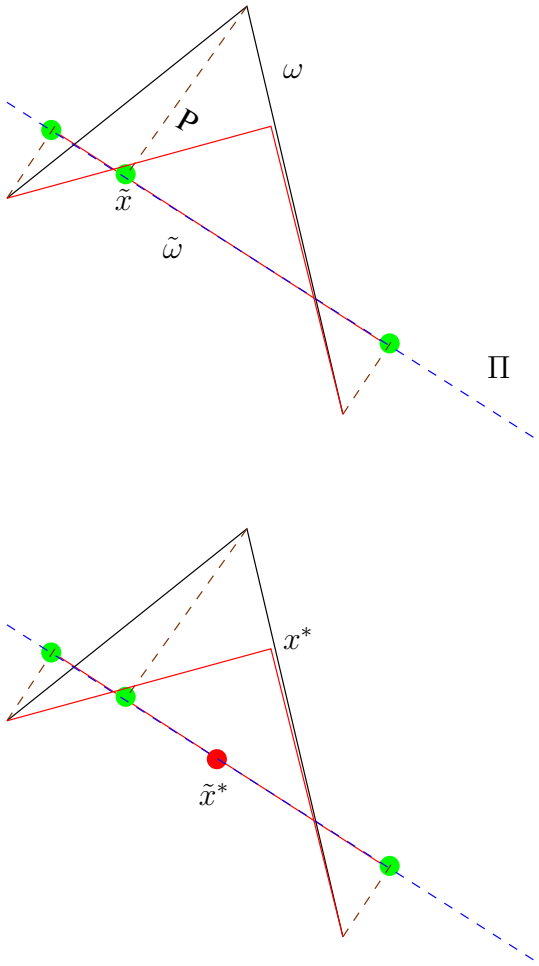


Figure 7.15: Smoothing process for an affine surface star. The picture on the left shows a 1D curve start in $\mathbb{R}^2$. The star ω (which has 2 elements) is shown together with a projection plane Π and the projection map P that define the projected star $\tilde{\omega}$. A quality optimization routine is applied on $\tilde{\omega}$ producing a new center $\tilde{x}^*$ (picture on the right). The new star $\tilde{\omega}^*$ is then lifted by $l = P^{-1}$ to yield the new surface star ω^*.

to the quadratic star if Algorithm 7.4.3 is applied to the corresponding affine star

and the quadratic coordinates are passed for interpolation.

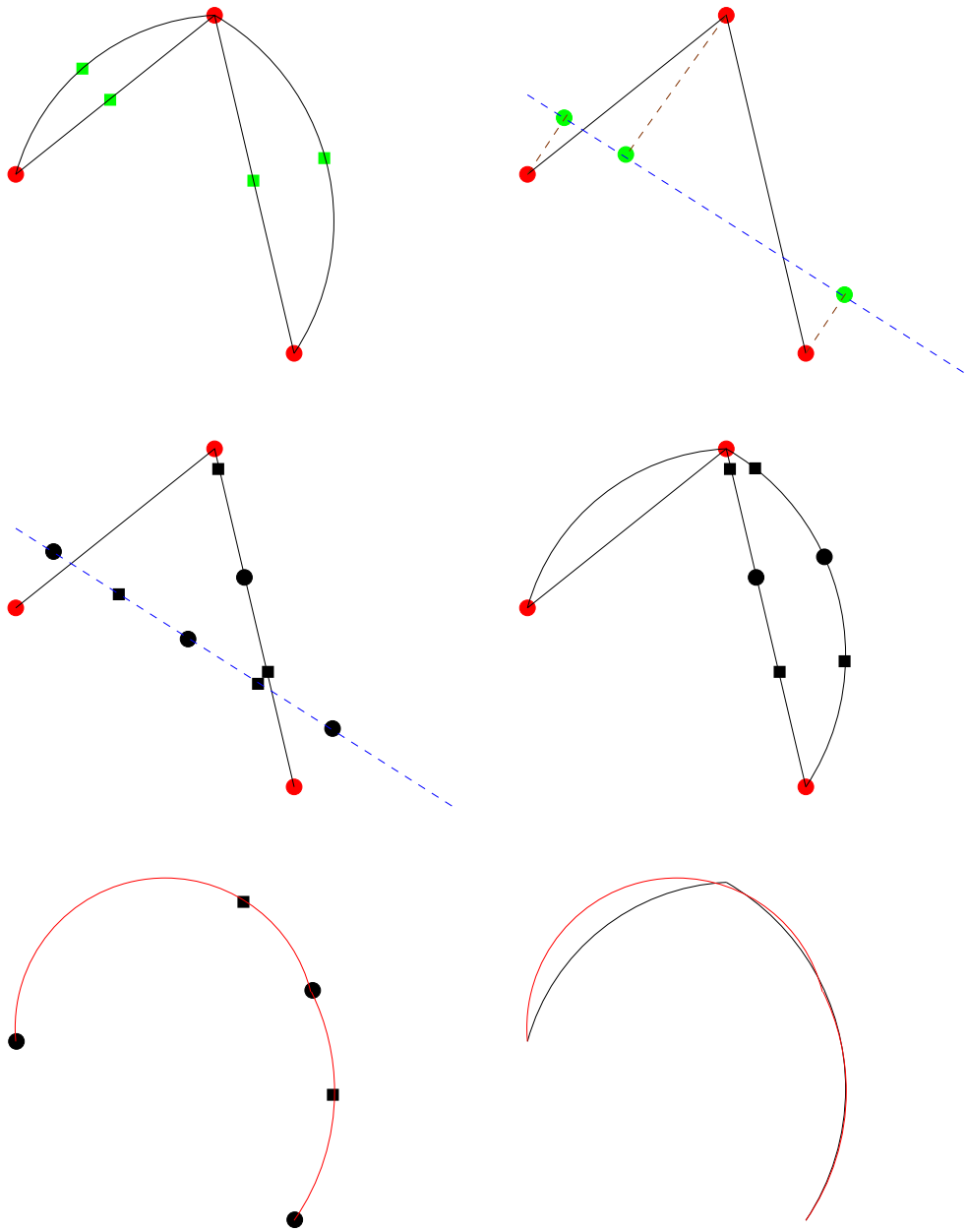


Figure 7.16: This Figure shows what happens to the quadratic star if Algorithm 7.4.3 is applied to the corresponding affine star and the quadratic coordinates are passed for interpolation. The first frame show the quadratic star together with its affine base. The red dot are the linear dofs and the green boxes the quadratic ones. In the second frame we proceed to optimize the linear star as described in Figure 7.15. In the third frame a quadratic function is passed for interpolation, so the interior quadratic dofs have to be considered, these are mapped through the lift to the original affine star. In the forth frame these are map by the quadratic function. The fifth frame shows the newly obtained dofs which determine the new shape of the quadratic star.

7.5 Quadratic Correction

Quadratic isoparametric elements are good for boundary approximation and computation of curvatures. But for moving meshes it is difficult to control their behavior to avoid crossing and mesh distortion (see Figure 7.17). In this work a hybrid affine-

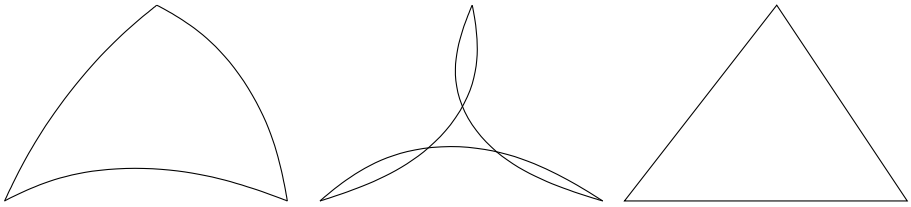


Figure 7.17: The picture shows two quadratic isoparametric elements with the same affine support shown on the right. The first one is a good element whereas in the second nodes are crossing.

quadratic approach to the surface/boundary isoparametric elements is proposed. The idea is to keep the quadratic element not far from its affine support, but still allow it to have the characteristic rounded shape coming from the quadratic bubble. Recalling the notation of Section 5.6 (summarized in Figure 7.18) let K be the quadratic isoparametric element and $\tilde{K}$ its affine base or support. Let $\hat{\boldsymbol{x}}_i$, $\tilde{\boldsymbol{x}}_i$ and $\boldsymbol{x}_i$

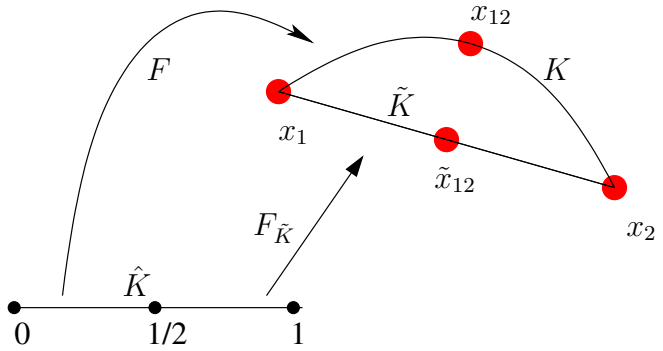


Figure 7.18: Picture to recall the notation of the different objects involved with quadratic isoparametric element and its affine base.

be the vertices of the reference, affine support and quadratic element respectively. We have that $\tilde{\boldsymbol{x}}_i = \boldsymbol{x}_i$. Let $\hat{\boldsymbol{x}}_{ij}$ and $\tilde{\boldsymbol{x}}_{ij}$ be the midpoint between $\hat{\boldsymbol{x}}_i$ and $\hat{\boldsymbol{x}}_j$, and $\boldsymbol{x}_{ij} = F(\hat{\boldsymbol{x}}_{ij})$.

The measure of how close K is to $\tilde{K}$ is based on the result of Theorem 5.6.1. To make the concept computationally precise, we define a number $\theta \in (0, 1)$ to be the *closeness threshold*. Define the ij-th edge length $e_{ij} := |\boldsymbol{x}_i - \boldsymbol{x}_j|$ and the ij-th discrepancy $d_{ij} := |\boldsymbol{x}_{ij} - \tilde{\boldsymbol{x}}_{ij}|$. Given θ we say that the quadratic mesh $\mathcal{T}$ is *under the control* of its affine base if

$$d_{ij} \leq \theta e_{ij}^2 \quad \forall \boldsymbol{x}_i, \boldsymbol{x}_j \in K \quad \forall K \in \mathcal{T}.$$

For d_{ij} to be a sharp measure of the discrepancy we require that the condition

$$(\boldsymbol{x}_{ij} - \tilde{\boldsymbol{x}}_{ij}) \perp (\boldsymbol{x}_i - \boldsymbol{x}_j) \tag{7.20}$$

is satisfied (see Remark 7.5.2).

Lemma 7.5.1 (Affine Control)**.** *Let the closeness threshold θ and a quadratic isoparametric mesh be given. Then by local refinement it is always possible to put it under the control of its affine mesh.*

Proof. When doing refinement of the mesh the new nodes are projected to the quadratic parent element. For the affine mesh it is equivalent to doing interpolation of a quadratic function with piecewise linears. Then from interpolation results it follows that the pointwise error is $O(h^2)$. □

When working with a quadratic mesh, after any motion we apply a quadratic correction. The motion can be either because the mesh is advanced with a given ve-

locity and time-step or because a smoothing is performed. The quadratic correction consist in:

- correction of the midnode to satisfy (7.20),
- computation of d_{ij} and e_{ij},
- refinement until $d_{ij} \leq \theta e_{ij}^2$.

Having this quadratic correction applied assures that the quality of the affine support controls the quality of the quadratic element. Then the affine techniques for mesh improvement of Section 7.4 can be used on quadratic meshes. Also as the quality of the affine element is used for time-step adaptivity in Section 7.6, the quadratic correction will allow us to apply the time-step control to quadratic meshes.

In the following Subsubsections we explain how the correction of the midnode to satisfy (7.20) is attained. First we treat the simpler case of isoparametric quadratic elements of dimension one in $\mathbb{R}^2$ and then we proceed to an element of dimension two in $\mathbb{R}^3$.

7.5.1 Quadratic Correction: One Dimensional Element

The key observation is that given three points in $\mathbb{R}^2$ there is a unique parabola through them. A quadratic isoparametric element in this setting is a parabola but controlled by three points $\boldsymbol{x}_1$, $\boldsymbol{x}_2$ and $\boldsymbol{x}_{12}$, see Figure 7.19. In order to to satisfy (7.20) we want to have this same parabola controlled by the points $\boldsymbol{x}_1$, $\boldsymbol{x}_2$ and $\boldsymbol{x}_{12}^*$, where $\boldsymbol{x}_{12}^*$ is the intersection of the parabola with the perpendicular bisector of the affine support.

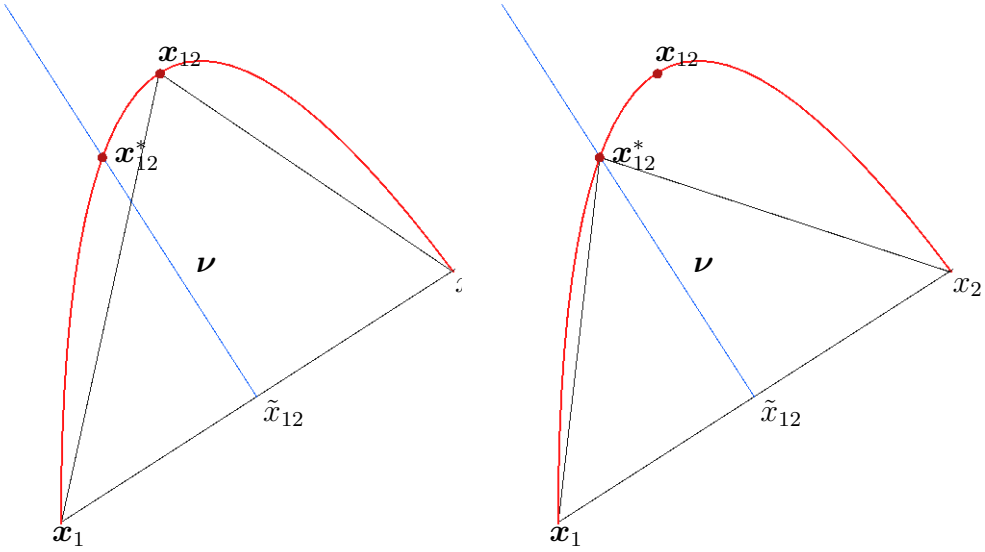


Figure 7.19: A quadratic isoparametric element in this setting is a parabola controlled by the three points $\boldsymbol{x}_1$, $\boldsymbol{x}_2$ and $\boldsymbol{x}_{12}$. In order to to satisfy (7.20) we want to have this same parabola controlled by the points $\boldsymbol{x}_1$, $\boldsymbol{x}_2$ and $\boldsymbol{x}_{12}^*$, where x_{12}^* is the intersection of the parabola with the perpendicular bisector of the affine support, namely the line emanating from the midpoint $\tilde{\boldsymbol{x}}_{12}$ and perpendicular to the segment $\boldsymbol{x}_1 - \boldsymbol{x}_2$. The figure shows the control nodes before and after the correction.

Now we proceed to describe the implementation of a method to find $\boldsymbol{x}_{12}^*$. It is convenient to express the equations using the barycentric coordinates. Let $\boldsymbol{x}(\boldsymbol{\lambda}) := F(\hat{\boldsymbol{x}}(\boldsymbol{\lambda}))$ where $\hat{\boldsymbol{x}}(\boldsymbol{\lambda})$ is the transformation between barycentric coordinates and the master element. Let K be the parabola and $\tilde{K}$ its affine support (the line segment connecting $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$. The first step is to find the unit normal $\boldsymbol{\nu}$ to $\tilde{K}$. Then find $\boldsymbol{\lambda} = (\lambda_1, \lambda_2) \in \mathbb{R}^2$ and $\alpha \in \mathbb{R}$ such that

$$\begin{aligned} \lambda_1 + \lambda_2 = 1, \quad \lambda_i \geq 0, \\ (\boldsymbol{x}(\boldsymbol{\lambda}) - \tilde{\boldsymbol{x}}_{12}) - \alpha \boldsymbol{\nu} = \boldsymbol{0}. \end{aligned} \tag{7.21}$$

The root $(\boldsymbol{\lambda}^*, \alpha^*)$ of system (7.21) yields the desire position $\boldsymbol{x}_{12}^* = \boldsymbol{x}(\boldsymbol{\lambda}^*)$ and α gives the quantity d_{12} (i.e. the measure of how far the quadratic element is from its affine

support).

Remark 7.5.1 (Newton Method Initial Guess). As a Newton method is used to solve system (7.21) a reasonable initial guess is to take $\boldsymbol{\lambda} = (0.5, 0.5)$ and $\alpha = \theta h^2$, where θ is the closeness threshold of the quadratic mesh.

Remark 7.5.2 (Perpendicularity Condition). The previous discussion shows that given a one dimensional parametric quadratic mesh in $\mathbb{R}^2$ it is always possible to move the midnodes to satisfy equation (7.20) without changing the shape of the mesh.

7.5.2 Quadratic Correction: Two Dimensional Element

There are two differences between the one dimensional case and the two dimensional one. The first one is that the quadratic midnode is not anymore the sole position of one element but now it is shared by two. And the other one is the there is an additional degree of freedom for the parabolic side (see Figure 7.20). Now in the quadratic correction we want to exploit this extra degree of freedom to reallocate the the quadratic midnode. So on top of requiring that the perpendicularity condition (7.20) is satisfied; we require that $\boldsymbol{x}_{ij} - \tilde{\boldsymbol{x}}_{ij}$ is perpendicular to the element. One problem that appears is that $\boldsymbol{x}_{ij}$ is shared by to different elements K_1 and K_2 that may have different normals. The solution is to take a weighted average normal $\boldsymbol{\nu}$ as in Section 7.2.1. Then the corrected node $\boldsymbol{x}^*_{ij} \in K_1 \cap K_2$ is the solution to

$$\boldsymbol{x}^*_{ij} = \tilde{\boldsymbol{x}}_{ij} + \alpha\boldsymbol{\nu} \tag{7.22}$$

$$(\boldsymbol{x}^*_{ij} - \tilde{\boldsymbol{x}}_{ij}) \cdot (\boldsymbol{x}_j - \boldsymbol{x}_i). \tag{7.23}$$

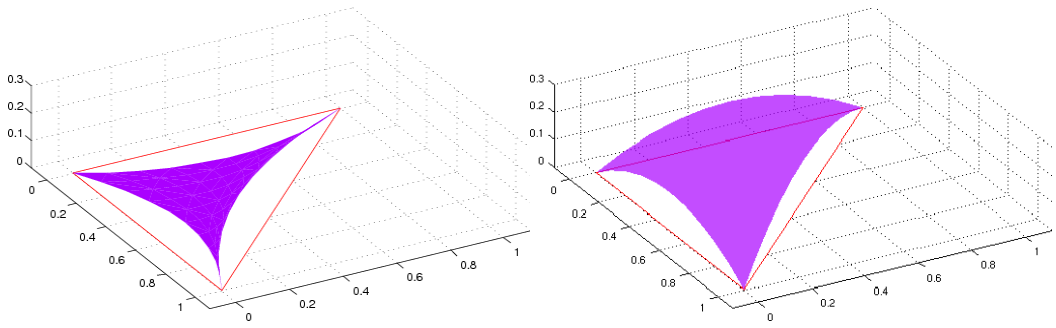


Figure 7.20: These pictures show two surface quadratic isoparametric elements in $\mathbb{R}^3$. Both of them have the same affine base (the red triangle) and the same discrepancy measures d_{ij}. The difference is that in the picture on the left the discrepancy is in a tangential direction to the affine base, and in the picture on right it is in the normal direction to the affine base. Observe that the element on the right has a much better quality. This shows that a 2D quadratic isoparametric element in $\mathbb{R}^3$ has an extra degree of freedom coming from the angle between the element normal $\boldsymbol{\nu}$ and $\boldsymbol{x}_{ij} - \tilde{\boldsymbol{x}}_{ij}$.

7.6 Time Adaptivity

In general nonlinear time dependent fourth order problems present a highly varying time scale along its evolution (see for example simulation 8.3.4). Consequently, an adaptive timestep control indispensable. But even more important for a parametric finite element method is the geometric aspect of the timestep control. The timestep control presented here serves two geometric purposes. The first one is to ensure that it is not too big as to produce node crossing. Secondly, it guarantees that if the velocity is too small the timestep is big enough so that it does not take too many timesteps to produce a negligible evolution.

Another timestep control more suited for optimization problems is backtracking. This technique is used in particular when close to a minimizer. It consists of checking, after a time iteration, if the energy has decreased enough, and if not the

timestep is reduced and the iteration repeated.

For a surface gradient flow we use a time step control in the spirit of Algorithm 5.3 in [BMN05] for surface diffusion. One difference is that we make use and take advantage of the element quality in the computation. For fluid-structure problems the previous time stepping does not apply so we present an appropriate version of timestep control. As a matter of facts, first we present the timestep control for fluid-structure problem and then we mention a modification that can be used for surfaces.

The basic idea of the geometric time step control is that given a velocity to evolve the mesh one wants to find the maximum timestep that will maintain the mesh at reasonable quality and avoid node crossing.

First we show a lemma that justify the timestep control through the use of the element quality. Any reasonable quality metric q_K will be equivalent to $\frac{\rho_K}{h_k}$. This is the case for all quality metrics mentioned in Section 7.4).

Lemma 7.6.1 (Quality for Perturbed Simplex)**.** *Let K be a non-degenerate simplex, $\{\boldsymbol{x}_i\}^n$ its vertices. Given $\alpha < 1$ there is $C = C(\alpha) > 0$ such that if:*

- $B_i := B(\boldsymbol{x}_i, Ch_K q_K)$,
- *we choose* $\boldsymbol{x}_i^* \in B_i$,
- *let* $K^* = span\{\boldsymbol{x}_i^*\}$,

then $q_{K^*} \geq \alpha q_K$.

Proof. Without loss of generality we can assume that $q_k = \frac{\rho_K}{h_k}$. Let

$$C = \frac{1-\alpha}{\alpha\rho_k + 1}.$$

Then $d := Ch_K q_K = \frac{\rho_k(1-\alpha)}{\alpha\rho_k+1}$. Using the definition of convex hull it can be shown that the ball $B(\boldsymbol{y}, \rho_k - d) \subset K^*$. Also $h_{K^*} \leq h_K + d$ then

$$q_{K^*} = \frac{\rho_{K^*}}{h_{K^*}} \geq \frac{\rho_k - d}{h_K + d} = \alpha\frac{\rho_K}{h_k},$$

which is the assertion. □

Based on Lemma 7.6.1, given a mesh and a velocity Algorithm 7.6.1 presents a geometric timestep selection.

The mesh is given by $(\mathcal{N}, \mathcal{T})$ and $\boldsymbol{V}$ is the nodal velocity. A parameter ϑ to determine how aggressive to be with the timestep selection is also provided. For each $K \in \mathcal{T}$, the quantity $h_K q_K$ is a measure of how far a vertex of K can be moved in any direction without entangling K. The nodal function $d(N)$ takes the minimum of $h_K q_K$ over all $K \in \mathcal{T}$ that share this N. The quotient $\frac{d(\boldsymbol{N})}{|\boldsymbol{V}(\boldsymbol{N})|}$ gives the maximum time step node N to move without entangling the mesh. Finally, if all the nodes are moved at the same time with the same timestep τ, then such $\tau = \vartheta\rho$ with parameter $\vartheta < 0.5$ avoids no crossing. For linear meshes $theta = 1/3$ is a good choice. For quadratic meshes controlled by the method of Section 7.5, $\theta = 1/6$ is the safe choice.

Remark 7.6.1 (Surface Geometric Timestep Control). For surfaces one can be more aggressive with the timestep. The idea is use the normal component of the velocity. This is obtained replacing line 9 of Algorithm 7.6.1 by $\rho = \min_{\boldsymbol{N}\in\mathcal{N}}\{\frac{d(\boldsymbol{N})}{|v(\boldsymbol{N})|}\}$, where $v(\boldsymbol{N}) = \max\{\boldsymbol{V}(\boldsymbol{N}) \cdot \boldsymbol{\nu}_K : \boldsymbol{N} \in K\}$.

Algorithm 7.6.1 Algorithm for Timestep Selection

procedure SELECT TIMESTEP($\mathcal{T}$, $\mathcal{N}$, $\boldsymbol{V}$, ϑ)
  Compute q_K for each $K \in \mathcal{T}$
  $d(\boldsymbol{N}) = \infty$ for each $\boldsymbol{N} \in \mathcal{N}$
  for $K \in \mathcal{T}$ **do**
    for $\boldsymbol{N} \in (\mathcal{N} \cap K)$ **do**
      $d(\boldsymbol{N}) = \min(d(\boldsymbol{N}), h_K q_K)$
    end for
  end for
  $\rho = \min_{\boldsymbol{N} \in \mathcal{N}} \{ \frac{d(\boldsymbol{N})}{|\boldsymbol{V}(\boldsymbol{N})|} \}$
  $\tau = \vartheta \rho$
end procedure

An illustration on the timestep selection in action is shown in Figure 7.21.

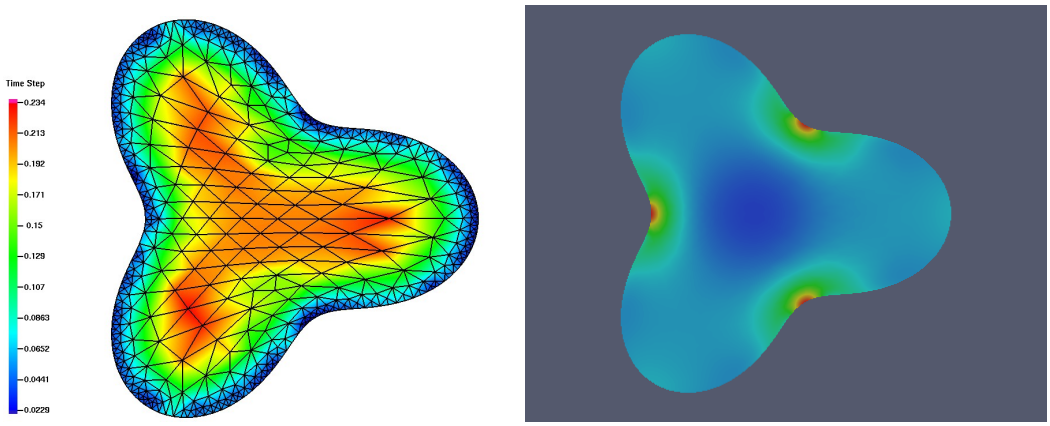


Figure 7.21: The picture on the left shows the time step selected for each node for the given mesh and the given velocity depicted in the picture on the right. The red coloring corresponds to big timesteps the blue to small. The picture on the right is color by the velocity. The timestep assigned to each node depends on the velocity, the size and the quality of the elements.

7.7 Full Algorithms

In this Section we present the general parametric AFEM algorithm with the incorporation of the computational tools previously developed in the chapter. The order in which the tools are applied is important to potentiate themselves in a synergic way. The general order is given in Algorithm 7.7.1.

Algorithm 7.7.1 Full Algorithm

1: **procedure** PARAMETRIC AFEM

2: Start with a good inital mesh and final time T

3: Let $t = 0$ and τ the initial time step

4: **while** $t < T$ **do**

5: ok = false

6: **repeat**

7: Assemble and Solve System

8: $\tau^* =$ time adaptivity $(\boldsymbol{V})$

9: **if** $\tau \geq \tau^*$ **then** ok=true

10: **else** $\tau = \tau^*$

11: **end if**

12: **until** ok

13: Advance Mesh $(\boldsymbol{V}, \tau)$

14: $t+ = \tau$ and $\tau = \tau^*$

15: Enhance the mesh

16: Geometric space Adaptivity

17: **end while**

18: **end procedure**

Chapter 8

Numerical Results

This chapter presents a number of interesting simulations using the methods and tools of chapter 7 to solve the problems discussed in chapter 4 with the schemes of chapter 6. The simulations are meant first to examine the effect of the various computational tools developed. But also they serve to investigate the nonlinear dynamics under large deformations and discover some illuminating similarities and differences with an without fluid.

In this chapter the simulations are divided in four sections. The first two deal with geometric flows (mean curvature flow and Willmore flow with and without constraints). The others with coupling membrane with fluid (capillarity and fluid biomembrane). We end the chapter stating some conclusions drawn from the simulations.

8.1 Software and computers

The software implementation is based on the finite element library `ALBERTA` developed by Schmidt and Siebert. It is based on hierarchical affine grids and employs refinement by bisection. It handles mesh refinement and coarsening, matrix assembling and various quadratures for numerical integration. A printed manual is available [SS05]. Also with the addition by Koster [KKS08] it allows the use of grids

of different dimension in one simulation. Most tools described in Chapter 7 were coded by us in C. They include

- mesh smoothing,
- time adaptivity,
- space adaptivity,
- mesh generation.

The following auxiliary libraries and programs were used:

- The GNU Scientific Library (GSL) (http://www.gnu.org/software/gsl) was used for the optimization routines.
- The GNU library libmatheval http://www.gnu.org/software/libmatheval was used to parse and evaluate symbolic expressions input as text.
- The UMFPACK library http://www.cise.ufl.edu/research/sparse/umfpack/, was use for direct solvers. UMFPACK is a set of routines for solving unsymmetric sparse linear systems using the Unsymmetric Multi Frontal method.
- Paraview http://www.paraview.org was one the the programs used for visualization. Paraview is an open source visualization tool developed by Kitware, Sandia National Laboratories, Los Alamos National Laboratory, Army Research Laboratory and CSimSoft.
- The General Mesh Viewer (GMV) http://www-xdiv.lanl.gov/XCM/gmv was

also used for visualization. GMV was developed at the Los Alamos Laboratory.

- The utility Gnuplot http://www.gnuplot.info/ was used for the plot graphs. Gnuplot is a portable command-line driven interactive data and function plotting utility

Most simulation were run in a PC with an AMD Athlon 64 5000+ processor and 4GB of memory. The computational time for interesting 3D geometric flows was about 30 minutes to an hour. For fluid-membrane interaction it took about 2 days.

The phase field approach has been used to produce quite interesting simulations using a geometric model for biomembranes in the work by Qiang Du et al [DLW04, DLW06]. One advantage of the parametric method over the phase field is the computational cost. In [DLW06] it is reported that for the 3D simulation they used an OPENMP platform on a shared memory system with 16 CPUs (cluster) compare with the low end laptop we can use. On the other hand the a phase field approach allows for topological changes which the parametric one does not.

8.2 Mean Curvature Flow

In this section we present simulations for the mean curvature flow (Problem 4.2.1) with and without enclosed volume constraint.

We start with two curves: the first one is the collapsing circle for which we have an exact formula (see Remark 4.2.1). The other curve is a star shape curve that ends up collapsing to a point with the regime of a circle. Then we show an ellipsoid

and a twisted banana subject to the mean curvature flow with volume constraint.

8.2.1 Collapsing Circle

This simulation shows a circle of initial radius $\sqrt{8}$ evolved under a mean curvature flow. From Remark 4.2.1 we know it is collapsing to a point in finite time. The exact solution given by

$$R(t) = \sqrt{8 - 2t}. \tag{8.1}$$

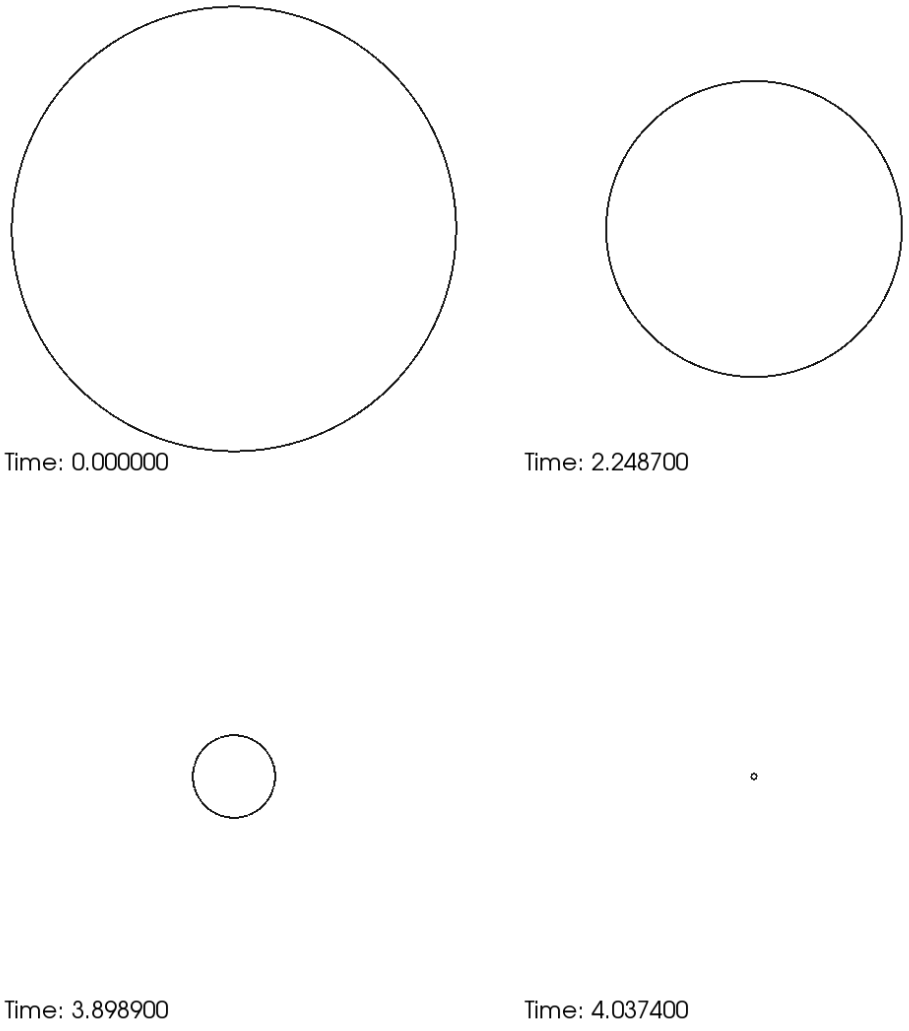


Figure 8.1: Evolution of a circle of initial radius $\sqrt{8}$ subject to mean curvature flow. The evolution is characterize by a symmetric shrinking to a point in finite time with radius given by (8.1). Figure 8.2 shows the evolution of the corresponding perimeter and kinetic energy.

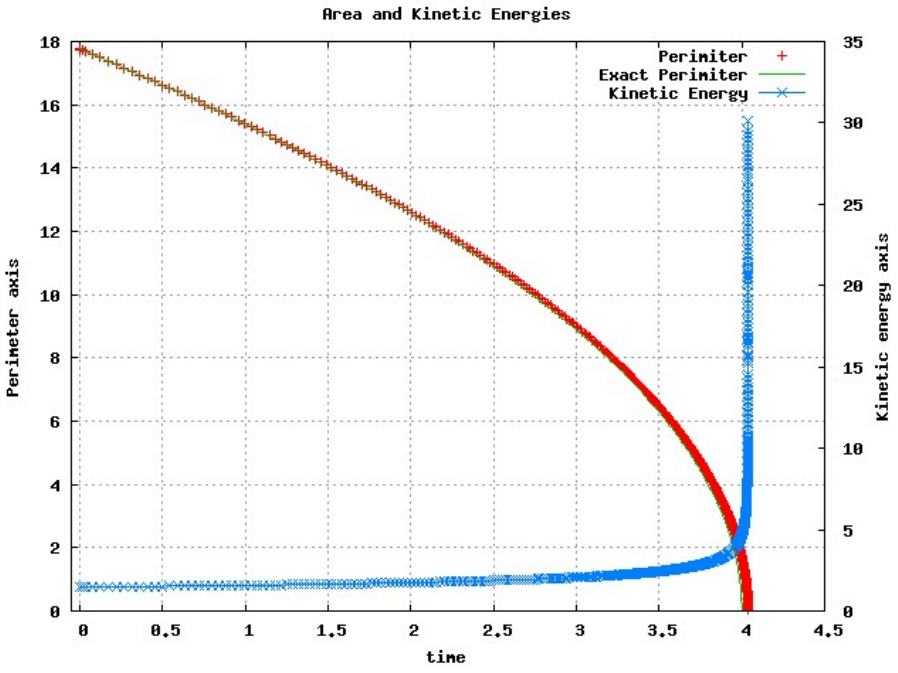


Figure 8.2: The figure shows a combined graph of the perimeter and kinetic energy as function of time corresponding to the simulation of Figure 8.1. Also the exact perimeter computed with (8.1) is plotted. The left axis shows the perimeter scale while the right one does it for the kinetic energy. Notice that the kinetic energy goes to infinity when the shape approaches the point as expected. Also the numerical predicted time for reaching the point is 4.03.

8.2.2 2D Star Shape

This simulation shows another type of curve collapsing to a point. We also take the opportunity to show how space adaptivity follows the shrinking by reducing the number of degrees of freedom accordingly.

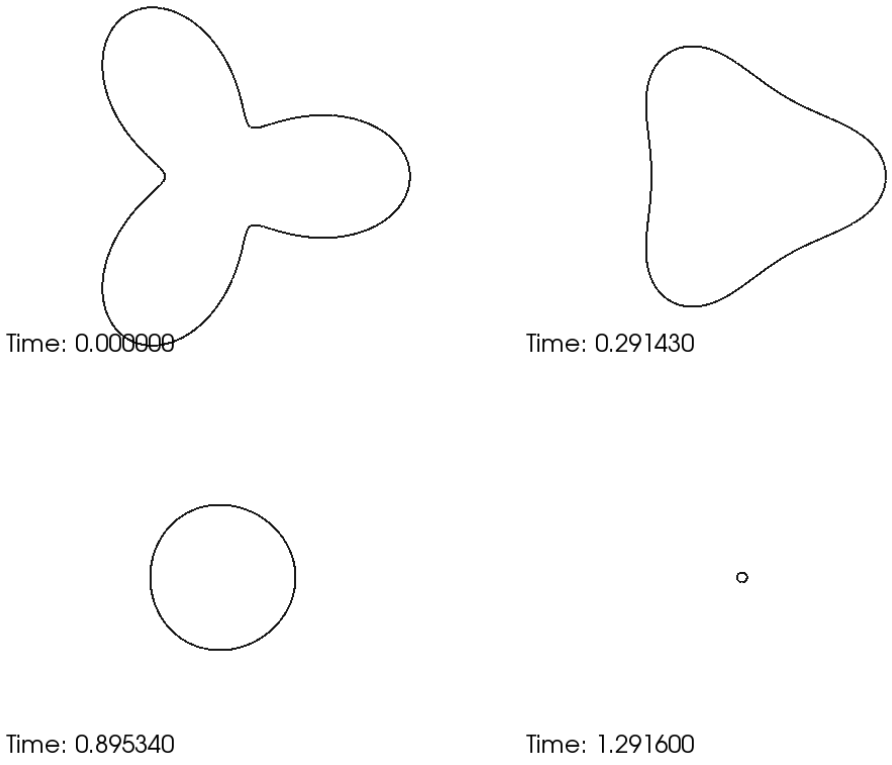


Figure 8.3: Evolution of a star shape curve subject to mean curvature flow. The evolution is characterize first by a stage where the curves moves in and out reducing its curvature until it reaches a circular shape and then it shrinks to a point in finite time, following the regime prescribed in Remark 4.2.1. Figure 8.4 shows the evolution of the corresponding perimeter and kinetic energy. And Figure 8.5 shows the evolution of the of the number of degrees of freedom.

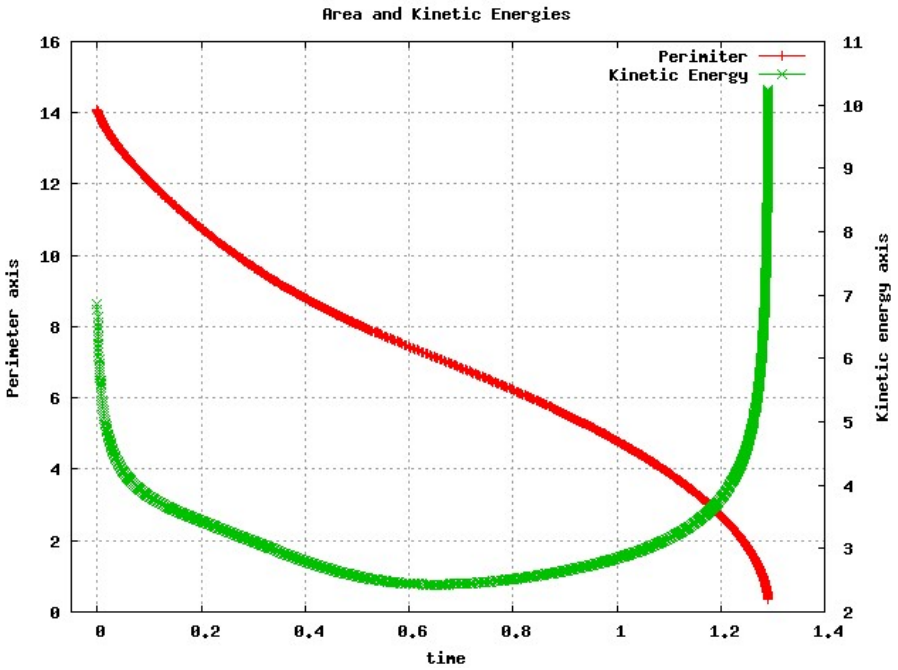


Figure 8.4: The figure shows a combined graph of the perimeter and kinetic energy as function of time corresponding to the simulation of Figure 8.3. The left axis shows the perimeter scale while the right one does it for the kinetic energy. Notice that the kinetic energy goes to infinity when the shape approaches the point.

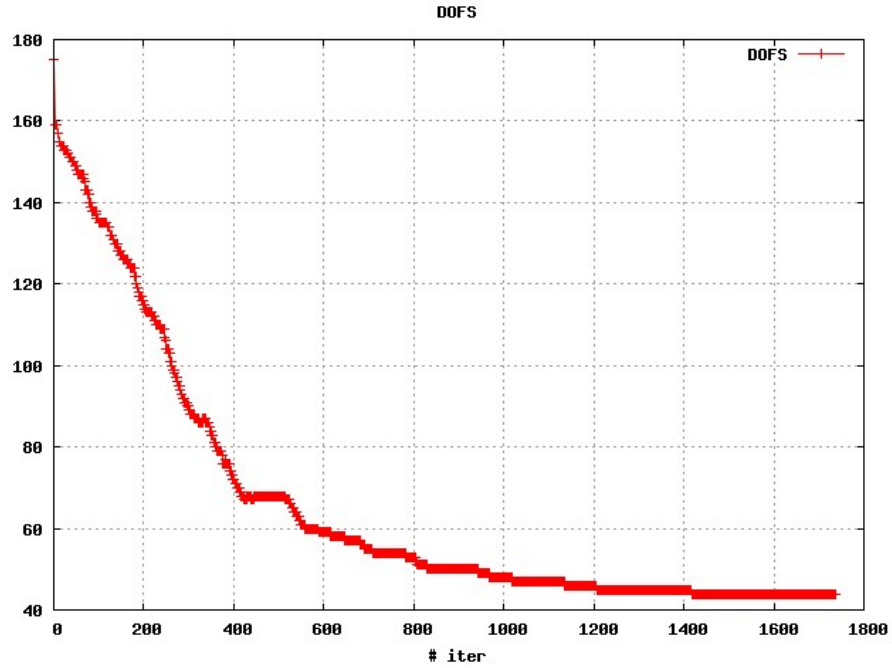


Figure 8.5: The figure shows a graph of the number of degrees of freedom as a function of the number of iterations corresponding to the simulation of Figure 8.3. Notice that as the size of the shape decreases also does the number of degrees of freedom.

8.2.3 Ellipsoid

Evolution of an initial axisymmetric ellipsoid of aspect ratio 2x1x1 subject to a mean curvature flow with volume constraint. As expected it reaches the sphere in finite time.

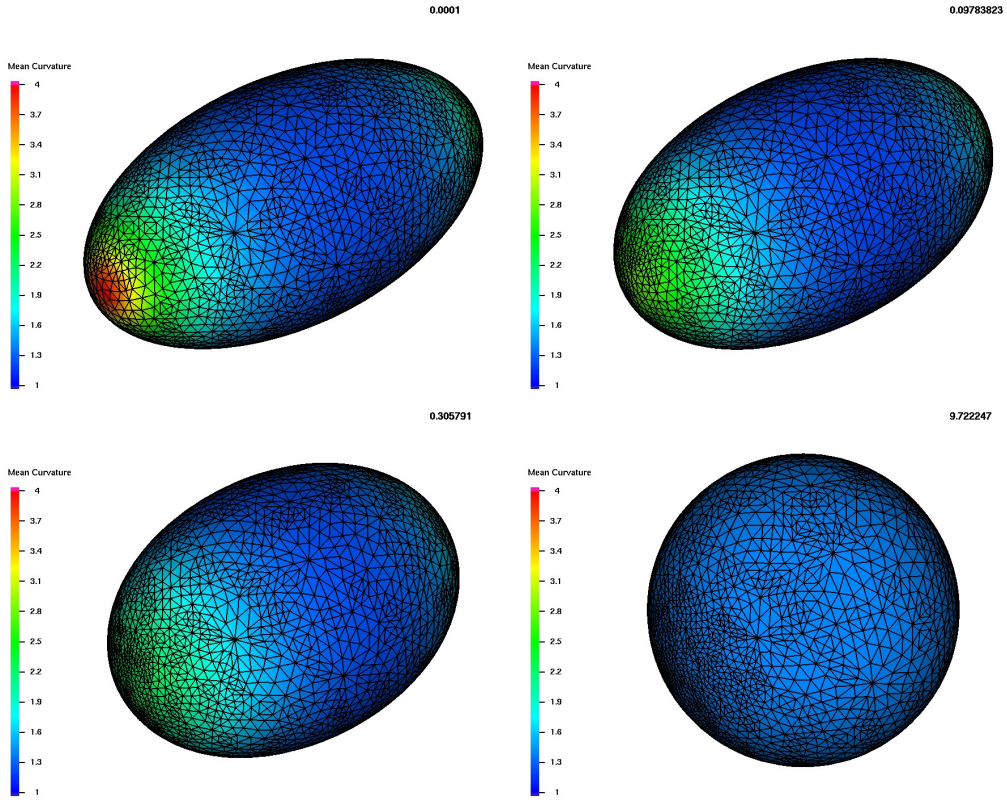


Figure 8.6: Evolution of an initial axisymmetric ellipsoid of aspect ratio 2x1x1 subject to mean curvature flow with volume constraint. The picture shows a 3D view of the surface colored by mean curvature together with the wire-framed mesh. As expected it reaches a sphere with the same initial volume in finite time, and is numerically very stable once there. Figure 8.7 shows the evolution of the corresponding area and kinetic energy. And Figure 8.8 shows the evolution of the area and volume Lagrange multipliers.

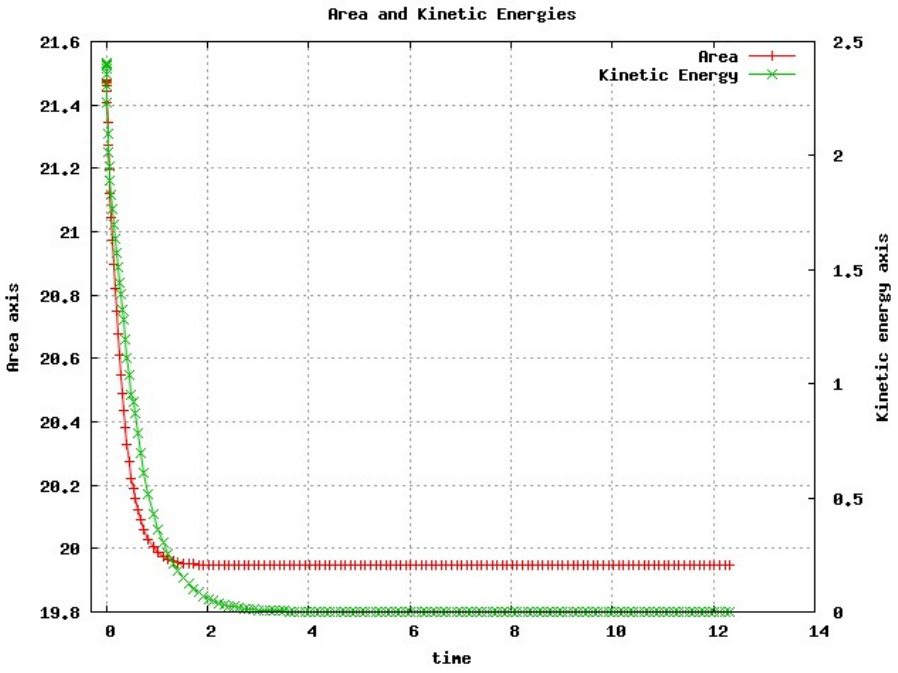


Figure 8.7: The figure shows a combined graph of the area and kinetic energy as function of time corresponding to the simulation of Figure 8.6. The left axis shows the area scale while the right one does it for the kinetic energy. Notice that the kinetic energy becomes zero in finite time when the shape becomes the sphere. And is very stable numerically.

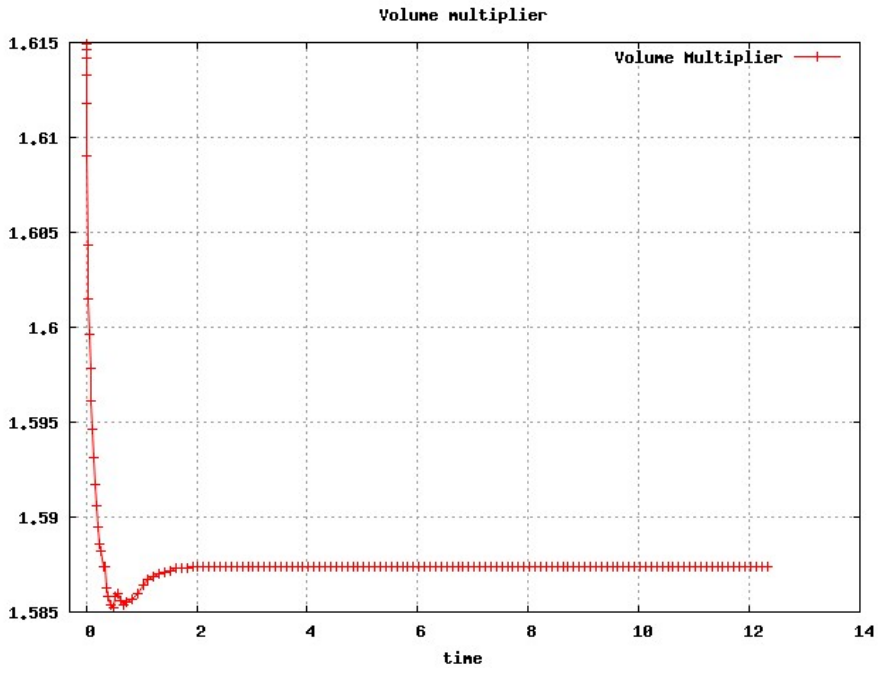


Figure 8.8: The figure shows a graph of the volume Lagrange multiplier as function of time corresponding to the simulation of Figure 8.6.

8.2.4 Twisted banana

This is the twisted banana shape introduce in Section 8.3.4 for its characteristics to check bending energy driven flows (please refer to Section 8.3.4 for more details). This time it is subject to mean curvature flow with volume constraint. We include it in here to show some mesh adaptivity in action.

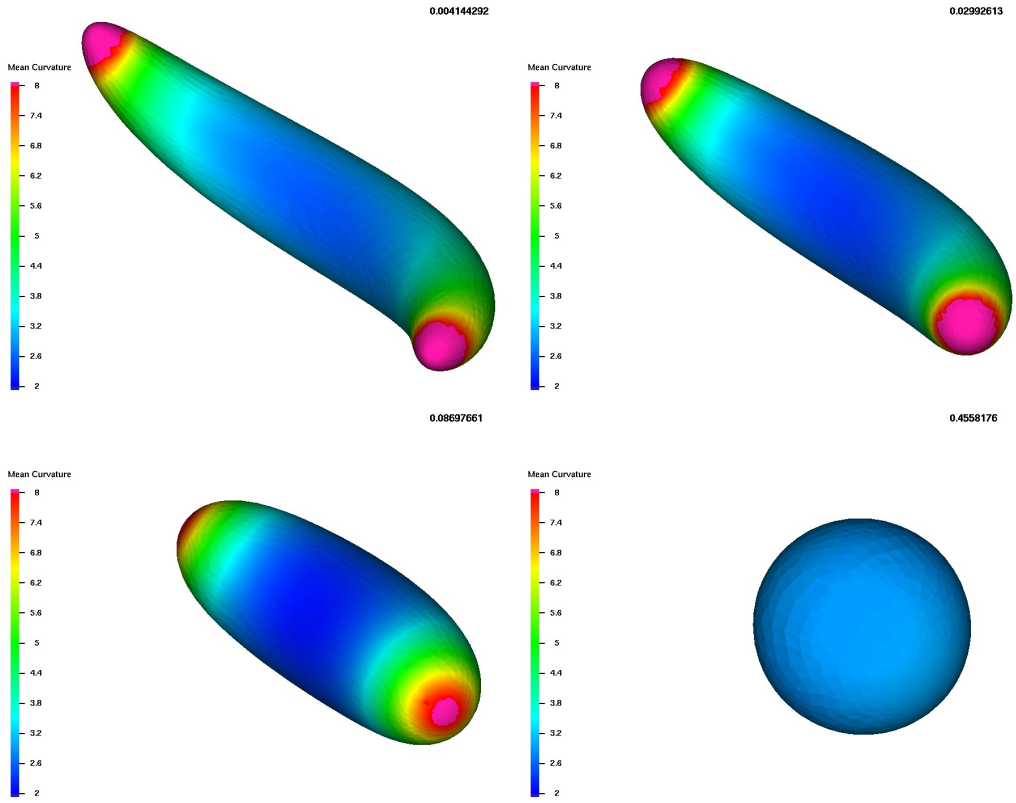


Figure 8.9: Evolution of the initial twisted banana shape subject to mean curvature flow with volume constraint. The initial banana reaches the a ball shape in finite time and stays there. The picture shows a 3D view of the surface colored by mean curvature. Figure 8.11 shows the space adaptivity in action for one end of the shape. Figure 8.10 shows the area and kinetic energy as a function of time.

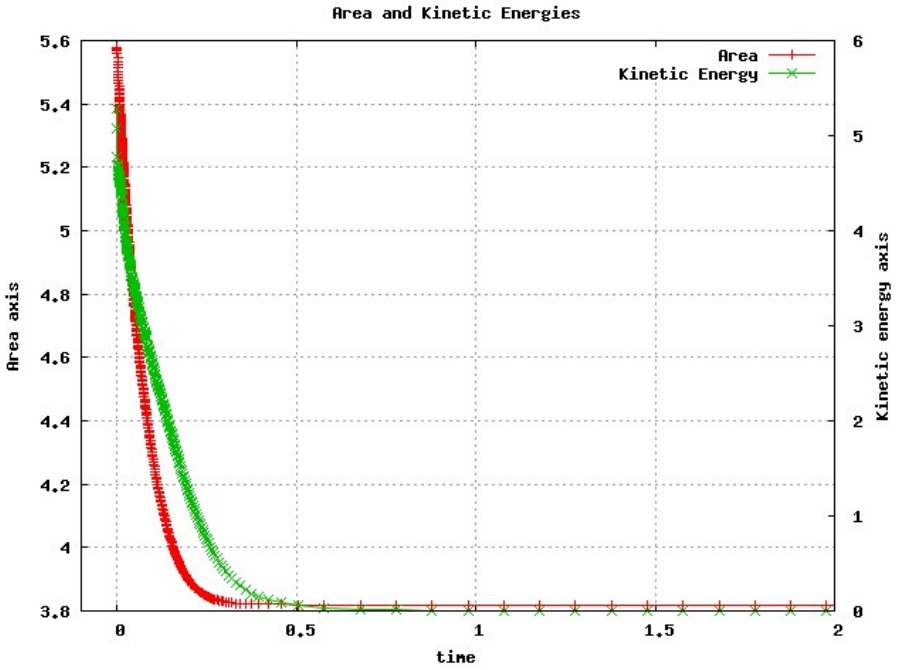


Figure 8.10: The figure shows a combined graph of the area and kinetic energy as function of time corresponding to the simulation of Figure 8.9. The axis on the left has the area scale and the one on the right the kinetic energy scale. Observe that when the area stabilizes (spherical shape) the kinetic energy is 0.

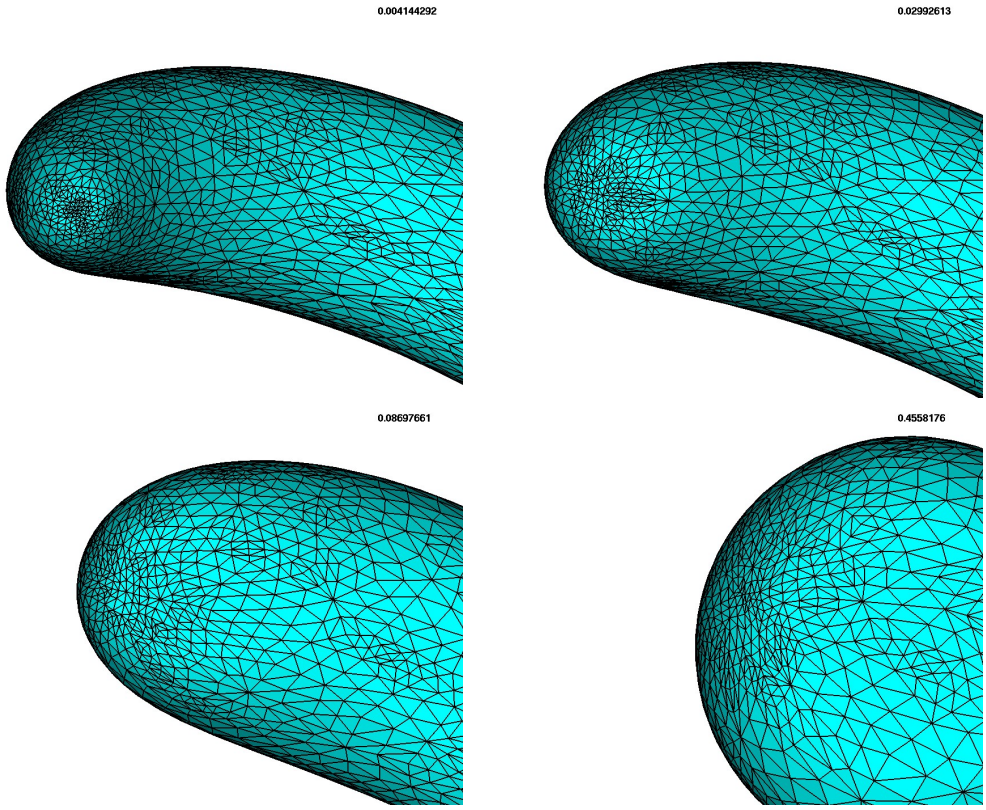


Figure 8.11: The figure shows a zoom of one end of the simulation of Figure 8.9. Each frame shows a 3D view of the zoomed in surface together with the wire-framed mesh. Here we see that at the beginning the regions of high curvature are more densely populated by degrees of freedoms. Throughout time as the banana is evolving degrees of freedom are removed in such a way that at the end they are approximately equidistribute.

8.3 Willmore Flow and Geometric Biomembrane

In this section we present simulations for the Willmore flow (Problem 4.2.5) and the geometric biomembrane model (Problem 4.2.7) which is a Willmore flow with surface area and enclosed volume constraints.

We start with two families of shapes generated from initial axisymmetric ellipsoids. For the Willmore flow they all end in a sphere. For the biomembrane flow we end up with axisymmetric shapes that have been obtained [Jen77b, SBL91] by exploiting the axisymmetry an so reducing the problem to a system of ODEs. We do not take advantage of the axisymmetry and we run the simulations as if they were not axisymmetric. The fact that during the evolution the axisymmetry is preserved is a good indication of the stability of the code.

Given that the ellipsoid has to be axisymmetric it implies that 2 of its 3 axes are equal. Also from Lemma 4.2.4 for a surface under Willmore Flow what actually matters is the aspect ratio of the ellipsoid and not the absolute value of its axes.

The first family of shapes is the one that yields a "dumbbell bar" shape. We discover that it is obtained when the third axis is greater than about twice the length of the other axes.

The second family of shapes is the one that yields a "red blood cell" or "discocyte" shape. It is obtained when the third axis on the initial ellipsoid is less than about half the length of the other axes. In this case pinching is observed when the third axis is 5 times smaller than the others. For higher aspects ratios actual crossing of the upper and lower sides occurs. But given the local nature of the para-

metric method it does not realize about the global crossing. For aspects ratios in between (i.e. from twice to half) a "pill" like shape is obtained. We also include a non axisymmetric ellipsoid. The simulation shows that at equilibrium it ends being axisymmetric. Next we apply the flow to a "twisted banana" shape. This shape is interesting for bending flows because it is not axisymmetric and also has two different bendings in it (the banana bending plus the twist). The timestep adaptivity was crucial for this simulation as we detected two very different time scales along the evolution.

8.3.1 Dumbbell Bar Shapes

This family of shapes is obtained when the third axis of an initial axisymmetric ellipsoid is greater than about twice the length of the other axes. We present three simulations for the aspects ratios of 8x1x1, 4x1x1 and 2x1x1. The last one is not literally in the family but makes the transition phase to the "pill" shaped family through the sphere to the next family of Section 8.3.2.

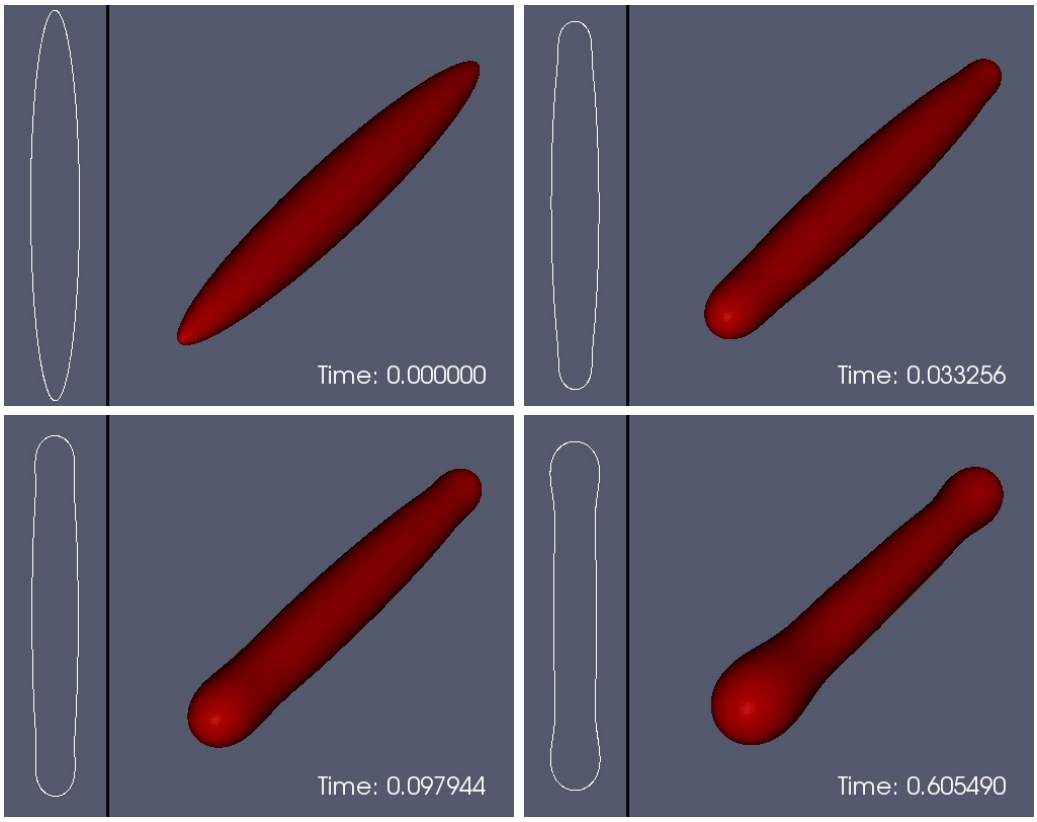


Figure 8.12: Evolution of an initial axisymmetric ellipsoid of aspect ratio 8x1x1 subject to a geometric biomembrane model. For each frame the picture on the right is a 3D view of the surface mesh and the picture on the left a corresponding 2D cut through a symmetry plane. The evolution is characterize by the formation of spherical shaped ends with a strongly cylindrical and long neck connecting them. Figure 8.13 shows the evolution of the corresponding bending and kinetic energy. And Figure 8.14 shows the evolution of the area and volume Lagrange multipliers.

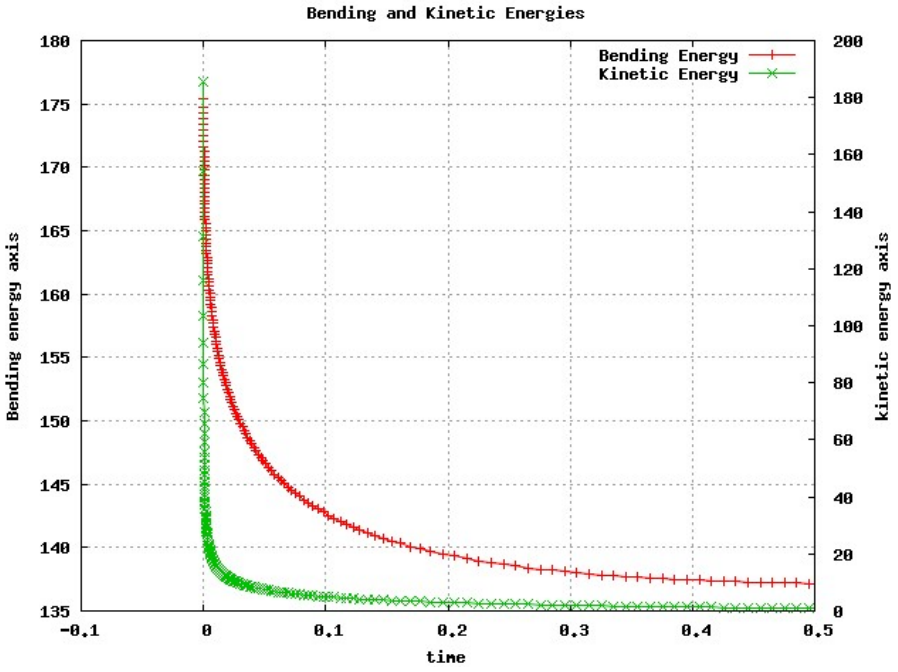


Figure 8.13: The figure shows a combined graph of the bending and kinetic energy as function of time corresponding to the simulation of Figure 8.12. The left axis shows the bending energy scale while the right one does it for the kinetic energy. Notice that the bending energy approaches the equilibrium as the kinetic energy approaches 0. The bending energy for this aspect ratio is reduced approximately 22%.

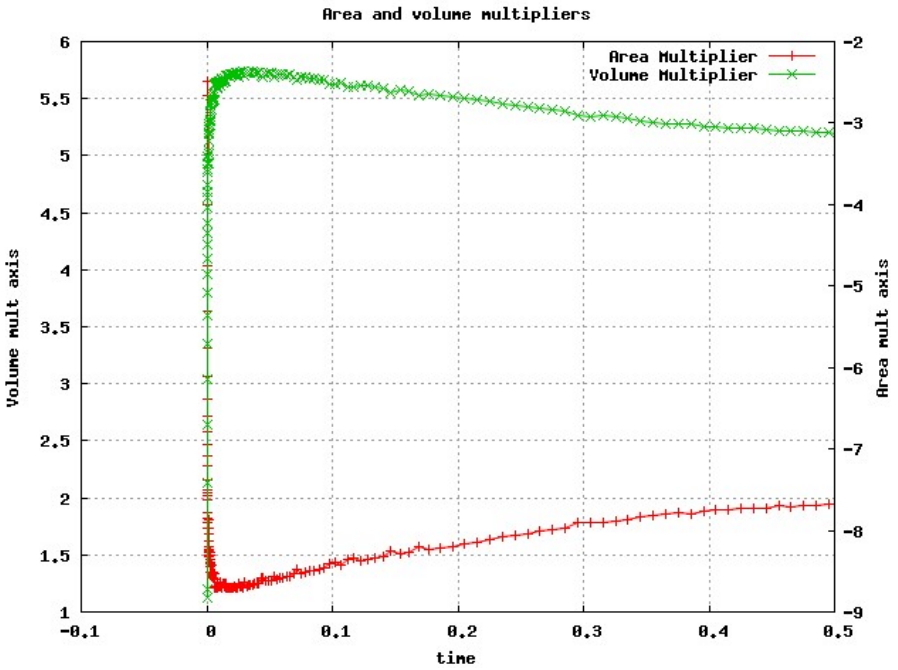


Figure 8.14: The figure shows a combined graph of the area and volume Lagrange multipliers as function of time corresponding to the simulation of Figure 8.12. The left axis shows the area multiplier scale while the right one does it for the volume multiplier.

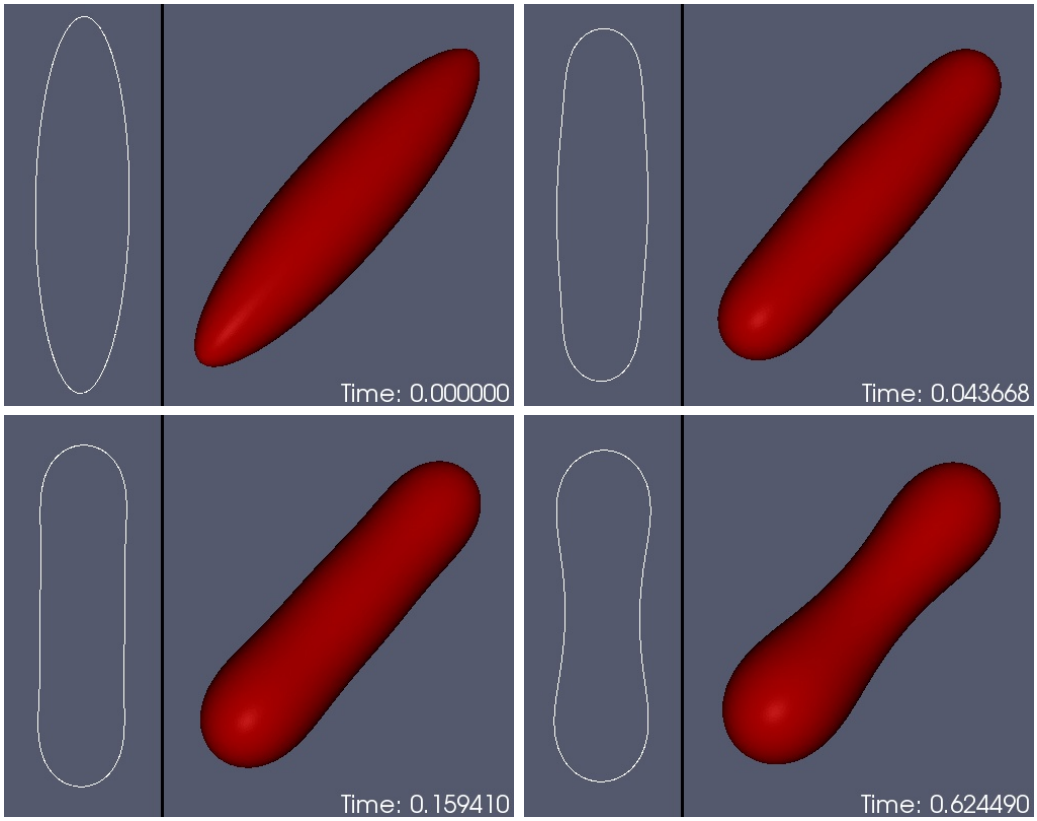


Figure 8.15: Evolution of an initial axisymmetric ellipsoid of aspect ratio 4x1x1 subject to a geometric biomembrane model. For each frame the picture on the right is a 3D view of the surface mesh and the picture on the left a corresponding 2D cut through a symmetry plane. In this evolution the formation of spherical shaped ends are observed but the connection is not so cylindrical and exhibits an indentation in the center (compare with Figure 8.12). Figure 8.16 shows the evolution of the corresponding bending and kinetic energy. And Figure 8.17 shows the evolution of the area and volume Lagrange multipliers.

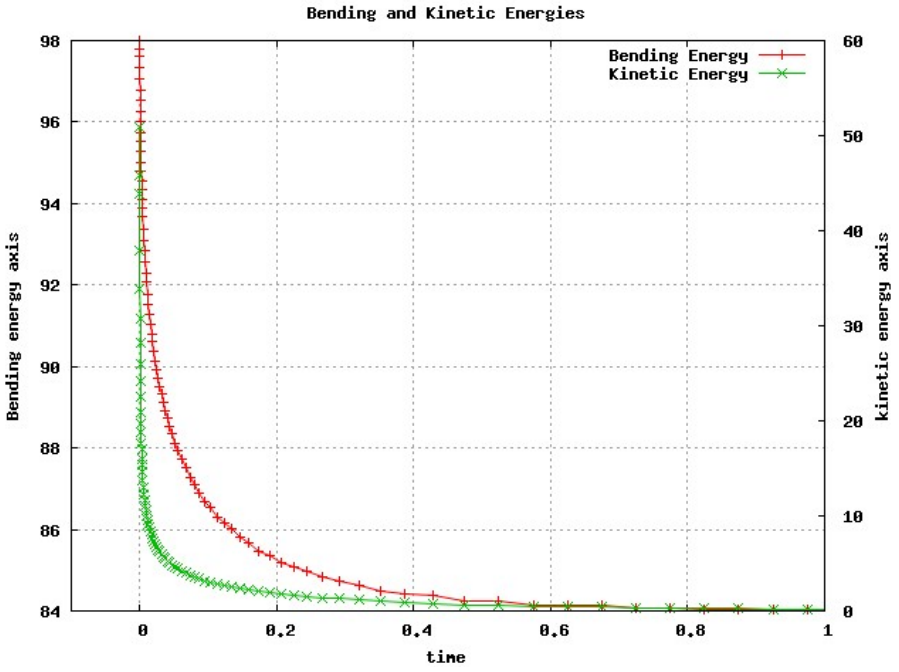


Figure 8.16: The figure shows a combined graph of the bending and kinetic energy as function of time corresponding to the simulation of Figure 8.15. The left axis shows the bending energy scale while the right one does it for the kinetic energy. Notice that the bending energy approaches the equilibrium as the kinetic energy approaches 0. The bending energy for this aspect ratio is reduced approximately 14%.

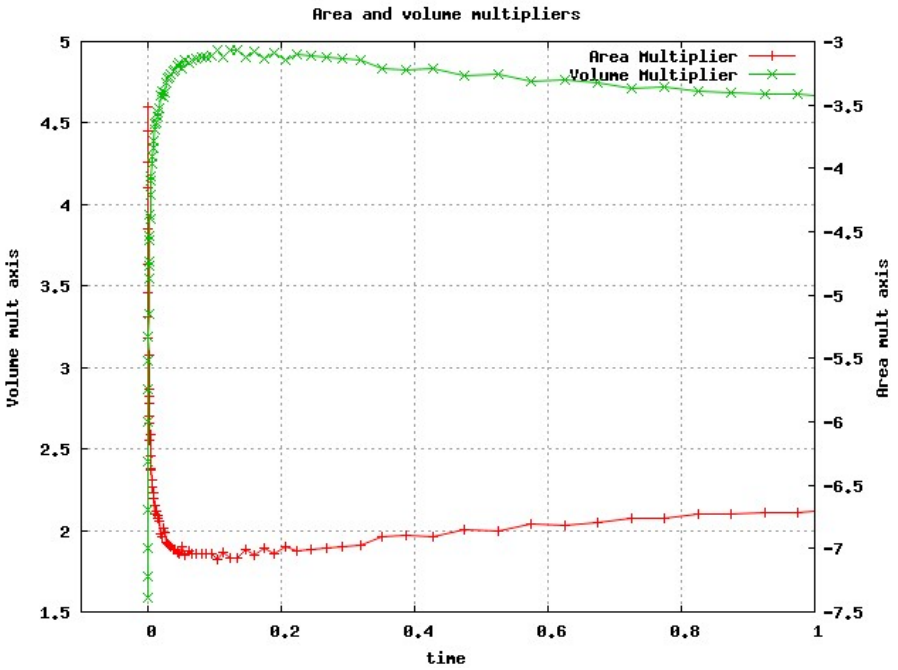


Figure 8.17: The figure shows a combined graph of the area and volume Lagrange multipliers as function of time corresponding to the simulation of Figure 8.15. The left axis shows the area multiplier scale while the right one does it for the volume multiplier.

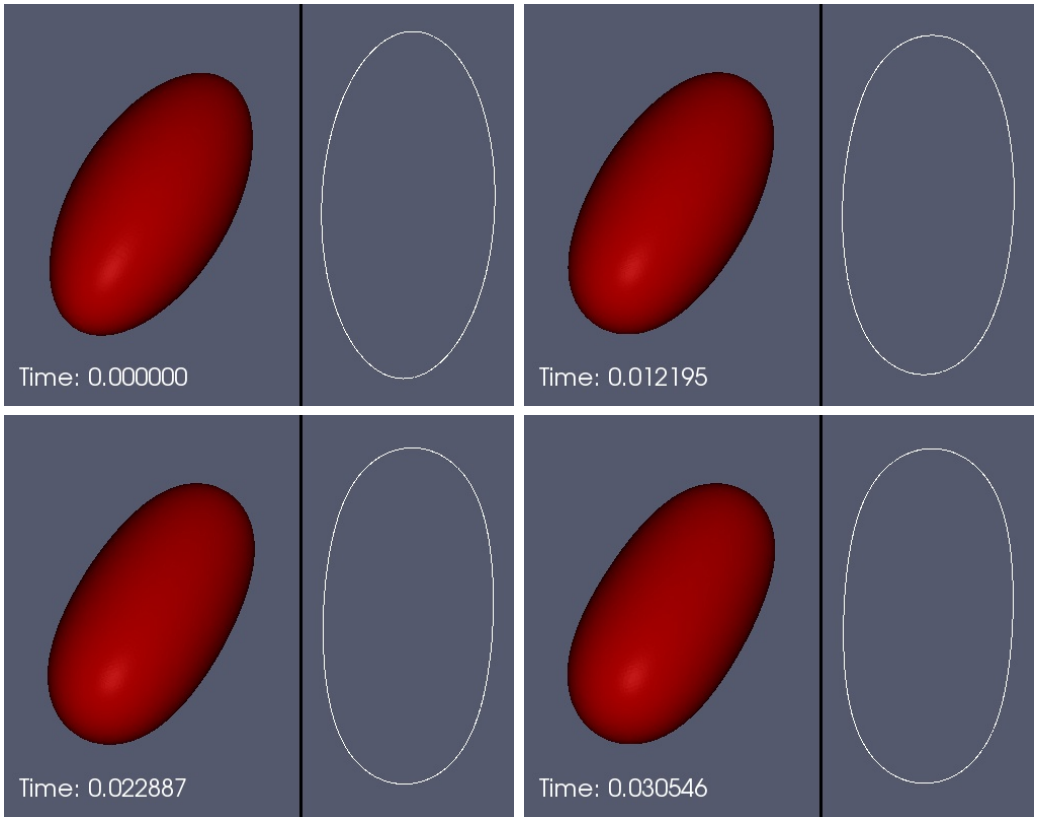


Figure 8.18: Evolution of an initial axisymmetric ellipsoid of aspect ratio 2x1x1 subject to a geometric biomembrane model. For each frame the picture on the left is a 3D view of the surface mesh and the picture on the right a corresponding 2D cut through a symmetry plane. In this evolution instead of the formation of separate spherical shaped ends they prefer to be part of the same "pill" shape. (compare with Figures 8.12 and 8.15). Figure 8.19 shows the evolution of the corresponding bending and kinetic energy. And Figure 8.20 shows the evolution of the area and volume Lagrange multipliers.

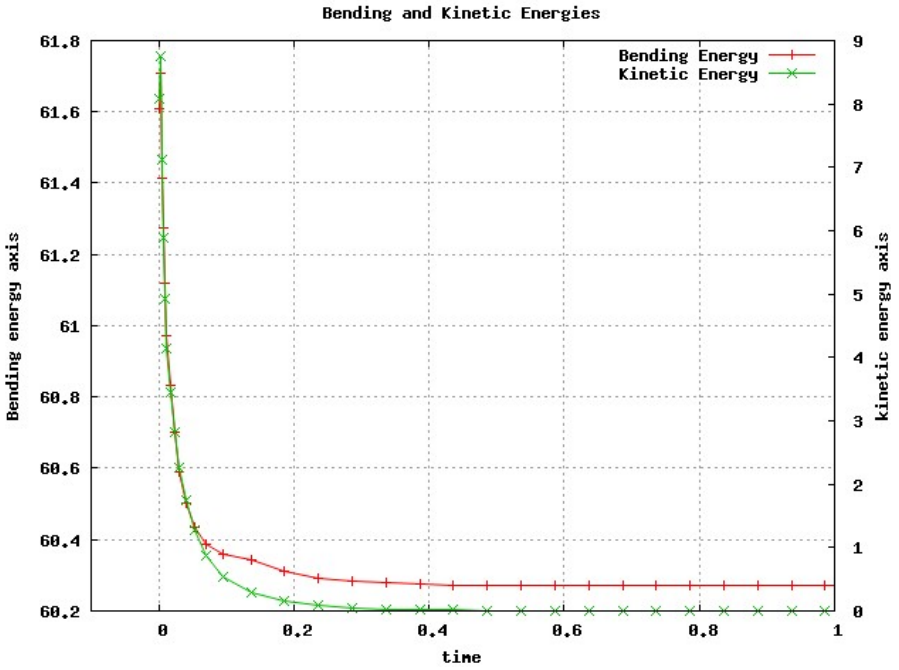


Figure 8.19: The figure shows a combined graph of the bending and kinetic energy as function of time corresponding to the simulation of Figure 8.18. The left axis shows the bending energy scale while the right one does it for the kinetic energy. Notice that the bending energy approaches the equilibrium as the kinetic energy approaches 0. The bending energy for this aspect ratio is reduced approximately 2%.

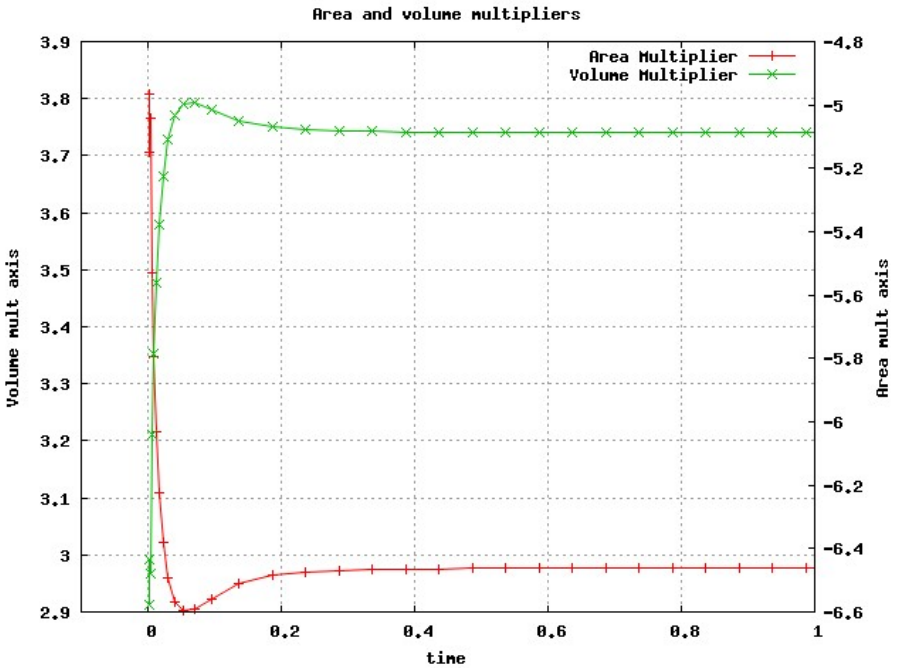


Figure 8.20: The figure shows a combined graph of the area and volume Lagrange multipliers as function of time corresponding to the simulation of Figure 8.18. The left axis shows the area multiplier scale while the right one does it for the volume multiplier.

8.3.2 Red Blood Cell Shapes

This family of shapes is obtained when the third axis on the initial ellipsoid is less than about half the length of the other axes. We present two simulations for the aspects ratios of 3x3x1 and 5x5x1. Pinching is observed when the third axis is 5 or more times smaller than the others. For higher aspects ratios actual crossing of the upper and lower sides occurs. Given the local nature of the parametric method it does not realize about the global crossing. But then the evolution after the crossing is not useful anymore, at least to model a biomembrane. The aspect ratio 5x5x1 yields an equilibrium shape which barely touches the other side.

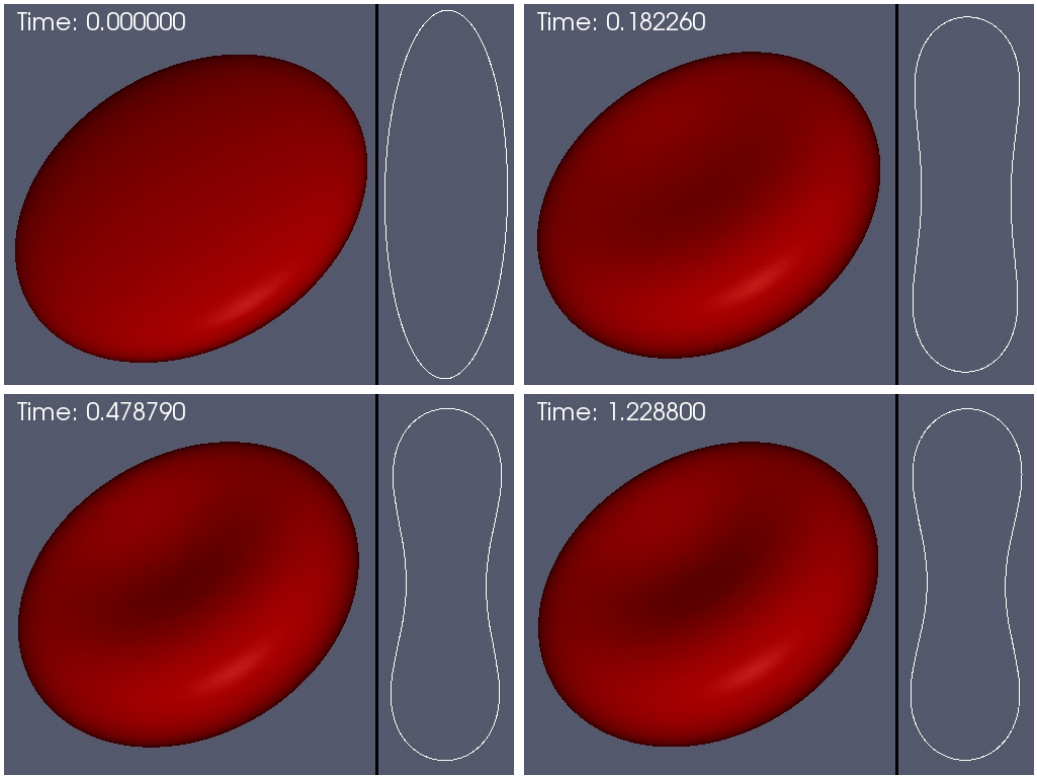


Figure 8.21: Evolution of an initial axisymmetric ellipsoid of aspect ratio 3x1x1 subject to a geometric biomembrane model. For each frame the picture on the left is a 3D view of the surface mesh and the picture on the right a corresponding 2D cut through a symmetry plane. The evolution is characterize by the formation of a depression in the center together with a rounding and thickening of the outer circular edge. Figure 8.22 shows the evolution of the corresponding bending and kinetic energy. And Figure 8.23 shows the evolution of the area and volume Lagrange multipliers.

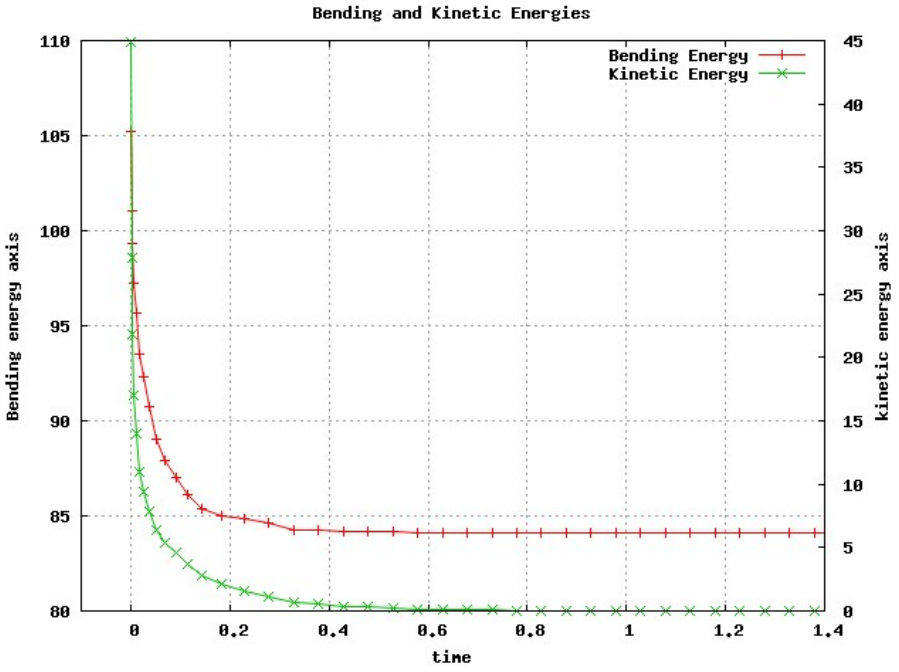


Figure 8.22: The figure shows a combined graph of the bending and kinetic energy as function of time corresponding to the simulation of Figure 8.21. The left axis shows the bending energy scale while the right one does it for the kinetic energy. Notice that the bending energy approaches the equilibrium as the kinetic energy approaches 0. The bending energy for this aspect ratio is reduced approximately 24%.

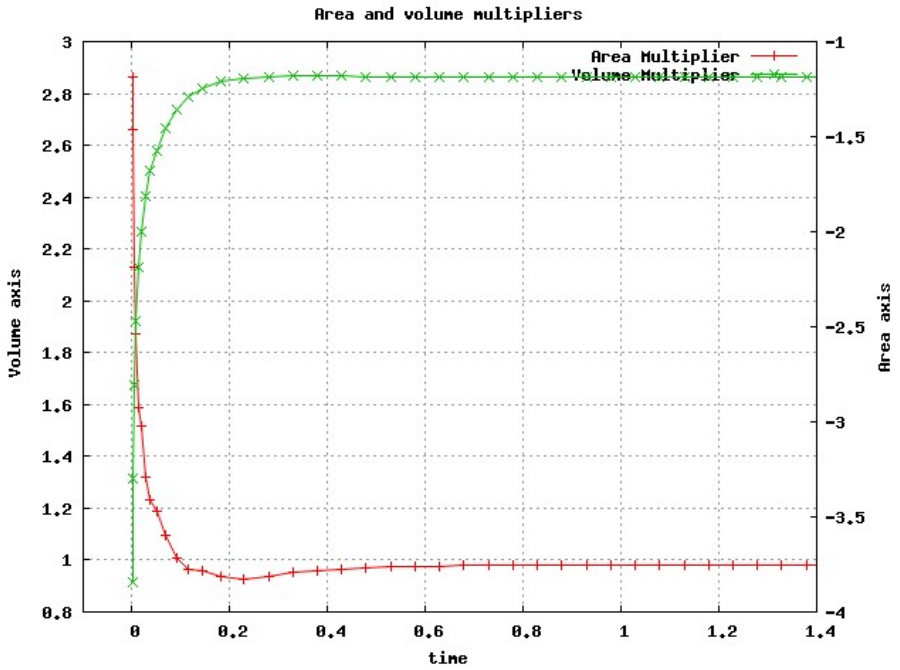


Figure 8.23: The figure shows a combined graph of the area and volume Lagrange multipliers as function of time corresponding to the simulation of Figure 8.21. The left axis shows the area multiplier scale while the right one does it for the volume multiplier.

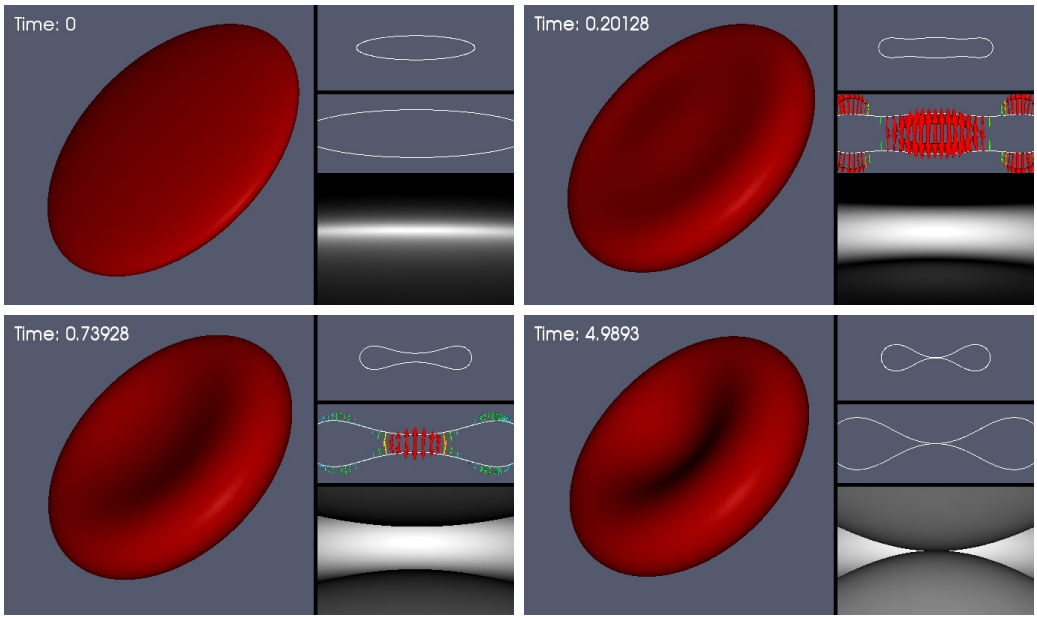


Figure 8.24: Evolution of an initial axisymmetric ellipsoid of aspect ratio 5x5x1 subject to a geometric biomembrane model. For each frame the picture on the left is a 3D view of the surface mesh and the pictures on the right from top to bottom show a 2D cut through a symmetry plane; this cut with the velocity field shown as arrows; and a 3D view from inside the surface. In this case the equilibrium shape presents the formation of a depression of the center more to the point of almost pinching. But the evolution to get there is different from the one of Figure 8.21. As can be seen from the second frame, the thickening of the outer circular edge occurs faster than what happens in the middle, so that instead of a depression a bump is formed in the middle. Latter the evolution continues squeezing this bump to a depression at the expense of thickening and rounding even more the outer circular edge. Figure 8.25 shows the evolution of the corresponding bending and kinetic energy. And Figure 8.26 shows the evolution of the area and volume Lagrange multipliers.

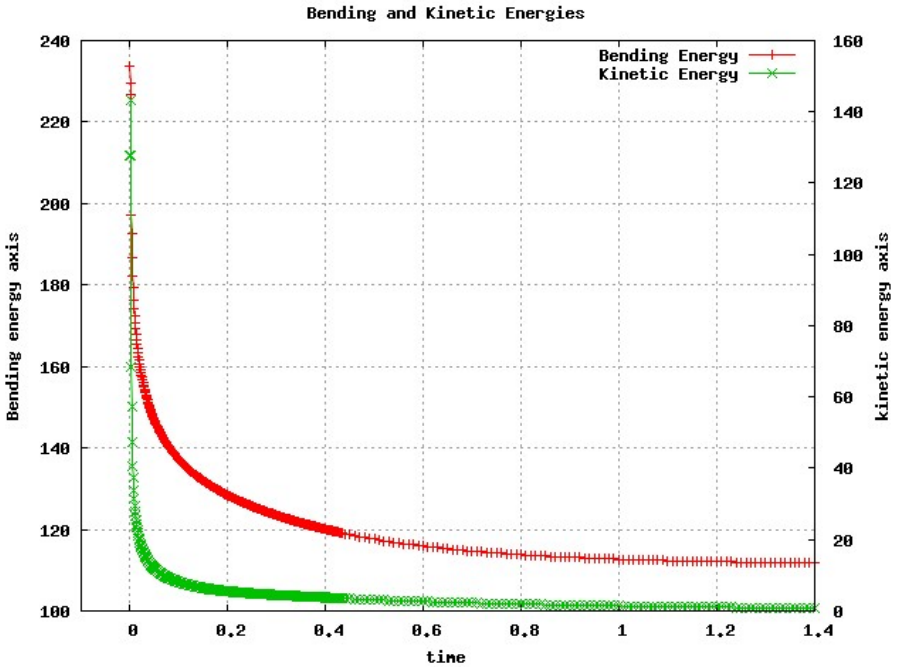


Figure 8.25: The figure shows a combined graph of the bending and kinetic energy as function of time corresponding to the simulation of Figure 8.24. The left axis shows the bending energy scale while the right one does it for the kinetic energy. Notice that the bending energy approaches the equilibrium as the kinetic energy approaches 0. The bending energy for this aspect ratio is reduced approximately 50%.

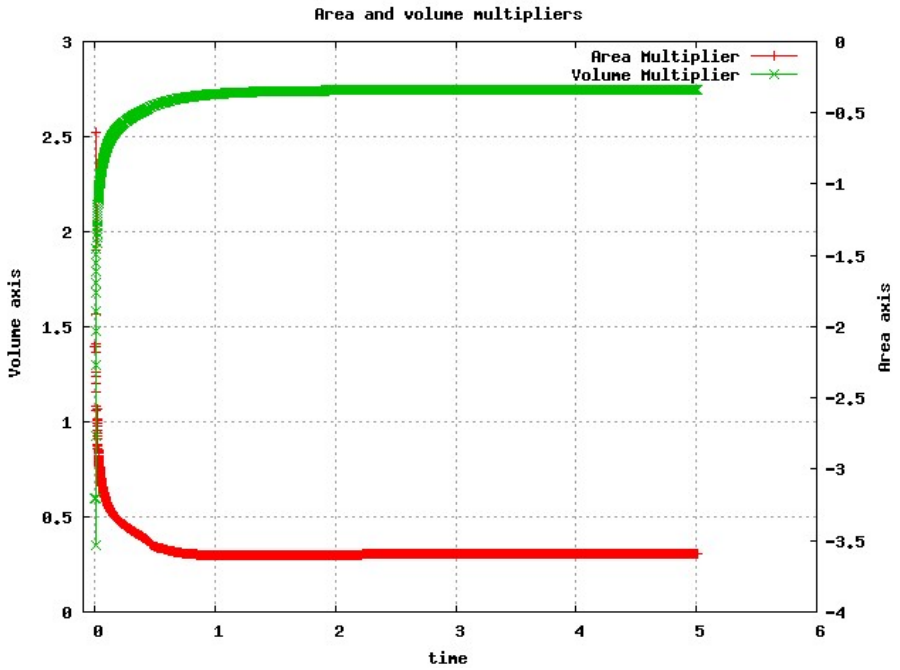


Figure 8.26: The figure shows a combined graph of the area and volume Lagrange multipliers as function of time corresponding to the simulation of Figure 8.24. The left axis shows the area multiplier scale while the right one does it for the volume multiplier.

8.3.3 Non-axisymmetric Ellipsoid

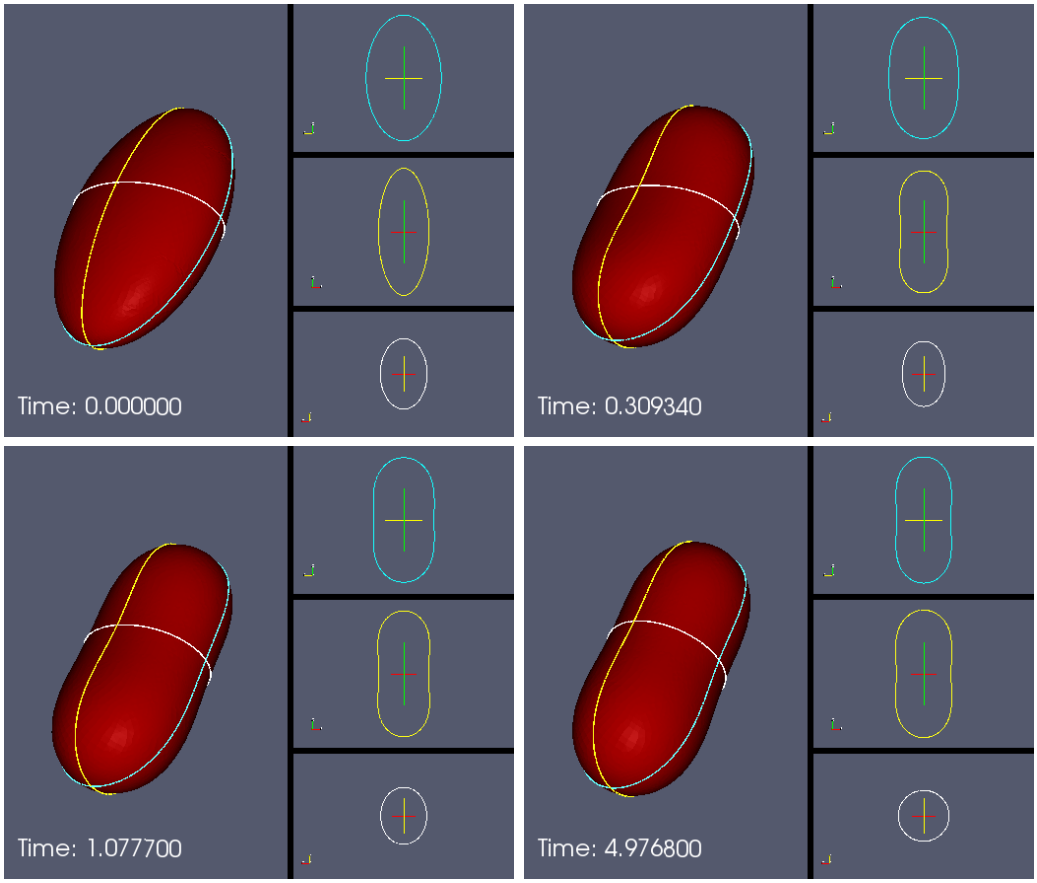


Figure 8.27: Evolution of an initial non-axisymmetric ellipsoid of aspect ratio 2x3x5 subject to a geometric biomembrane model. For each frame the picture on the left is a 3D view of the surface mesh and the picture on the right from top to bottom are three cuts by the coordinate planes. The evolution seems to produce an axisymmetric equilibrium. Figure 8.28 shows the evolution of the corresponding bending and kinetic energy. And Figure 8.29 shows the evolution of the area and volume Lagrange multipliers.

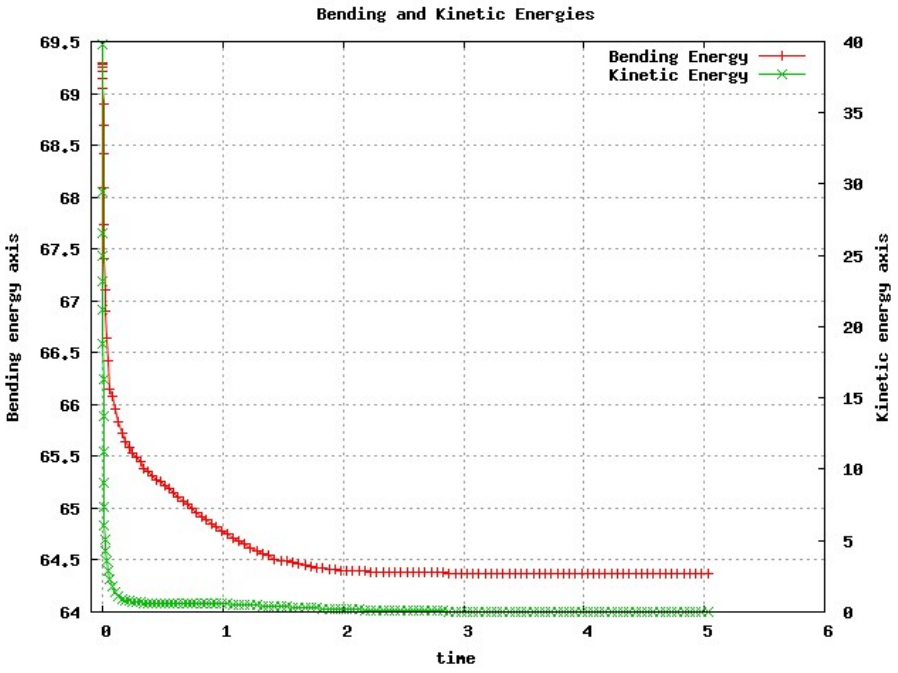


Figure 8.28: The figure shows a combined graph of the bending and kinetic energy as function of time corresponding to the simulation of Figure 8.27. The left axis shows the bending energy scale while the right one does it for the kinetic energy. Notice that the bending energy approaches the equilibrium as the kinetic energy approaches 0.

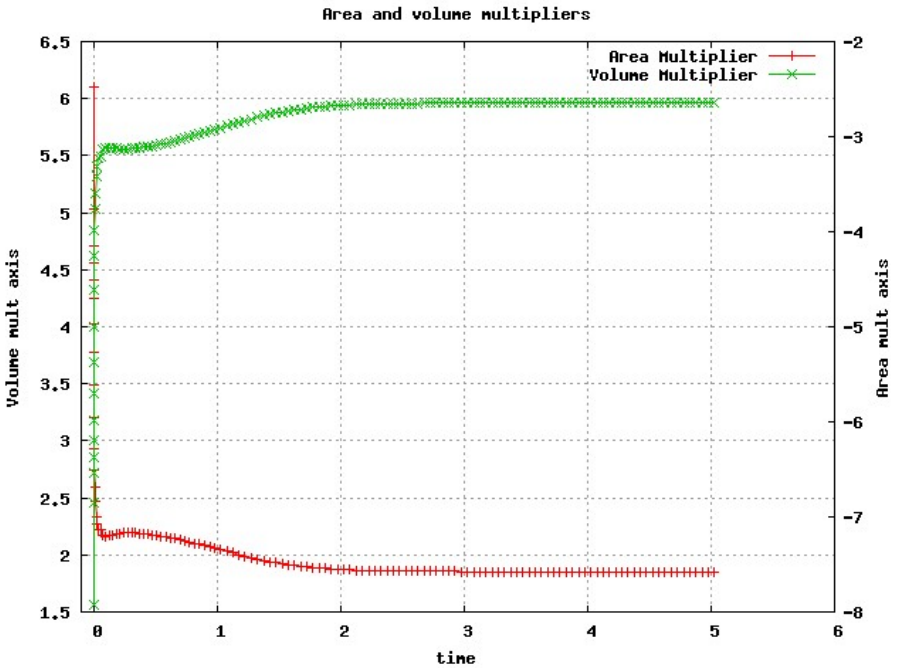


Figure 8.29: The figure shows a combined graph of the area and volume Lagrange multipliers as function of time corresponding to the simulation of Figure 8.27. The left axis shows the area multiplier scale while the right one does it for the volume multiplier.

8.3.4 Twisted Banana

The twisted banana is interesting for bending flows because it is not axisymmetric and also has two different bendings in it (the banana bending plus the twist). The timestep adaptivity was crucial for this simulation as we detected two very different time scales along the evolution. First we introduce the shape and then show the full simulation and the two time scales.

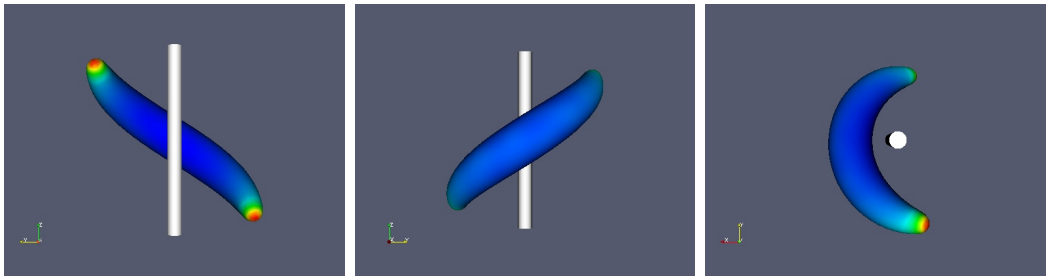


Figure 8.30: Introducing the twisted banana. The picture shows from left to right the front, back an top views of the twisted banana shape. To help visualize the shape it is plotted next to a thin cylinder in the z direction.

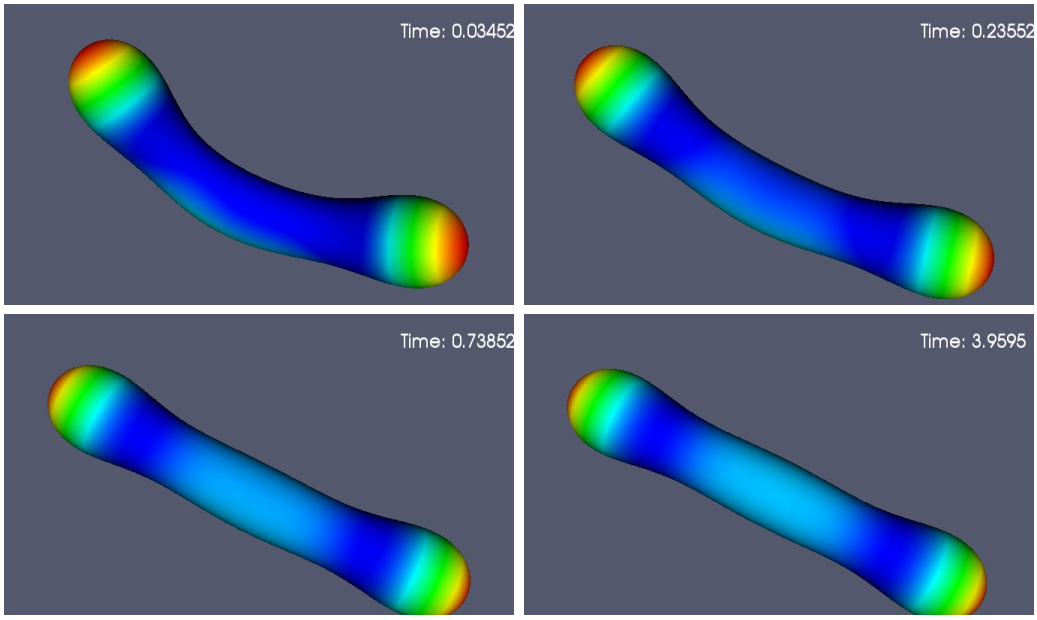


Figure 8.31: Evolution of the initial twisted banana shape of Figure 8.30 subject to a geometric biomembrane model. The frames show a 3D view of the surface mesh colored by bending energy. The simulation was run in the time interval $[0, 5]$. the final equilibrium was reached at about 0.6 this is reflected in the almost similar shape of the last two frames despite the different times. Already in the first frame there has been a lot of change in the shape. In Figures 8.33 and 8.35 we will describe the two time scale and behavior leading to the first and third frame in this figure. Figure 8.32 shows the evolution of the corresponding bending and kinetic energy.

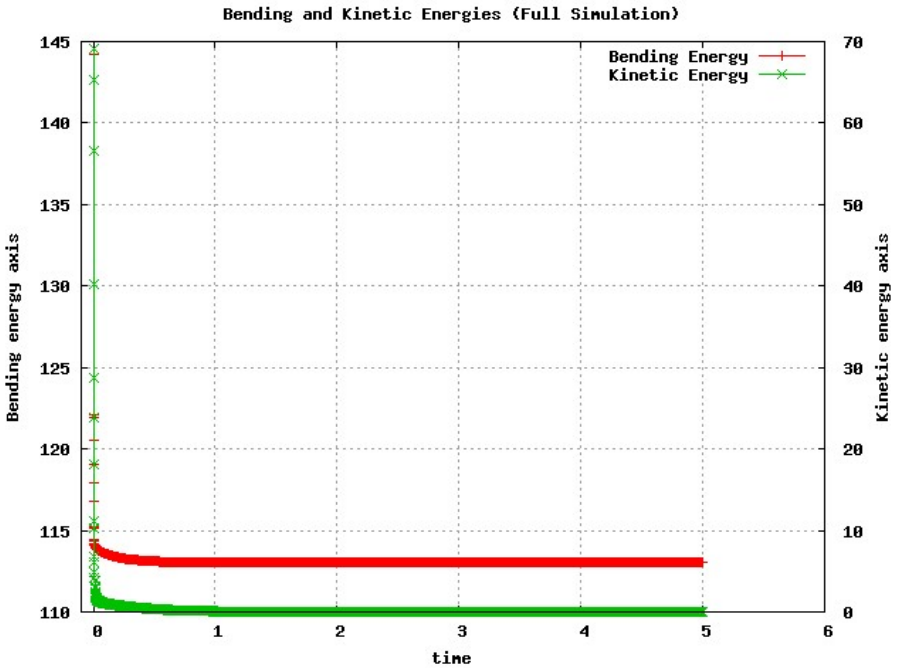


Figure 8.32: The figure shows a combined graph of the bending and kinetic energy as function of time corresponding to the simulation of Figure 8.31. The left axis shows the bending energy scale while the right one does it for the kinetic energy. The balls are formed in the interval $[0, 0.02]$. For the time scale of the simulation it looks like a vertical line but it is not.

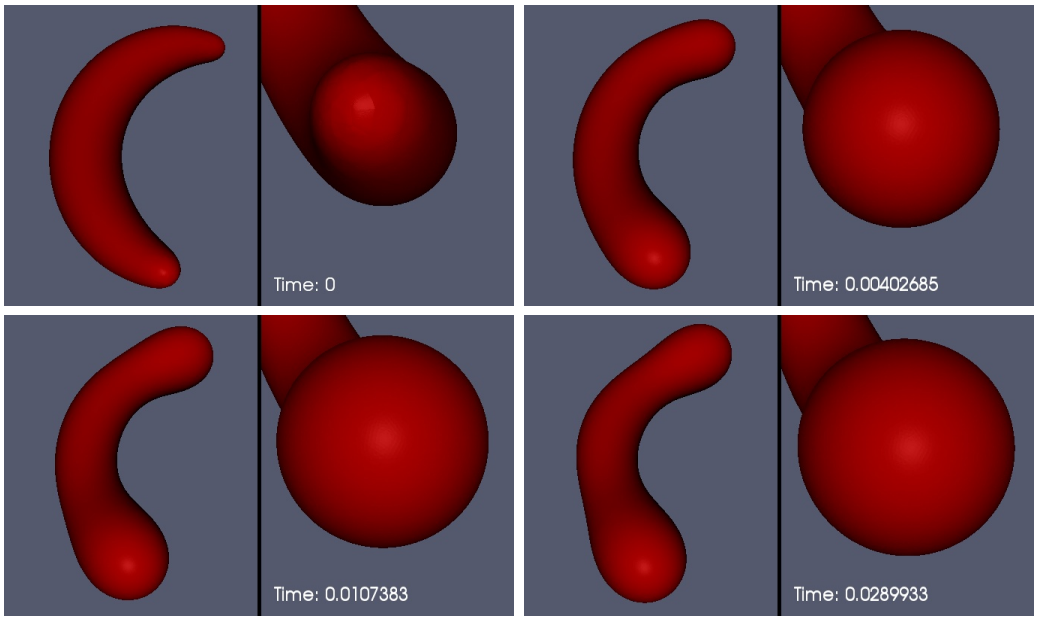


Figure 8.33: Time zooming to the interval $[0, 0.02]$ for the evolution of the simulation described in Figure 8.31. For each frame the picture on the left is a 3D view of the whole surface mesh and the picture on the right is a corresponding zoomed an angled to 3D view to see one of the ends. This stage that we refer to as the fast time scale of the simulation, is characterize by the formation of spherical shaped ends and a loose of the twist. At $t = 0.01$ the balls are formed and the evolution seems to reach a steady state. The bending the energy decreases about 21% in this time interval $\Delta t = 0.01$ (see Figure 8.34 for the energy plot). This is the first time scale that is detected.

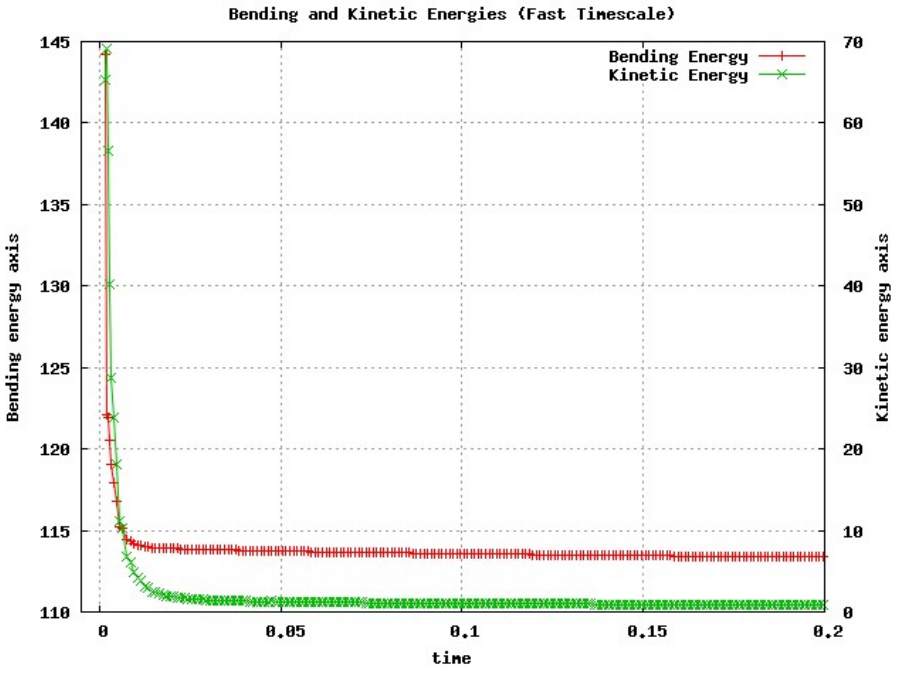


Figure 8.34: The figure shows a combined graph of the bending and kinetic energy as function of time corresponding to the simulation of Figure 8.31 in the time interval $[0, 0.2]$. The formation of balls occurs in the interval $[0, 0.02]$. The bending energy is reduced in this interval approximately 21%. Compare with the decrease observe in Figure 8.36.

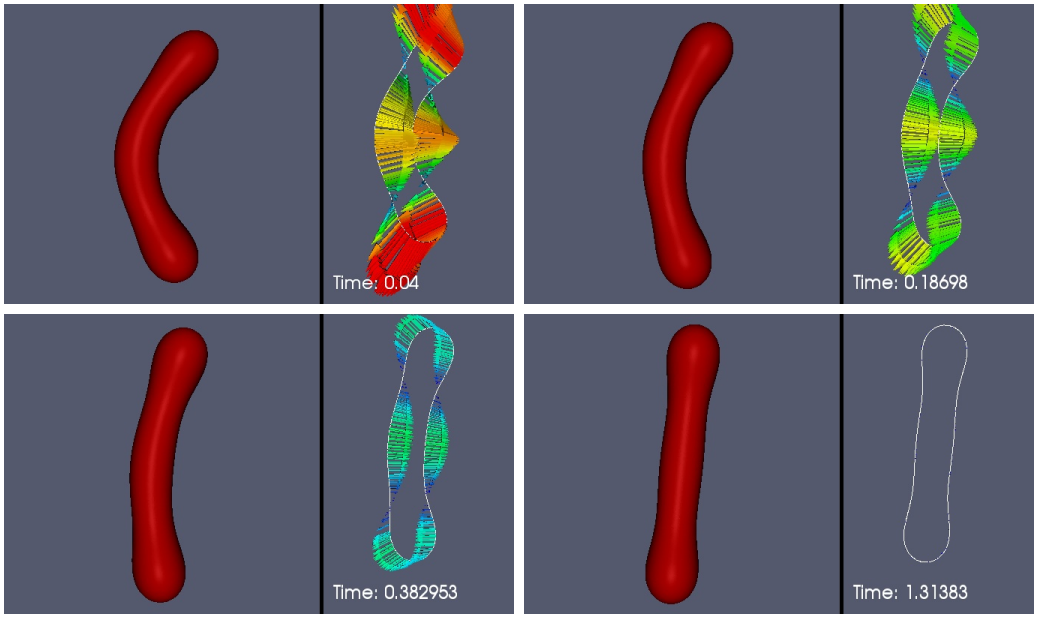


Figure 8.35: Time zooming to the interval $[0.04, 1.3]$ for the evolution of the simulation described in Figure 8.31. For each frame the picture on the left is a 3D view of the surface mesh and the picture on the right shows a 2D cut through an appearing symmetry plane together with the velocity field shown as arrows. This stage ,that we refer to as the slow time scale of the evolution, is characterize by the straitening of the ball shaped ends boomerang left at the end of the fast time scale of the evolution. The bending the energy decreases about .3% in this time interval $\Delta t = 1.0$ (see Figure 8.36 for the energy plot). This is the second time scale that is detected. The reason for the steps in the graph is that the output was saved with 5 significant digits and the decrease inside the steps is in the 6th.

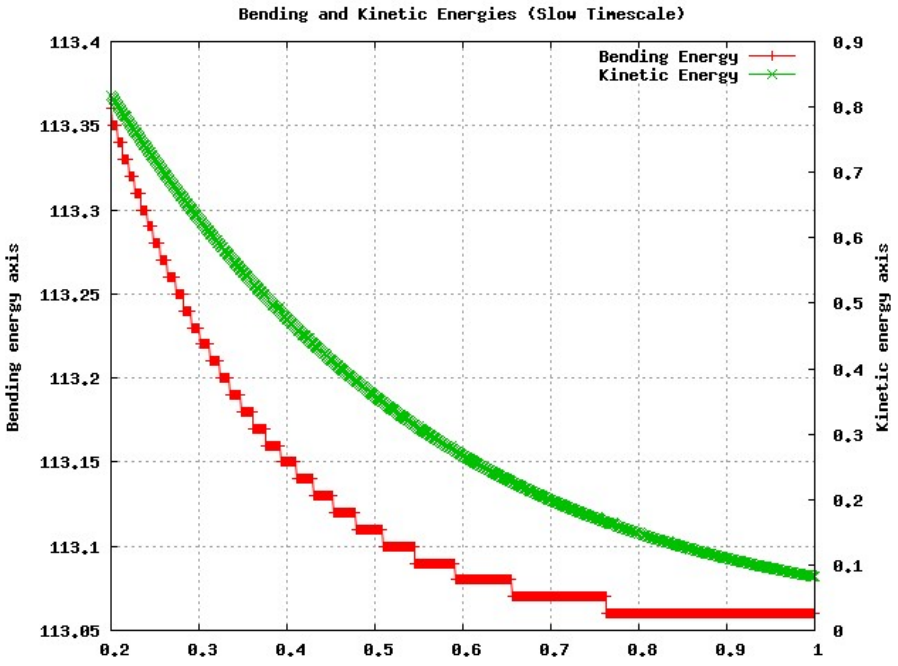


Figure 8.36: The figure shows a combined graph of the bending and kinetic energy as function of time corresponding to the simulation of Figure 8.31 in the time interval $[0.2, 1.0]$. The boomerang shape is straitened at about $t = 0.8$. The bending the energy decreases about .3%. Compare with the decrease observe in Figure 8.34.

8.4 Capillarity

In this section we present simulations for the capillarity problem of Section 4.3.1.

8.4.1 2D Star Shape

This is a star shape very similar to the one on Section 8.2.2 but with a fluid interior.

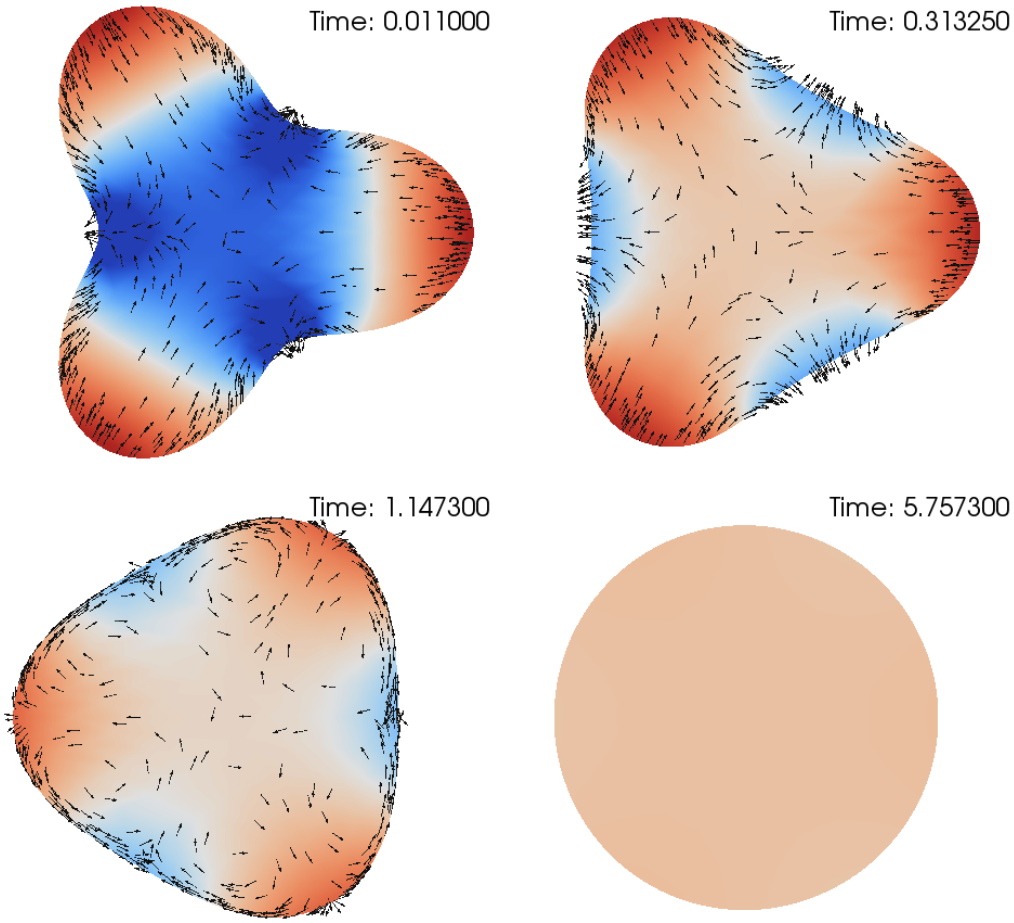


Figure 8.37: Evolution of a star shape subject to capillarity flow. A damping oscillating behavior falling to a circle is observed. The color represents the pressure and the arrows the direction of the velocity field (not the magnitude). Figure 8.38 shows the evolution of the corresponding perimeter and kinetic energy.

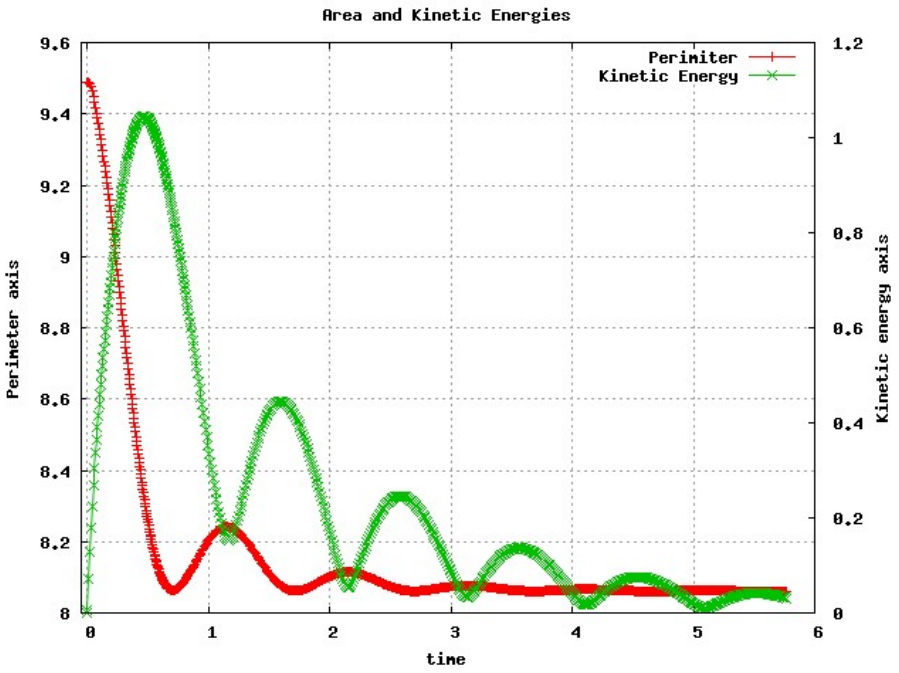


Figure 8.38: The figure shows a combined graph of the perimeter and kinetic energy as function of time corresponding to the simulation of Figure 8.37. The left axis shows the perimeter scale while the right one does it for the kinetic energy. Notice the damped oscillation characteristics of the inertial and viscous effects brought by the fluid.

8.4.2 Ellipsoid

8.5 Fluid Biomembrane

In this section we present simulations for the fluid biomembrane model of equations (4.50). In few word, the membrane is a massless object endowed with a bending and surface area energy that exerts force on and moves with the fluid.

In this section the inertial and viscous effects of the fluid extensively appear in the evolution (compare with Section 8.3 where these effects are absent). We also observe how there seems to be a coincidence between the the points of maximum bending energy and minimum kinetic energy and vice-versa. Both damped in time

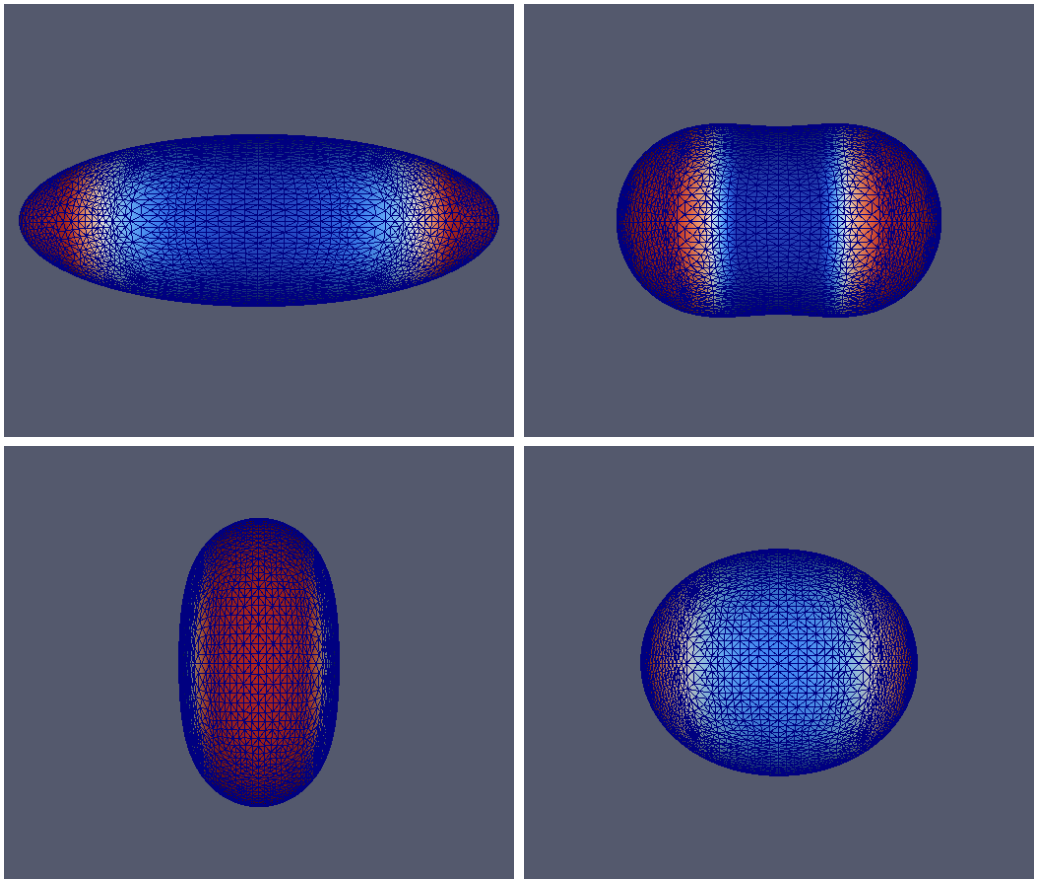


Figure 8.39: Evolution of an initial axisymmetric ellipsoid of aspect ratio 4x1x1 subject to a capillarity flow. A damped oscillating behavior falling to a sphere is observed. The color represents the pressure.

in the absence of external forces.

8.5.1 2D Fluid Banana in a Shear Force Field

The 2D initial mesh we consider has a banana or boomerang shape as shown in Figure 8.40. This simulation documents the effectivity of the smoothing techniques described in Section 7.4. In this simulation the domain suffers quite a dramatic motion. Not only it evolves with a area constrained bending surface force but it is also subject to an external shearing force that cause it to rotate faster and faster

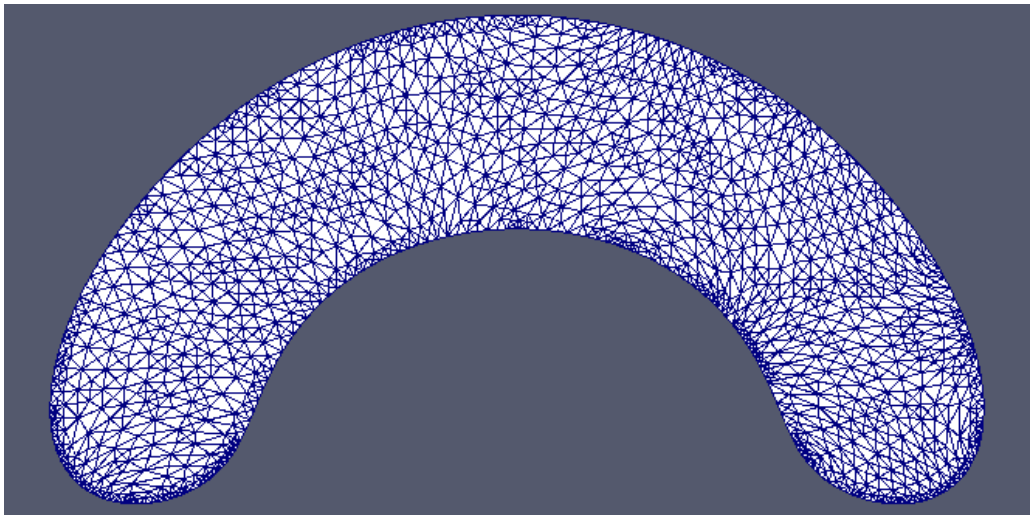


Figure 8.40: Initial 2D banana shaped mesh to be evolved with the fluid-biomembrane model.

in time. Even though all the nodes are moving with the flow the mesh preserves its quality due to the smoothing described in Algorithm 7.4.1 being applied in each time iteration.

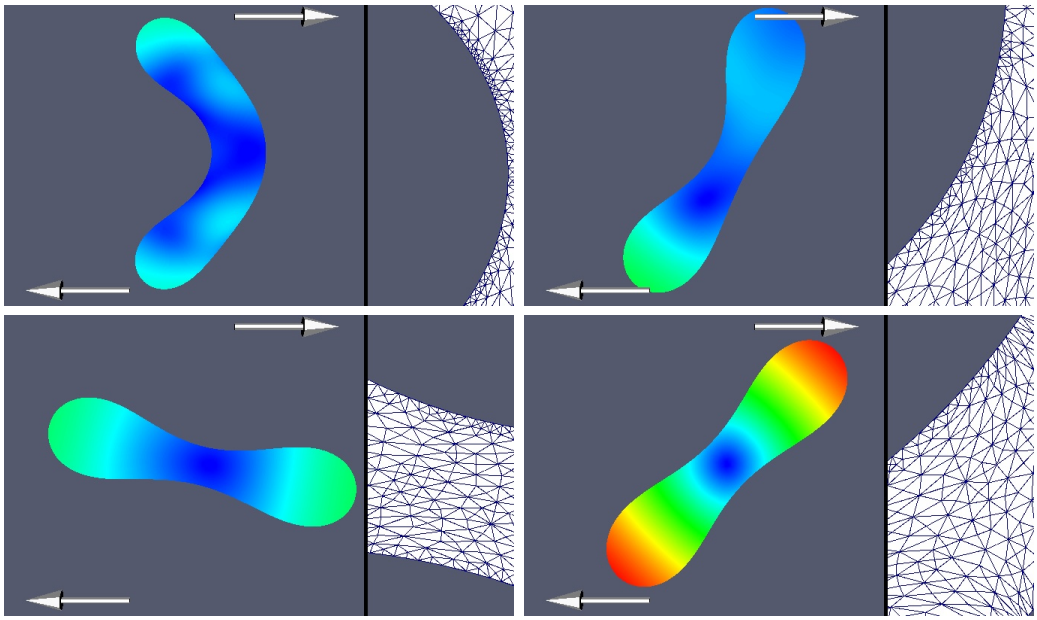


Figure 8.41: Evolution of the initial 2D banana shaped mesh of Figure 8.40 subject to a biomembrane fluid model evolution plus a discontinuous shearing force depicted in the picture with arrows. For each frame the picture on the left is a 2D view of the mesh colored by speed and the picture on the right a corresponding zoom showing the mesh wire frame. The evolution is characterize by the rounding and thickening of the ends while unbending its shape. But also at the same time it stars slowly rotating due to the external shear force. Then the object stars rotating faster and faster. It's important to notice how well the mesh is preserved by the methods of Section 7.4, without any remeshing done. Figure 8.42 shows the evolution of the corresponding bending and kinetic energy. And Figure 8.43 shows the evolution of the area Lagrange multiplier.

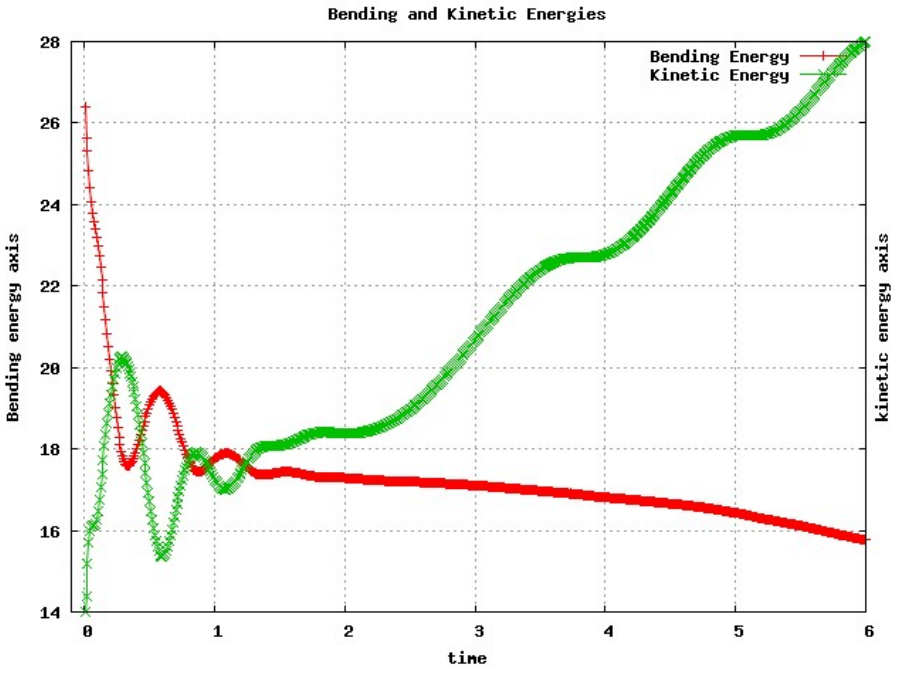


Figure 8.42: The figure shows a combined graph of the bending and kinetic energy as function of time corresponding to the simulation of Figure 8.41. The left axis shows the bending energy scale while the right one does it for the kinetic energy. Notice how the kinetic energy keeps increasing due to the applied external couple.

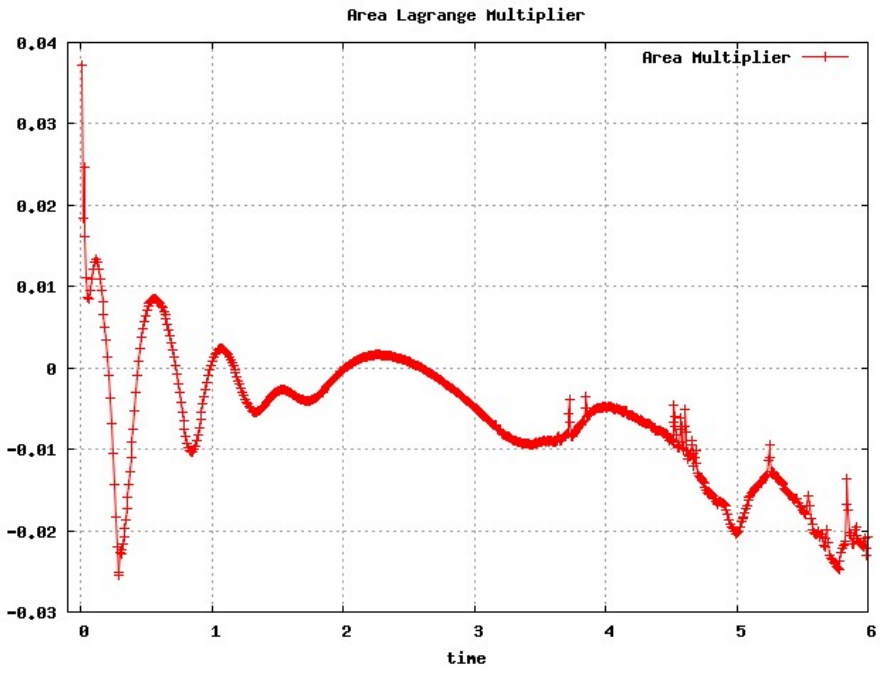


Figure 8.43: The figure shows the graph of the area Lagrange multiplier as function of time corresponding to the simulation of Figure 8.41.

8.5.2 3D Fluid Banana

As in the case of Willmore and geometric biomembrane flow the banana shape is an interesting candidate to use as initial shape because of its bended form. But now we consider the banana with fluid interior.

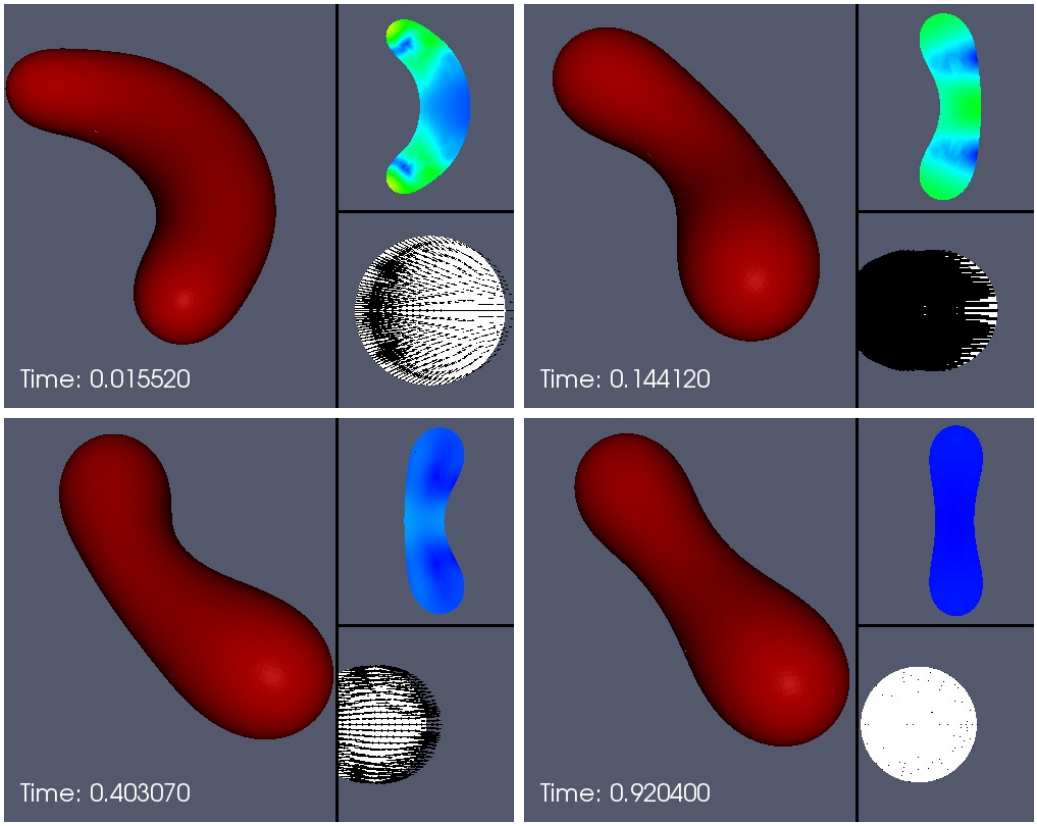


Figure 8.44: Evolution of an initial non axisymmetric 3D banana shaped volume mesh subject to a biomembrane fluid model evolution. For each frame the picture on the left is a 3D view of the mesh and the pictures on the right from to bottom show a longitudinal 2D cut colored by the fluid pressure and a transversal cut with the velocity field represented by arrows. Figure 8.45 show some stream lines of this flow. Figure 8.46 shows the evolution of the corresponding bending and kinetic energy. And Figure 8.47 shows the evolution of the area Lagrange multiplier.

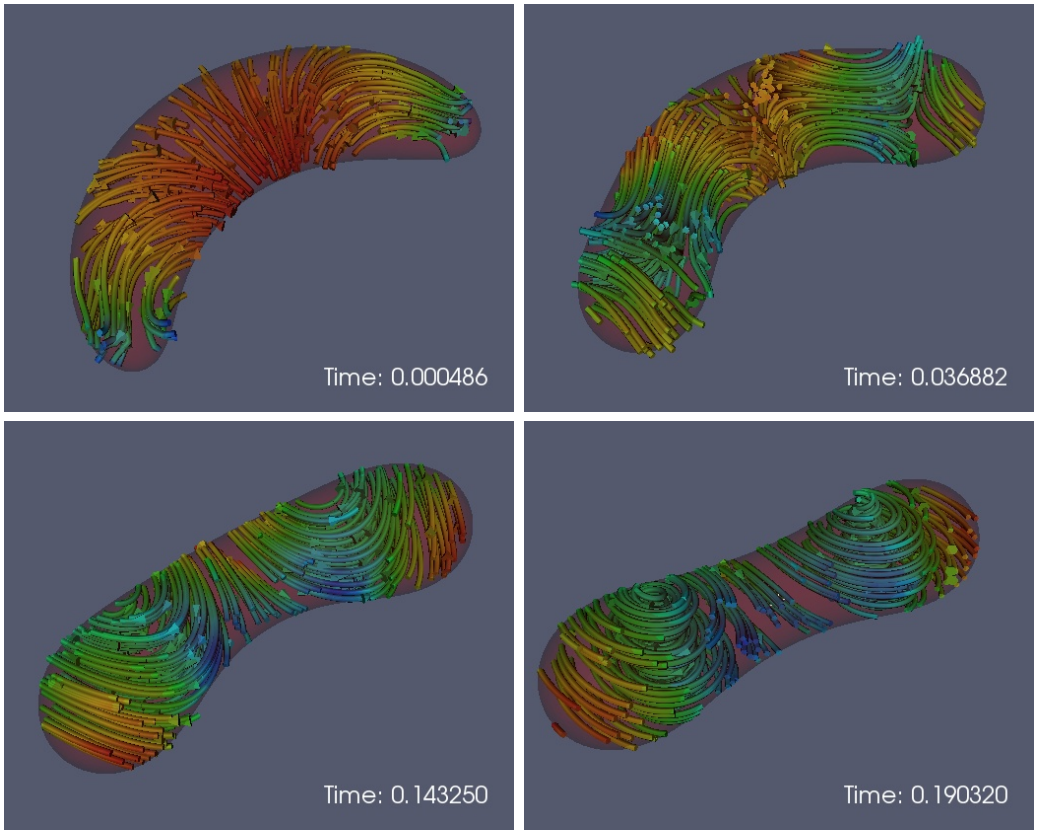


Figure 8.45: The figure shows the streamlines for the simulation of Figure 8.44 colored by fluid speed. In the first frame, fluid starts flowing from the center to the ends. At this point there is no indication of unbending. After the ends have been thickened by the pumped fluid, taking the characteristic spherical shape, we see some unbending or straitening of the banana. In the second frame we see how the fluid in the ends now flows in an unbending direction. Observe that still fluid is coming from the center of the banana toward the ends in a direction favorable to bending. At some point this last behavior changes to a favorable one for unbending, as can be seen in the third frame where the fluid goes from the center to the ends but from the other side. This behavior ends up producing two distinctive curls as shown in the fourth frame.

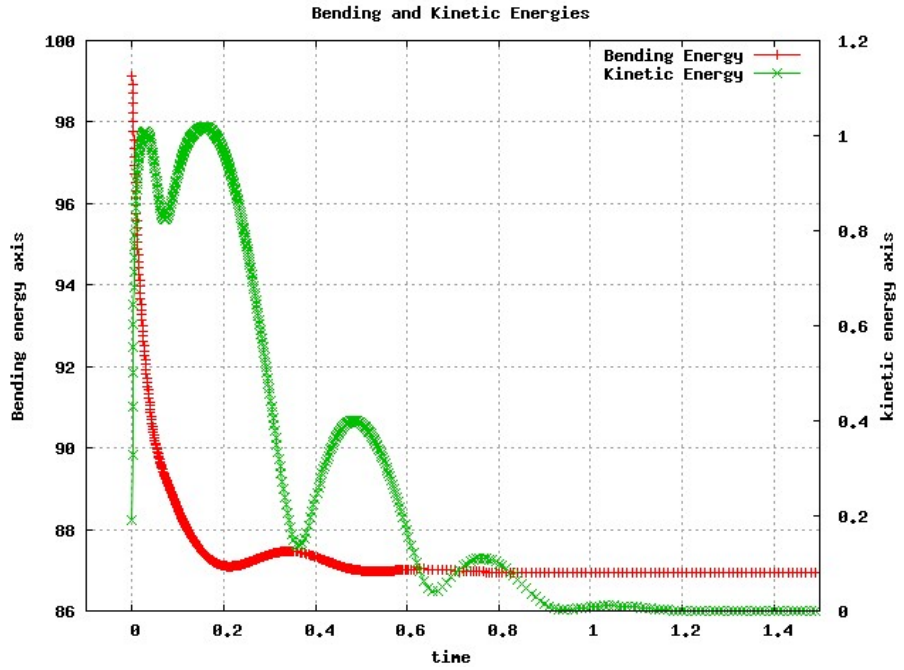


Figure 8.46: The figure shows a combined graph of the bending and kinetic energy as function of time corresponding to the simulation of Figure 8.44. The left axis shows the bending energy scale while the right one does it for the kinetic energy. Observe how there seems to be a coincidence between the the points of maximum bending energy and minimum kinetic energy and vice-versa. Also notice the damped behavior of the kinetic energy characteristic of the viscous dissipation. Finally the bending energy approaches the equilibrium as the kinetic energy approaches 0.

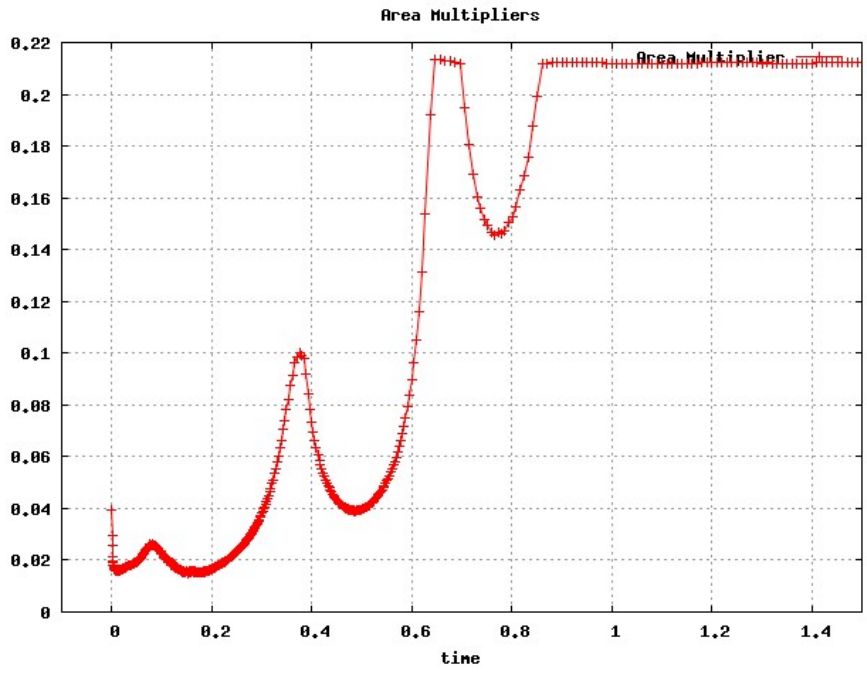


Figure 8.47: The figure shows the graph of the area Lagrange multiplier as function of time corresponding to the simulation of Figure 8.44.

8.5.3 3D Pinching Fluid Ellipsoid

In Section 8.3.2 we discovered for axisymmetric ellipsoids the approximate aspect ratio for which the geometric biomembrane flow pinches off. In this section we try this same aspect ratio for the initial shape but this time with fluid inside and governed by the fluid biomembrane flow.

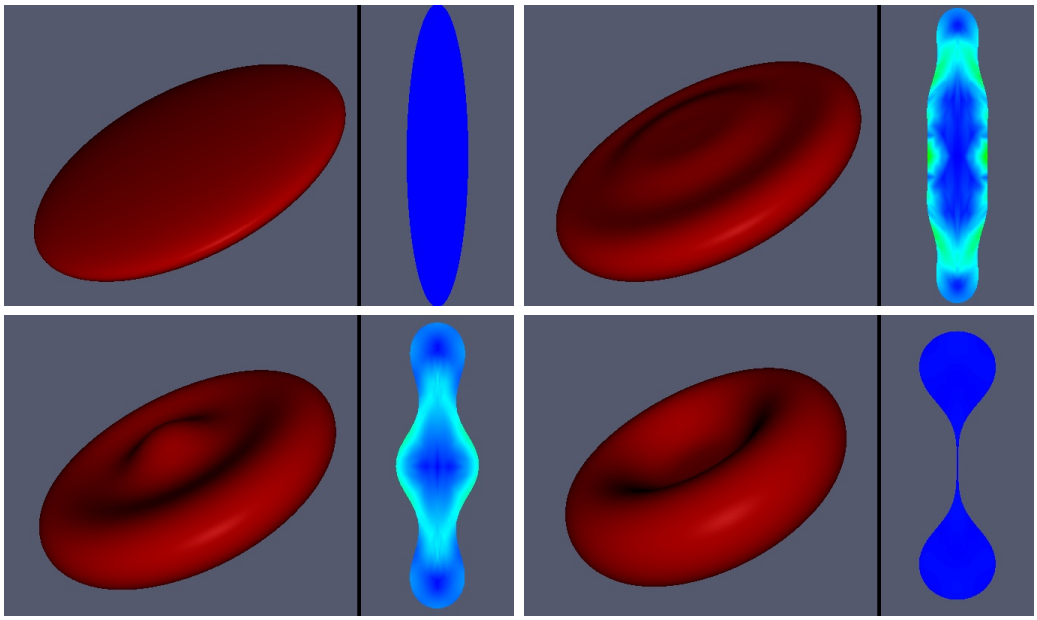


Figure 8.48: Evolution of an initial axisymmetric ellipsoid of aspect ratio 5x5x1 subject to a biomembrane fluid model evolution. For each frame the picture on the left is a 3D view of the mesh and the picture on the right is a 2D cut through a symmetry plane colored by the fluid pressure. In the first frame fluid from an annular region between 2 and 3 thirds of the circular radius flows toward the outer edge. At the same time fluid from the outer edge flows in collapsing with the first. This produces the thickening of the outer edge and a circular annular depression. In the second frame fluid from this depressed annular region flows also toward the center producing a central bump. Then the fluid from this bump flows to the outer edge turning the shape to look more and more like a torus.

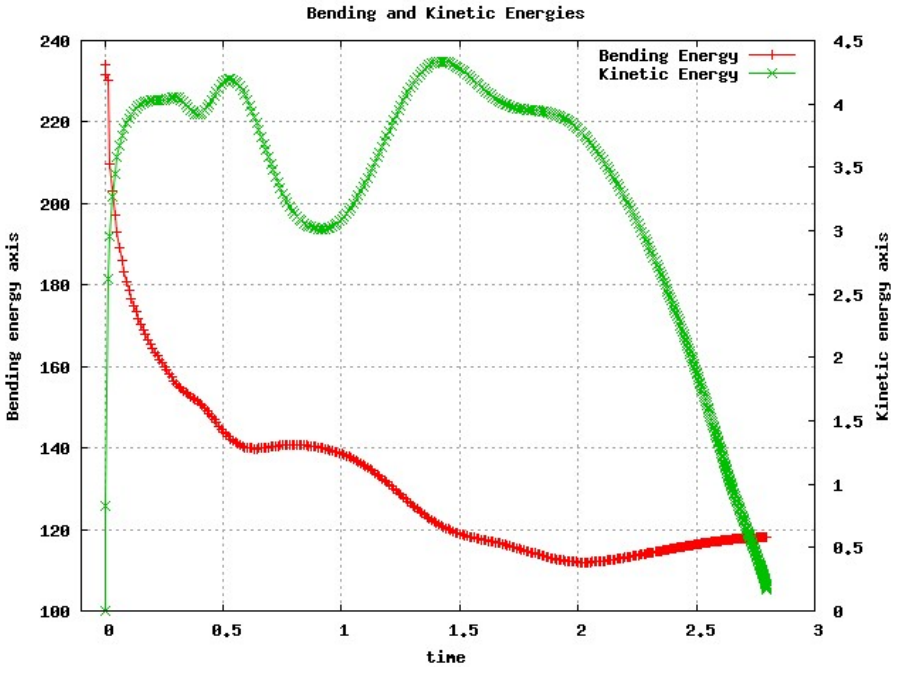


Figure 8.49: The figure shows a combined graph of the bending and kinetic energy as function of time corresponding to the simulation of Figure 8.48. The left axis shows the bending energy scale while the right one does it for the kinetic energy. At about $t = 2$ it can be observed that a local minimum corresponding more or less to the minimum of the geometric flow was reach but the fluid bypassed is indicating the pinching will be produce in this other shape (compare with Figure 8.25).

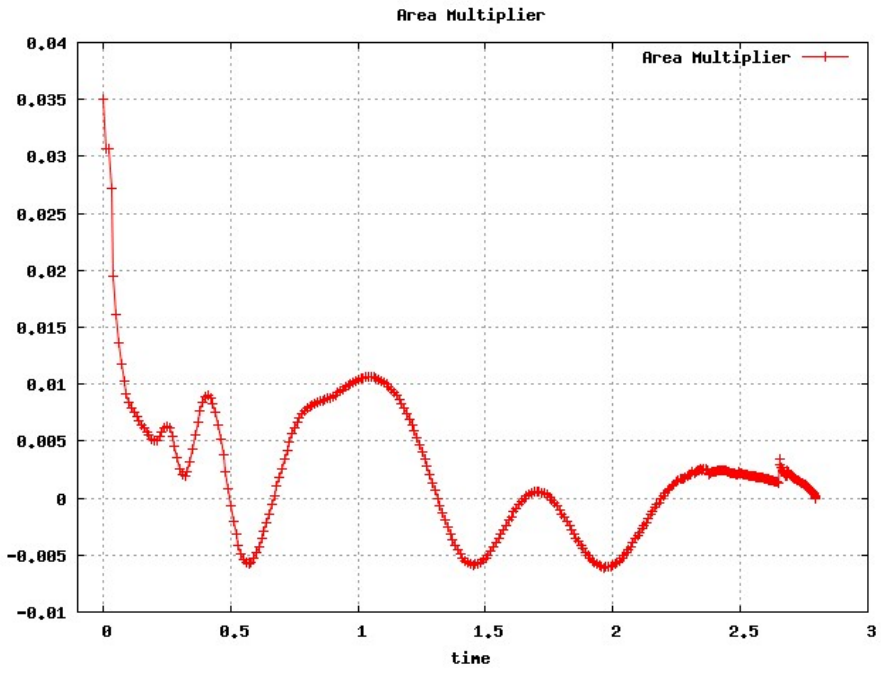


Figure 8.50: The figure shows the graph of the area Lagrange multiplier as function of time corresponding to the simulation of Figure 8.48.

8.5.4 3D Comparison: Geometric vs Fluid Ellipsoid

In this section we are going to compare the behaviors of the geometric biomembrane model of Figure 8.24 and the fluid biomembrane model of Figure 8.48 for an initial axisymmetric ellipsoid of aspect ratio 5x5x1. Off course the dynamics leading to equilibrium is different, but what is most remarkable is the different shape taken before pinching. This simulation illustrates that the fluid may change the shape in which topological changes happens.

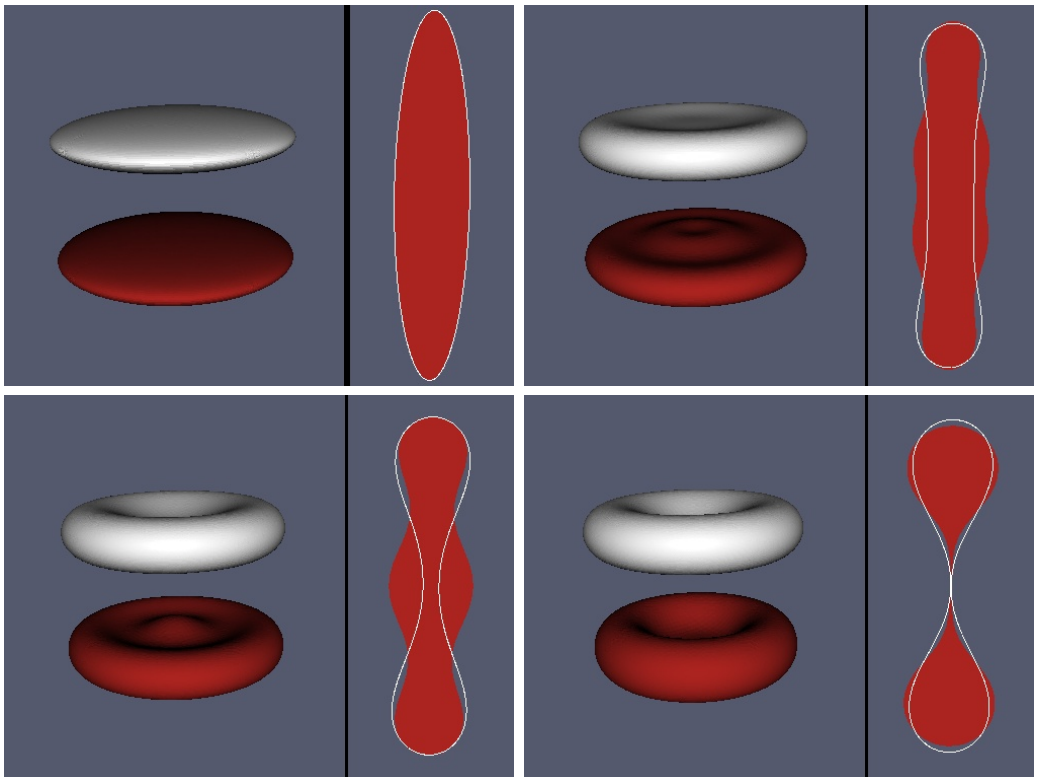


Figure 8.51: Comparison of the evolution of an initial axisymmetric ellipsoid of aspect ratio 5x5x1 subject to a geometric and fluid biomembrane model. For each frame the picture on the left is a 3D view of the surface meshes for the geometric (white) and fluid (red) flows. The picture on the left show corresponding 2D cuts through a symmetry plane. The detailed description for each separate simulation can be found in Figures 8.24 and 8.48. What is particularly interested is the different shapes of the last frame.

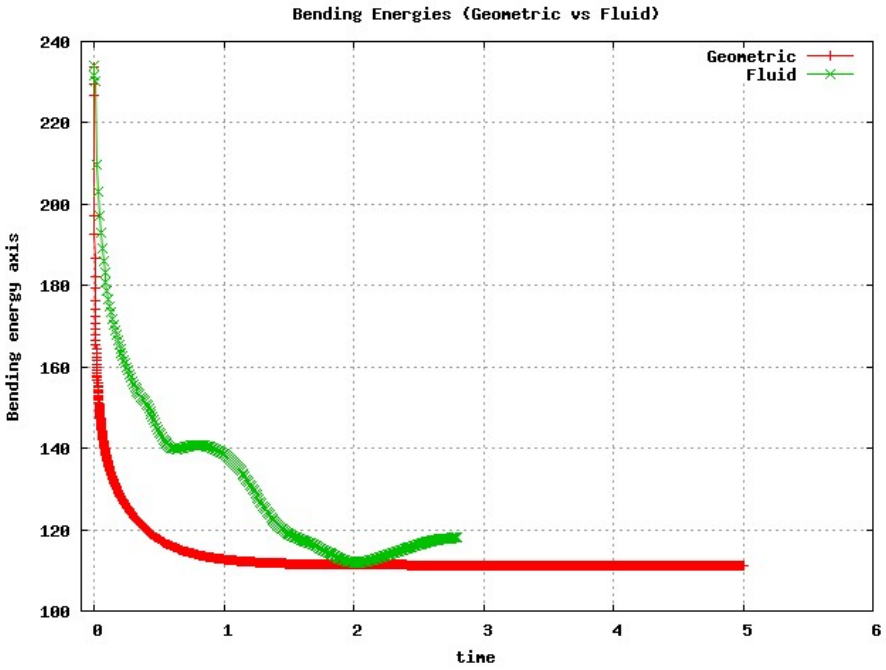


Figure 8.52: The figure shows the comparison of the bending energies for the geometric and the fluid biomembrane models as function of time. At $t \approx 2$ the fluid flow reaches the energy equilibrium level of the geometric flow. The fact that there is a fluid inside does not let it stay comfortable here so it changes its "about to pinch" shape. The fluid simulation does not continue because is indicating an imminent pinching that the parametric method cannot handle.

8.6 Conclusions

From the simulations presented in this chapter the following conclusions can be drawn:

- **Spherical caps**. For a geometric flow when the initial shape has distinctive ends (as in Simulations 8.3.1 and 8.3.4), the formation of spherical-shaped ends connected by a cylindrical neck seems to be the most effective way to minimize the energy. In fact, the formation of spherical-shaped ends decreases the energy several orders of magnitude more than straightening of a bended shape (see Simulation 8.3.4).

- **Red cells**. For a geometric flow when the initial shape is disc like (as Simulation 8.3.2) the evolution to decrease the energy is characterized by the thickening of the outer circular edge with the formation of a depression in the center.

- **Exponential decay of kinetic energy**. The geometric flow shows a clear exponential decay of the kinetic energy when approaching the equilibrium shape. The rate of decay seems to depend in a non trivial way on the equilibrium shape.

- **Fluid effect**. The coupling fluid produces quite a different dynamics from the geometric flow as it brings in the inertial and viscous effects of the fluid.

- **Equilibrium shapes**. Sometimes the final stationary shape coincides for the geometric and fluid models. But some other times (as in Simulations 8.5.3

and 8.3.2) they do not. In particular the comparison in Section 8.5.4 indicates that the fluid is an important factor dictating the shape prior to a topological change.

- **Geometry vs fluid**. The geometry seems to be initially the leading mechanism to decrease the energy, but later the fluid catches up and affects the dynamics. At this point an exchange between the bending and kinetic energy is observed.

Chapter 9

Mesh Generation

When we originally planned the project of implementing and using an AFEM as described in the previous chapters the initial mesh was supposed to be a given. The emphasis of our job was supposed to be the numerical approximation of a PDE over that given initial mesh using the AFEM. What in paper seemed to be reasonable was not so in practice. This initial mesh turned out not to be such a given.

Due to the adaptive finite element library we are using (`ALBERTA`) we need an initial mesh:

- compatible for refinements (i.e. no recursive infinite loop is created),
- with not too many macro elements (because they cannot be coarsened).

For the applications presented in Chapter 4 we need interesting initial 3D bulk and surface meshes. After some search and trials we could not find software that would generate a good quality, compatible for refinements, binary tree structured mesh that we could provide to our library. Specially the last two requirements were missing. A first attempt was to use DISTMESH [PS04]. The idea of distmesh is based on the physical analogy between a simplex mesh and a truss structure. Mesh points are nodes of the truss. Assuming an appropriate force-displacement function for the bars in the truss at each iteration, they solve for equilibrium. The forces move the nodes, and a delaunay triangulation is performed to swap edges. The real

signed distance function is essential to project boundary nodes and to distinguish the interior from the exterior. The code is in Matlab and very friendly to the user modification. The drawbacks are:

- the distance function is hard to provide (think of the distance function to a simple ellipse),
- the suggested spring driven force is quite slow to converge,
- in 3D, slivers (flat tetrahedra) are a comfortable position for the springs.

Faced with these issues, and inspired by the drawbacks of distmesh we decided to use the same `ALBERTA` library and our own tools of Chapter 7 to generate the initial mesh. We replace the spring driven force by an optimization flow. This works in 3D and is fast. And we replace the distance function by a deformation of a simple reference domain. It is important to say that the method thus obtained to generate the mesh has its own contribution to the mesh generation community. Some features like the compatibility for refinement together with a coarse underlying macro mesh are not available elsewhere.

In Chapter 7 we have developed tools to handle large deformations of a mesh. The idea is to apply them to mesh generation in the following way. Start with a simple region that can be easily meshed (square, circle, etc). Provide a motion that deforms the simple region into the desire one. Then the tools of Chapter 7 applied to this motion will give the desire mesh.

In Section 9.1 we describe how we represent a region to be meshed and the algorithm for mesh generation. One step required by the algorithm is the projection

to the boundary of newly created nodes. In Section 9.2 we discuss how this projection can be done. Finally in Section 9.3 we present a few examples of the method in action.

9.1 Mesh Generation Method

The first thing that it is necessary to mesh a domain is a description of its shape. We chose to describe the shape as a deformation of a reference domain with a rather simple shape (see figure 9.1). It is important that the reference domain has a simple

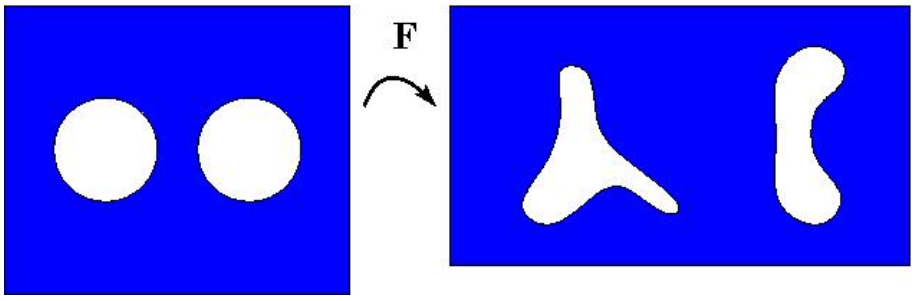


Figure 9.1: Illustration of a deformation. The region on the left is the reference domain, the region on the right is the domain we intend to mesh and the function $\boldsymbol{F}$ is the deformation.

shape because:

- we assume we have a good mesh of the reference domain;
- we take advantage of the analytical expressions that define its boundary (sphere or planes for example).

In case the deformation is too dramatic it is convenient to generate the mesh in progressive stages. For this purpose rather than a deformation we consider the

mesh Ω_T that we want to generate as the result of a motion $\mathbf{x}(\boldsymbol{p}, t)$ of the reference domain. In summary the necessary ingredients for the mesh generation to work are:

- a motion $\mathbf{x} : \hat{\Omega} \times [0, T] \mapsto \mathbb{R}^{d+1}$,
- a mesh of Ω_{t_0} for some time $t_0 \in [0, T]$, such that it is refinement compatible, of good quality and with a coarse underlying macro mesh,
- a reference mesh of the boundary $\hat{\Gamma} = \partial\hat{\Omega}$.

Remark 9.1.1 (Boundary Motion). The motion is only used on a small neighborhood of the boundaries, so in practice it is not necessary to provide it on the whole $\hat{\Omega}$, but only around the boundary $\hat{\Gamma}$.

The mesh generation process is described in Algorithm 9.1.1. For line 4 of Algorithm 9.1.1 we use the time selection tools of Section 7.6. For line 6 we use the optimization tools of Section 7.4. In this case for the boundary nodes the constraint is known (see Section 7.4.3.3) so we can provide it to the optimization method. As we assume a simple reference domain, it is more convenient to do the optimization here and take advantage of the simple analytical expression that the boundary takes. Line 7 and the projection to the boundary involve here are described in the next section.

9.2 Refinements and Coarsenings

When doing refinements in line 7 of Algorithm 9.1.1 the new boundary nodes have to be projected to the actual boundary. Depending on how good the approximation of the mesh to the boundary is, we propose two different strategies to project the

Algorithm 9.1.1 Algorithm for Mesh Generation

1: **procedure** GENERATE MESH($\mathbf{x}$, Ω_{t_0},$\hat{\Gamma}$)

2: Let T be the final time and $t = t_0$

3: **while** $t < T$ **do**

4: Find τ s.t. $\Gamma_{t+\tau}$ can be obtain by moving the nodes of Γ_t without crossing nodes in $\Omega_{t+\tau}$.

5: Let $t = t + \tau$, and move the boundary $\Gamma_t = \mathbf{x}(\hat{\Gamma}, t)$.

6: Adjust the interior and boundary nodes by an optimization flow.

7: Perform refinements and coarsening.

8: Repeat line 6 (optional)

9: **end while**

10: **end procedure**

nodes. The first one is very cheap computationally. But if the approximation is not good enough it may deteriorate the quality of the mesh even to the point of inverting elements. The second one is much better in terms of keeping the good quality but it is computationally more demanding. Below we describe each strategy.

9.2.1 Projection on Reference Configuration

This is the simple, computationally cheap strategy to project a boundary node to the actual boundary. It takes advantage of the analytically simple expression of the reference domain. We describe the method for the particular case when the reference boundary is the unit sphere.

Let $\boldsymbol{F}$ be the domain deformation, and $\boldsymbol{p}$ the reference coordinate correspond-

ing to a newly created boundary node. First the bisection algorithm does it job and defines $\boldsymbol{p}$ by linear interpolation. The location of the new node $\boldsymbol{x}^*$ in the deformed configuration is then obtained as follows: first let $\boldsymbol{p}^* = \frac{\boldsymbol{p}}{|\boldsymbol{p}|}$ and then $\boldsymbol{x}^* = \boldsymbol{F}(\boldsymbol{p})$ will give the projected new node.

9.2.2 Projection Through Distance Function

The natural way to project a point to a set is by using the distance function. It is based on the following mathematical result.

Theorem 9.2.1. *Let the surface Γ be the level set $f = 0$ of a smooth function f. Then there exists Γ^ϵ an ϵ-neighborhood of Γ such that given $\boldsymbol{x} \in \Gamma^\epsilon$ there is a unique $\boldsymbol{x}^* \in \Gamma$ such that $d(\boldsymbol{x}, \Gamma) = d(\boldsymbol{x}, \boldsymbol{x}^*)$. Moreover the following equation*

$$\nabla f(\boldsymbol{x}^*) = \alpha(\boldsymbol{x}^* - \boldsymbol{x}), \tag{9.1}$$

is satisfied for some α.

Proof. See [GT83, Appendix 14.6]. □

Now let $\boldsymbol{x}$ be the coordinates corresponding to a newly created boundary node. The bisection algorithm defines $\boldsymbol{x}$ by linear interpolation. The projection of $\boldsymbol{x}$ we are looking for is the solution $\boldsymbol{x}^*$ of equation (9.1). One way to find $\boldsymbol{x}^*$ taking advantage of the simple reference domain is described below for the particular case when the reference boundary is the unit sphere. Assume that the reference boundary $\hat{\Gamma} = S^d$ is the unit sphere. Let $\boldsymbol{F}$ be the domain deformation and $\Gamma := \boldsymbol{F}(\hat{\Gamma})$. Then the

surface Γ is the level set of the function

$$f(\boldsymbol{x}) := |\boldsymbol{F}^{-1}(\boldsymbol{x})|^2 - 1,$$

which in reference domain becomes

$$f(\boldsymbol{F}(\boldsymbol{p})) = |\boldsymbol{p}|^2 - 1. \tag{9.2}$$

Using the chain rule and equation (9.1) we get

$$(DF^{-1})^{\mathsf{T}}(\boldsymbol{p}) = \alpha(\boldsymbol{F}(\boldsymbol{p}) - \boldsymbol{x}),$$

which leads to the following system for $\boldsymbol{p}$ and α

$$\boldsymbol{p} - \alpha DF^{\mathsf{T}}(\boldsymbol{F}(\boldsymbol{p}) - \boldsymbol{x}) = \boldsymbol{0} \tag{9.3}$$

$$|\boldsymbol{p}|^2 - 1 = 0. \tag{9.4}$$

A Newton type of method can then be used to solve this system. Moreover equation (9.4) can be solve explicitly for one of the variables if desired.

9.3 Examples

In this section we present some examples of meshes generated using Algorithm 9.1.1. They include a two dimensional star shape, an ellipsoid and an ellipsoidal ring.

9.3.1 2D Star Shape

A three buds curve can be obtain as a deformation of the unit circle through the function $\boldsymbol{F} = ((0.8\cos(3\theta) + 1.5)\cos(\theta), (0.8\cos(3\theta) + 1.5)\sin(\theta))$, where θ is the

polar angle describing the circle. A motion that in time will reach this deformation starting from the unit circle can be

$$\mathbf{x}(\boldsymbol{p}, t) = (1 - t)\boldsymbol{p} + t\boldsymbol{F}(\boldsymbol{p}), \tag{9.5}$$

with $t \in [0, 1]$. In Figure 9.2 we show the work produced by Algorithm 9.1.1 when provided with the motion (9.5) and a reference unit disc mesh.

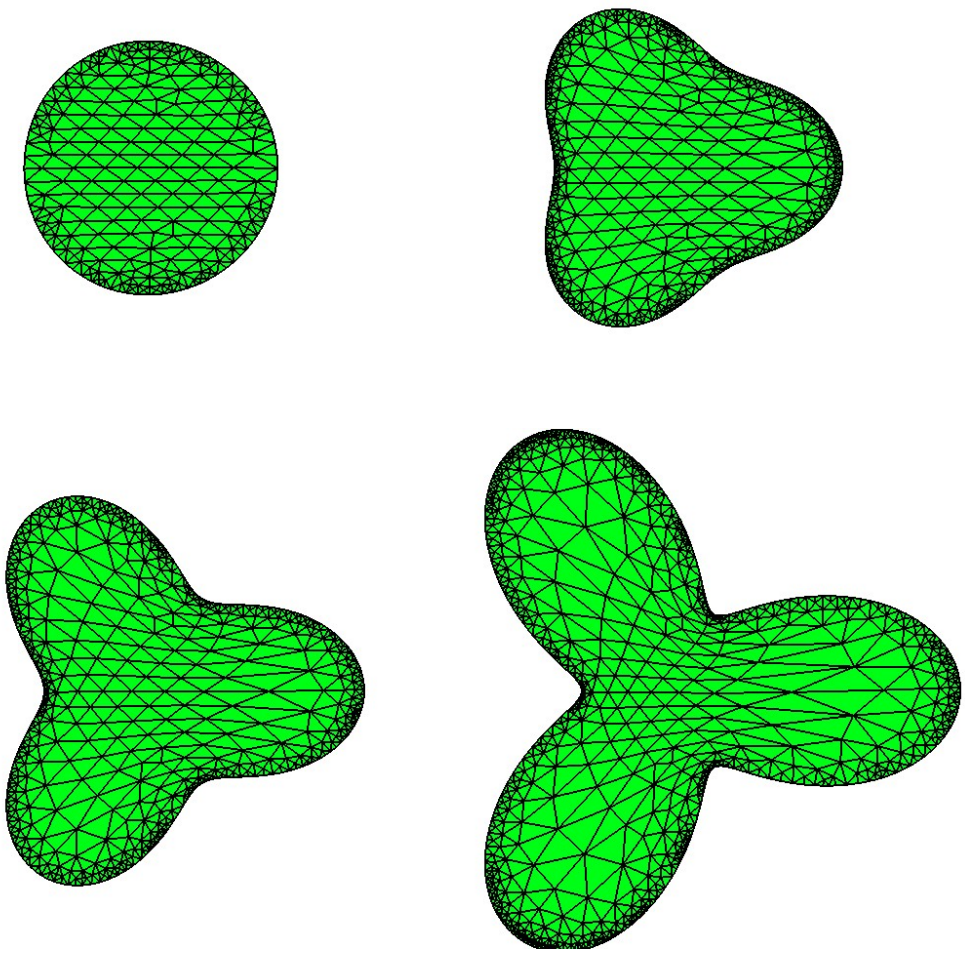


Figure 9.2: A three buds curve can be obtain as a deformation of the unit circle through the function $\boldsymbol{F} = ((0.8\cos(3\theta) + 1.5)\cos(\theta), (0.8\cos(3\theta) + 1.5)\sin(\theta))$, where θ is the polar angle describing the circle. A motion that in time will reach this deformation can be $\mathbf{x}(\boldsymbol{p}, t) = (1-t)\boldsymbol{p} + t\boldsymbol{F}(\boldsymbol{p})$, with $t \in [0, 1]$. The figure shows the work produced by Algorithm 9.1.1 when provided with the previous motion and a reference unit disc mesh. The iterations are 0 5 10 19.

9.3.2 3D Ellipsoid

A solid ellipsoid of aspect ratio 2x3x5 can be obtained as a deformation of the unit ball through the function

$$\boldsymbol{F}(\boldsymbol{p}) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1.5 & 0 \\ 0 & 0 & 2.5 \end{bmatrix} \boldsymbol{p}. \tag{9.6}$$

A motion that in time will reach this deformation can be

$$\mathbf{x}(\boldsymbol{p}, t) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \frac{1}{(t+1)} & 0 \\ 0 & 0 & \frac{1}{(3t+1)} \end{bmatrix} \boldsymbol{p}, \tag{9.7}$$

with $t \in [0, 0.5]$. In Figure 9.3 we show the work produced by Algorithm 9.1.1 when provided with the motion (9.7) and a reference mesh of the unit ball. Also transversal plane cuts showing some interior tetrahedra are shown in Figure 9.4. Shows the generation of an ellipsoid of aspect ratio 2x3x5.

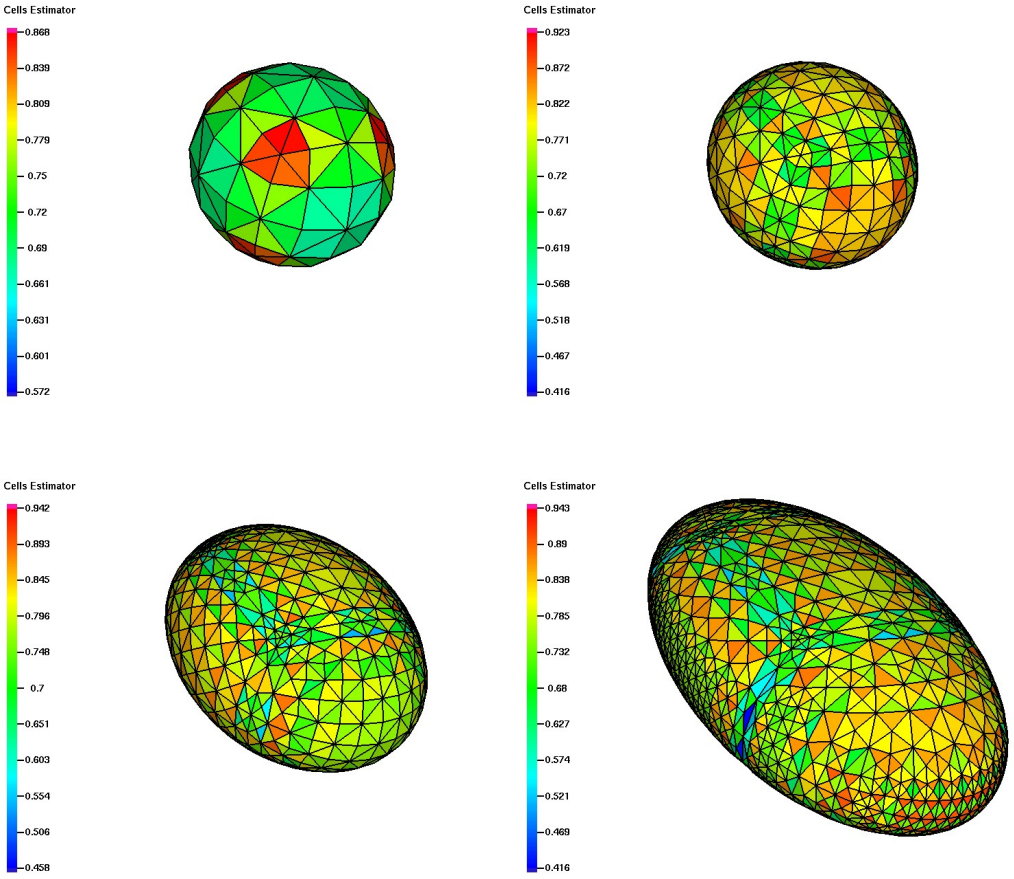


Figure 9.3: A solid ellipsoid of aspect ratio 2x3x5 can be obtained as a deformation of the unit ball through the function (9.6). A motion that in time will reach this deformation can be the one given in equation (9.7) with $t \in [0, 0.5]$. The figure shows the work produced by Algorithm 9.1.1 when provided with the previous motion and the refence mesh of the unit ball shown in the first frame. The tetrahedra are colored by its quality and the color bars on the left are scaled between the maximum and minimum values taken in each case. Also corresponding transversal plane cuts showing some interior tetrahedra are shown in Figure 9.4.

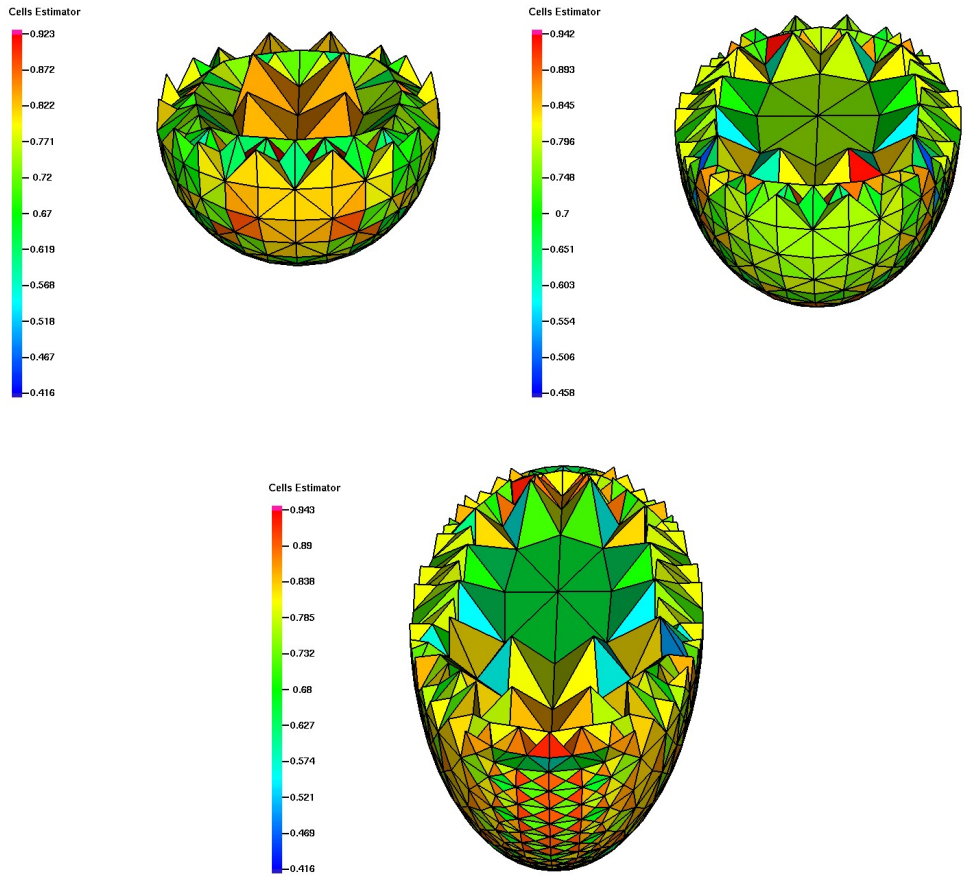


Figure 9.4: A solid ellipsoid of aspect ratio 2x3x5 can be obtained as a deformation of the unit ball through the function (9.6) as shown in Figure 9.3. This figure shows transversal plane cuts to appreciate some interior tetrahedra. The cuts correspond to the right side of the last three frames of Figure 9.3. The tetrahedra are colored by its quality and the color bars on the left are scaled between the maximum and minimum values taken in each case.

9.3.3 2D Ellipsoidal Ring

In this example we want to generate a 2D ellipsoidal ring, i.e. a ring whose outer and inner boundaries are ellipses. To accomplish this we proceed in two stages. First we generate a circular ring using the macro mesh of Figure 9.5. In this case

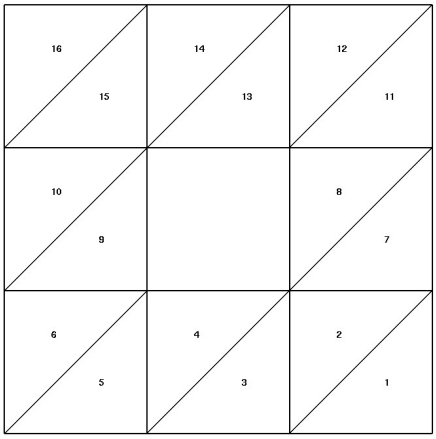


Figure 9.5: This macro mesh with 16 elements can be used to generate 2D shapes with and underlying ring topology. In particular this is the macro mesh used in the meshes of Figures 9.6 and 9.7.

only a deformation instead of a motion is used. And rather than providing the full deformation we only need to provide how the boundary nodes are affected by the deformation. Some frames corresponding to this process are shown in Figure 9.6. For the second part of the generation we use the circular ring generated in Figure 9.6 as the reference mesh. This one is evolved in time by a motion that sends the outer circle to a vertically elongated ellipse and the inner circle to a horizontally elongated one as shown in Figure 9.7.

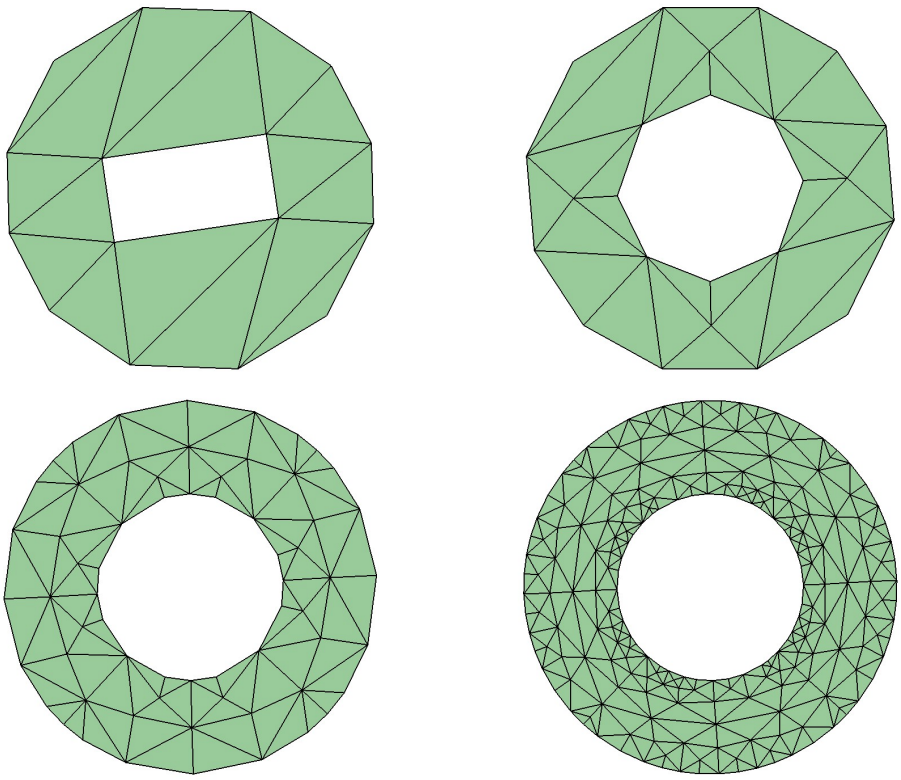


Figure 9.6: A circular ring using the macro mesh of Figure 9.5 is generated. This is the first step in the generation of the ellipsoidal ring shown in Figure 9.7. In this case only a deformation rather than a motion is necessary. The frames show different iterations of refinement, smoothing and projection to the boundary of the mesh generation process. The last frame is the reference mesh used in the generation shown in Figure 9.7.

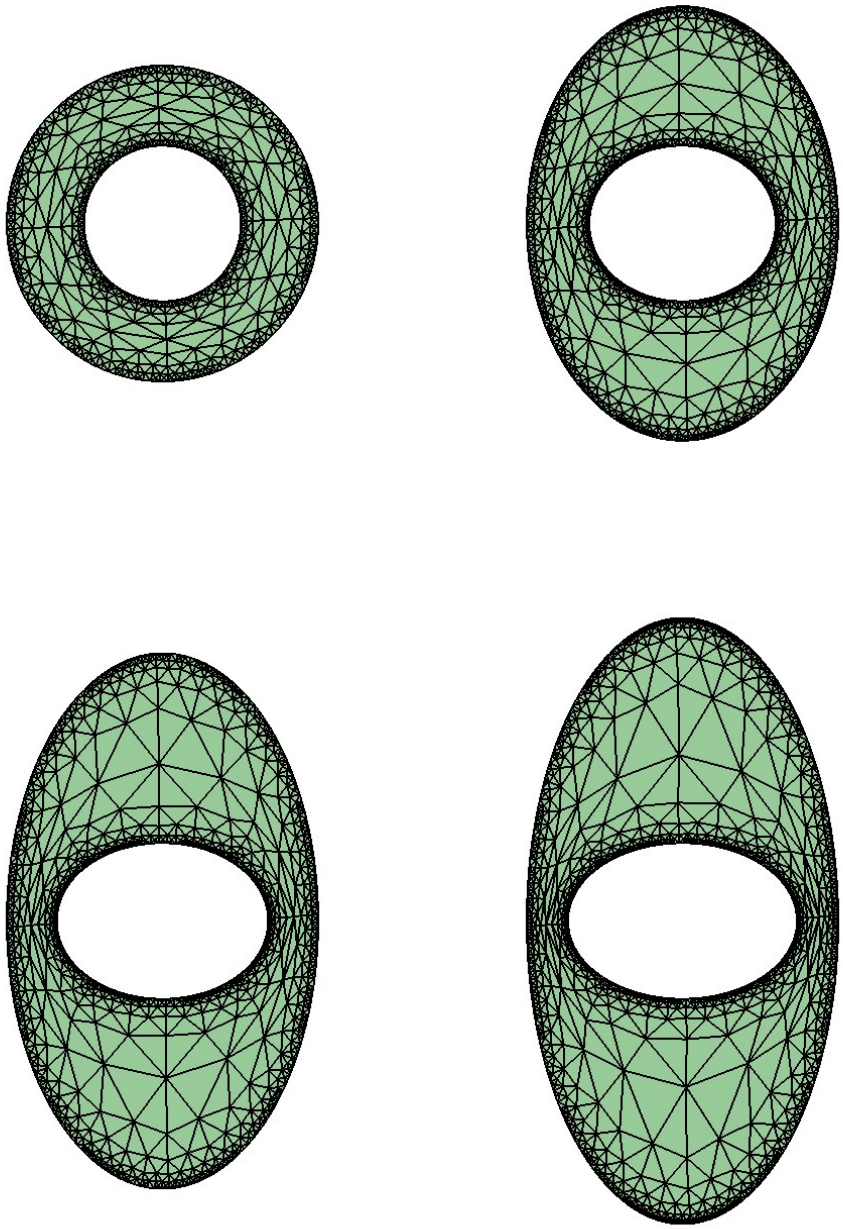


Figure 9.7: Generation of an ellipsoidal ring. The figure shows the second part of the generation process. We use the circular ring generated in Figure 9.6 as the reference mesh. This one is evolved in time by a motion that sends the outer circle to a vertically elongated ellipse and the inner circle to a horizontally elongated one.

Chapter 10

Open Problems and Future Work

Some open problems are:

- **Preconditioners**. The important issue of iterative solvers for large linear systems has not been addressed. Instead, a direct solver has been used for the simulations. It is still an open problem the development of practical and efficient preconditioner for the "basic" Willmore flow problem and related 4th order problems, both geometric and coupled with fluids. The resulting systems are extremely ill conditioned.

- **Initial curvature**. For the computation of the initial curvature of a polyhedral surface the method of gradient recovery has been proposed but not tested. The present simulations rely on piecewise quadratic representation of the surface which gives an acceptable, but not optimal, approximation of the initial curvature. This is an important issue since sometimes piecewise linear finite elements cannot get started.

- **Divergence free velocities**. The interpolation associated with the smoothing techniques may cause that an originally discrete divergence free velocity ends up having a non-zero divergence. In this case, the velocity should be projected to the space of discrete divergence free functions. The extend of this problem has not been fully checked, but seems to be minor with piecewise

quadratic finite elements.

- **Mesh generator**. The proposed mesh generator has been tested with domain topologies other than the sphere in 2d (see example of Section 9.3.3), but not yet in 3d.

Some future directions are:

- **Higher order time stepping**. For the computations, a semi implicit Euler scheme has been used for the time discretization. We believe using a higher order scheme for time discretization will improve the accuracy and efficiency of the computational method. Such methods have been used successfully for the capillarity problem [Bän01].

- **Optimal shape design**. The computational tools developed in this thesis can be applied to optimal shape design problems. We are currently investigating the shape of an object that immersed in a Stokes fluid minimizes the drag. Applications are ubiquitious in this field.

- **Reaction-diffusion PDEs**. Adding one or more reaction-diffusion PDEs on the membrane may account for the formation of domains (different types of lipids), line tension (interface within the surface), effect of surfactants and director fields, etc. These are all quite interesting applications that couple other physical and chemical effects to curvature, and are within grasp with the computational tools developed in this work.

Bibliography

[ABF84] D. N. Arnold, F. Brezzi, and M. Fortin, *A stable finite element for the Stokes equations*, Calcolo **21** (1984), no. 4, 337–344 (1985). MR MR799997 (86m:65136)

[Ant95] Stuart S. Antman, *Nonlinear problems of elasticity*, Applied Mathematical Sciences, vol. 107, Springer-Verlag, New York, 1995. MR MR1323857 (96c:73001)

[AO00] Mark Ainsworth and J. Tinsley Oden, *A posteriori error estimation in finite element analysis*, Pure and Applied Mathematics (New York), Wiley-Interscience [John Wiley & Sons], New York, 2000. MR MR1885308 (2003b:65001)

[AT72] R. J. Asaro and W. A. Tiller, *Interface morphology development during stress corrosion cracking: Part i. via surface diffusion*, Metallurgical and Materials Transactions B **3** (1972), no. 7, 1789–1796.

[Aub98] Thierry Aubin, *Some nonlinear problems in Riemannian geometry*, Springer Monographs in Mathematics, Springer-Verlag, Berlin, 1998. MR MR1636569 (99i:58001)

[Bän01] Eberhard Bänsch, *Finite element discretization of the Navier-Stokes equations with a free capillary surface*, Numer. Math. **88** (2001), no. 2, 203–235. MR MR1826211 (2002h:76079)

[BF91] F. Brezzi and M. Fortin, *Mixed and hybrid finite element methods*, Springer Series in Computational Mathematics, vol. 15, Springer-Verlag, New York, 1991. MR MR1115205 (92d:65187)

[BG04] Daniele Boffi and Lucia Gastaldi, *Stability and geometric conservation laws for ALE formulations*, Comput. Methods Appl. Mech. Engrg. **193** (2004), no. 42-44, 4717–4739. MR MR2091121 (2005d:65164)

[BHW03] Tobias Baumgart, Samuel T. Hess, and Watt W. Webb, *Imaging coexisting fluid domains in biomembrane models coupling curvature and line tension*, Nature **425** (2003), 821–824.

[BMN05] Eberhard Bänsch, Pedro Morin, and Ricardo H. Nochetto, *A finite element method for surface diffusion: the parametric case*, J. Comput. Phys. **203** (2005), no. 1, 321–343. MR MR2104399 (2005g:65139)

[BS01] Ivo Babuška and Theofanis Strouboulis, *The finite element method and its reliability*, Numerical Mathematics and Scientific Computation, The Clarendon Press Oxford University Press, New York, 2001. MR MR1857191 (2002k:65001)

[BS02] Susanne C. Brenner and L. Ridgway Scott, *The mathematical theory of finite element methods*, second ed., Texts in Applied Mathematics, vol. 15, Springer-Verlag, New York, 2002. MR MR1894376 (2003a:65103)

[Can70] P.B. Canham, *The minimum energy of bending as a possible explanation of the biconcave shape of the human red blood cell*, Journal of Theoretical Biology **26** (1970), no. 1, 61–81.

[CCS07] C. H. Arthur Cheng, Daniel Coutand, and Steve Shkoller, *Navier-Stokes equations interacting with a nonlinear elastic biofluid shell*, SIAM J. Math. Anal. **39** (2007), no. 3, 742–800 (electronic). MR MR2349865

[CDD+04] U. Clarenz, U. Diewald, G. Dziuk, M. Rumpf, and R. Rusu, *A finite element method for surface restoration with smooth boundary conditions*, Comput. Aided Geom. Design **21** (2004), no. 5, 427–445. MR MR2058390 (2005a:65017)

[Cia78] Philippe G. Ciarlet, *The finite element method for elliptic problems*, North-Holland Publishing Co., Amsterdam, 1978, Studies in Mathematics and its Applications, Vol. 4. MR MR0520174 (58 #25001)

[COS00] Fehmi Cirak, Michael Ortiz, and Peter Schroder, *Subdivision surfaces: a new paradigm for thin-shell finite-element analysis*, INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN ENGINEERING (2000).

[CS05] Daniel Coutand and Steve Shkoller, *Motion of an elastic solid inside an incompressible viscous fluid*, Arch. Ration. Mech. Anal. **176** (2005), no. 1, 25–102. MR MR2185858 (2006g:74045)

[CT94] J. W. Cahn and J. E. Taylor, *surface motion by surface diffusion*, Acta Metallurgica et Materialia **42** (1994), no. 4, 1045–1063.

[dC76] Manfredo P. do Carmo, *Differential geometry of curves and surfaces*, Prentice-Hall Inc., Englewood Cliffs, N.J., 1976, Translated from the Portuguese. MR MR0394451 (52 #15253)

[DD07] Alan Demlow and Gerhard Dziuk, *An adaptive finite element method for the Laplace-Beltrami operator on implicitly defined surfaces*, SIAM J. Numer. Anal. **45** (2007), no. 1, 421–442 (electronic). MR MR2285862

[DDE05] Klaus Deckelnick, Gerhard Dziuk, and Charles M. Elliott, *Computation of geometric partial differential equations and mean curvature flow*, Acta Numer. **14** (2005), 139–232. MR MR2168343 (2006h:65159)

[DE07] G. Dziuk and C. M. Elliott, *Finite elements on evolving surfaces*, IMA J. Numer. Anal. **27** (2007), no. 2, 262–292. MR MR2317005 (2008c:65253)

[Dem] Alan Demlow, *Higher-order finite element methods and pointwise error estimates for elliptic problems on surfaces*, Preprint from his webpage.

[DKS02] Gerhard Dziuk, Ernst Kuwert, and Reiner Schätzle, *Evolution of elastic curves in $\mathbb{R}^n$: existence and computation*, SIAM J. Math. Anal. **33** (2002), no. 5, 1228–1245 (electronic). MR MR1897710 (2003f:53117)

[DLW04] Qiang Du, Chun Liu, and Xiaoqiang Wang, *A phase field approach in the numerical study of the elastic bending energy for vesicle membranes*, J. Comput. Phys. **198** (2004), no. 2, 450–468. MR MR2062909 (2005b:74085)

[DLW06] ———, *Simulating the deformation of vesicle membranes under elastic bending energy in three dimensions*, J. Comput. Phys. **212** (2006), no. 2, 757–777. MR MR2187911 (2006f:74047)

[Dog06] Gunay Dogan, *A variational shape optimization framework for image segmentation*, Ph.D. thesis, University of Maryland, 2006.

[Dzi] Gerhard Dziuk, *Computational parametric willmore flow*, Preprint from.

[Dzi88] ———, *Finite elements for the Beltrami operator on arbitrary surfaces*, Partial differential equations and calculus of variations, Lecture Notes in Math., vol. 1357, Springer, Berlin, 1988, pp. 142–155. MR MR976234 (90i:65194)

[Dzi91] G. Dziuk, *An algorithm for evolutionary surfaces*, Numer. Math. **58** (1991), no. 6, 603–611. MR MR1083523 (91k:65042)

[ERM+03] J. M. Escobar, E. Rodriguez, R. Montenegro, G. Montero, and J. M. Gonzalez-Yuste, *Simultaneous untangling and smoothing of tetrahedral meshes, computer methods in applied mechanics and engineering*, Computer Methods in Applied Mechanics and Engineering **192** (2003), no. 25, 2775–2787.

[ES80] E.A. Evans and R. Skalak, *Mechanics and thermodynamics of biomembranes*, CRC Press (1980).

[FG00] Pascal Jean Frey and Paul-Louis George, *Mesh generation*, Hermes Science Publishing, Oxford, 2000, Application to finite elements, Translated from the 1999 French original by the authors. MR MR1807239 (2002c:65001)

[FN04] Luca Formaggia and Fabio Nobile, *Stability analysis of second-order time accurate schemes for ALE-FEM*, Comput. Methods Appl. Mech. Engrg. **193** (2004), no. 39-41, 4097–4116. MR MR2087006 (2005d:65135)

[Gas01] L. Gastaldi, *A priori error estimates for the arbitrary Lagrangian Eulerian formulation with finite elements*, East-West J. Numer. Math. **9** (2001), no. 2, 123–156. MR MR1836870 (2002m:65094)

[GH96] M. Giaquinta and S. Hildebrandt, *Calculus of variations. I*, Grundlehren der Mathematischen Wissenschaften [Fundamental Principles of Mathematical Sciences], vol. 310, Springer-Verlag, Berlin, 1996, The Lagrangian formalism. MR MR1368401 (98b:49002a)

[Gig06] Yoshikazu Giga, *Surface evolution equations*, Monographs in Mathematics, vol. 99, Birkhäuser Verlag, Basel, 2006, A level set approach. MR MR2238463 (2007j:53071)

[Glo03] R. Glowinski, *Finite element methods for incompressible viscous flow*, Handbook of numerical analysis, Vol. IX, Handb. Numer. Anal., IX, North-Holland, Amsterdam, 2003, pp. 3–1176. MR MR2009826

[GR86] V. Girault and P.-A. Raviart, *Finite element methods for Navier-Stokes equations*, Springer Series in Computational Mathematics, vol. 5, Springer-Verlag, Berlin, 1986, Theory and algorithms. MR MR851383 (88b:65129)

[GT83] David Gilbarg and Neil S. Trudinger, *Elliptic partial differential equations of second order*, second ed., Grundlehren der Mathematischen Wissenschaften [Fundamental Principles of Mathematical Sciences], vol. 224, Springer-Verlag, Berlin, 1983. MR MR737190 (86c:35035)

[Gur81] Morton E. Gurtin, *An introduction to continuum mechanics*, Mathematics in Science and Engineering, vol. 158, Academic Press Inc. [Harcourt Brace Jovanovich Publishers], New York, 1981. MR MR636255 (84c:73001)

[Hei] C. Heine, *Isoparametric finite element approximations of curvature on hypersurfaces*, Preprint from his webpage.

[Hel73] W Helfrich, *Elastic properties of lipid bilayers - theory and possible experiments*, Zeitschrift Fur Naturforschung C-A Journal Of Biosciences **28** (1973), 693.

[HSWW01] W. Hoffmann, A. H. Schatz, L. B. Wahlbin, and G. Wittum, *Asymptotically exact a posteriori estimators for the pointwise gradient error on each element in irregular meshes. I. A smooth problem and globally quasi-uniform meshes*, Math. Comp. **70** (2001), no. 235, 897–909 (electronic). MR MR1826572 (2002a:65178)

[HZE07] Dan Hu, Pingwen Zhang, and Weinan E, *Continuum theory of a moving membrane*, Phys. Rev. E (3) **75** (2007), no. 4, 041605, 11. MR MR2358577

[Jen77a] James T. Jenkins, *The equations of mechanical equilibrium of a model membrane*, SIAM J. Appl. Math. **32** (1977), no. 4, 755–764. MR MR0441076 (55 #13942)

[Jen77b] ———, *Static equilibrium configurations of a model red blood cell*, J. Math. Biology **4** (1977), no. 2, 150–169.

[KKS08] Daniel Koster, Oliver Kriessl, and Kunibert G. Siebert, *Design of finite element tools for coupled surface and volume meshes*, Numer. Math. Theor. Meth. Appl. **1** (2008)), no. 3, 245–274.

[Knu01] Patrick M. Knupp, *Algebraic mesh quality metrics*, SIAM J. Sci. Comput. **23** (2001), no. 1, 193–218 (electronic). MR MR1860911 (2003k:65162)

[Kos94] Igor Kossaczký, *A recursive approach to local mesh refinement in two and three dimensions*, J. Comput. Appl. Math. **55** (1994), no. 3, 275–288. MR MR1329875 (95m:65207)

[Kos06] Daniel Koster, *Numerical simulation of acoustic streaming on saw-driven biochips*, Ph.D. thesis, Universitat Augsburg, 2006.

[LAF97] Carl Ollivier-Gooch Lori A. Freitag, *Tetrahedral mesh improvement using swapping and smoothing*, International Journal for Numerical Methods in Engineering **40** (1997), no. 21, 0029–5981.

[LN05] Omar Lakkis and Ricardo H. Nochetto, *A posteriori error analysis for the mean curvature flow of graphs*, SIAM J. Numer. Anal. **42** (2005), no. 5, 1875–1898 (electronic). MR MR2139228 (2006b:65141)

[Mek05] Khamron Mekchay, *Convergence of adaptive finite element methods*, Ph.D. thesis, University of Maryland, 2005.

[MMN] Khamron Mekchay, Pedro Morin, and Ricardo H. Nochetto, *Afem for the laplace-beltrami operator: design and conditional contraction property*, to appear.

[MR05] Sebastián Montiel and Antonio Ros, *Curves and surfaces*, Graduate Studies in Mathematics, vol. 69, American Mathematical Society, Providence, RI, 2005, Translated and updated from the 1998 Spanish edition by the authors. MR MR2165790 (2006g:53006)

[Mul56] W. W. Mullins, *Two-dimensional motion of idealized grain boundaries*, Journal of Applied Physics **27** (1956), no. 8, 900–904.

[PS04] Per-Olof Persson and Gilbert Strang, *A simple mesh generator in Matlab*, SIAM Rev. **46** (2004), no. 2, 329–345 (electronic). MR MR2114458 (2005i:65005)

[Rus05] Raluca E. Rusu, *An algorithm for the elastic flow of surfaces*, Interfaces Free Bound. **7** (2005), no. 3, 229–239. MR MR2171130 (2006g:53099)

[Sam69] Hans Samelson, *Orientability of hypersurfaces in R^n*, Proc. Amer. Math. Soc. **22** (1969), 301–302. MR MR0245026 (39 #6339)

[SBL91] Udo Seifert, Karin Berndl, and Reinhard Lipowsky, *Shape transformations of vesicles: Phase diagram for spontaneous- curvature and bilayer-coupling models*, Phys. Rev. A **44** (1991), no. 2, 1182–1202.

[SBR+03] David Steigmann, Eveline Baesu, Robert E. Rudd, Jim Belak, and Mike McElfresh, *On the variational theory of cell-membrane equilibria*, Interfaces Free Bound. **5** (2003), no. 4, 357–366. MR MR2031462 (2004j:74095)

[SDV93] B. J. Spencer, S. H. Davis, and P. W. Voorhees, *Morphological instability in epitaxially strained dislocation-free solid films: Nonlinear evolution*, Phys. Rev. B **47** (1993), no. 15, 9760–9777.

[Sei97] Udo Seifert, *Configurations of fluid membranes and vesicles*, Advances in Physics **46** (1997), no. 1, 13–137.

[SS05] Alfred Schmidt and Kunibert G. Siebert, *Design of adaptive finite element software*, Lecture Notes in Computational Science and Engineering, vol. 42, Springer-Verlag, Berlin, 2005, The finite element toolbox ALBERTA, With 1 CD-ROM (Unix/Linux). MR MR2127659 (2005i:65003)

[Ste99] D. J. Steigmann, *Fluid films with curvature elasticity*, Arch. Ration. Mech. Anal. **150** (1999), no. 2, 127–152. MR MR1736701 (2000k:76014)

[Ste03] David J. Steigmann, *Irreducible function bases for simple fluids and liquid crystal films*, Z. Angew. Math. Phys. **54** (2003), no. 3, 462–477. MR MR2048665 (2004m:76011)

[SZ92] Jan Sokołowski and Jean-Paul Zolésio, *Introduction to shape optimization*, Springer Series in Computational Mathematics, vol. 16, Springer-Verlag, Berlin, 1992, Shape sensitivity analysis. MR MR1215733 (94d:49002)

[Tem84] R. Temam, *Navier-Stokes equations*, Studies in Mathematics and its Applications, vol. 2, North-Holland Publishing Co., Amsterdam, 1984. MR MR769654 (86m:76003)

[TH73] C. Taylor and P. Hood, *A numerical solution of the Navier-Stokes equations using the finite element technique*, Internat. J. Comput. & Fluids **1** (1973), no. 1, 73–100. MR MR0339677 (49 #4435)

[Wil93] T. J. Willmore, *Riemannian geometry*, Oxford Science Publications, The Clarendon Press Oxford University Press, New York, 1993. MR MR1261641 (95e:53002)