Geometric Elasticity for Graphics, Simulation, and Computation

Thesis by

Patrick Sanan

In Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy

California Institute of Technology

Pasadena, California

2014

(Defended November 27, 2013)

To my parents. Each day I see more of what you have given me.

Acknowledgments

The work in this thesis would not have been possible without the innumerable people who have supported me. Some I have known my entire life, some for a few years, some during a university course, and some only through writing. Many more helped without my knowledge, and I thank not only them and the people below but the organizations and society that have supported my education.

I've been blessed with wonderful teachers and advisors throughout my many years of school. I first encountered computer graphics at in Mr. Grunder's middle school industrial technology course; somehow I got course credit for animating battling robots. At Las Lomas High School, Ms. Lopilato would stay after to class to answer our physics questions, Ms. Furstenthal's English class was, hands-down, the most useful course I've taken, and Mr. Hilton wisely eschewed rehashing the textbook's geometry in favor of teaching us to avoid pseudoscience and how to best minimize a cat's surface area to volume ratio.

At UC San Diego, Sia Nemat-Nasser told us, "This is supposed to be difficult", Robert Bitmead demonstrated that narrative is the soul of signal processing pedagogy, Carl Fitzgerald led me to my first revelatory proof, Tom Erbe and Peter Otto revealed the wonders of computer music, Vlado Lubarda brought joy to solid mechanics, Stefan Llewellyn-Smith gave me my first applied mathematics research experience, and Michael Holst got me hooked on applied math and encouraged me to apply to Caltech.

At the University if Manchester, Ricardo Climent showed me the true meaning of 'interdisciplinary' and letting me turn the computer lab into a physical modeling audio render farm. Thanks to Jed Brown for mentoring me over the summer at Argonne National Lab and for vastly expanding my concept of how much a single person can be an expert in. My sincerest thanks and appreciation to my advisor at Caltech, Peter Schröder, who introduced me to the wonderful world of computational geometry and gave me incredible latitude to explore.

I thank the authors of two textbooks: Thomas Hughes and the late Jerrold Marsden for their book "Foundations of Mathematical Elasticity"[39], and Nick Higham for his "Functions of Matrices"[19]. Mathematical exposition is usually technically precise, often relevant, sometimes clear, and occasionally fun to read, but rarely combines all of these qualities. Beautiful monographs require the author's expertise but also from his or her passion for the topic and regard for the reader. I return to these texts as to timeless music; often and with revelation.

Joan and Irwin Jacobs and the UC Regents funded the scholarships that supported me as an undergraduate at UCSD, Tony Thornley sponsored the scholarship to send me to the University of Manchester, the Kaplun fellowship supported me during my first year at Caltech, Caltech employed me as a teaching assistant and instructor, Gautham Krishnamurthy and Rhythm and Hues Studios hired me as an intern, and to Argonne MCS and the TAO and PETSc teams supported me as a Givens Scholar.

It's been an honor to work with Applied and Computational Mathematics at Caltech. My sincerest thanks to my thesis committee: Peter Schröder, Mathieu Desbrun, Oscar Bruno, and Houman Owhadi. Thank you as well to Tom Hou and Joel Tropp, and to Jeff Ovall and Guo Luo. To be exposed to such diverse interests and uniform brilliance is a once-in-a-lifetime blessing. Thank you to my class: Mike McCoy, Xin Hu, and Tomasz Tyranowski. Thank you to Sheila Shull, Sydney Garstang, Maria Lopez, and Lisa Knox for keeping everything running so smoothly. Thanks to the Caltech and JPL communities for keeping me inspired. Thanks to Alex Gittens and Keenan Crane for fascinating discussions.

I have had the priviledge of working with inspirational collaborators on many parts of this work. Ulrich Pinkall and Peter Schröder developed the ideas of Chapter 2 and Isaac Chao extended and implemented them. Their examples set me on my way, and I still have so much to learn from each of them. Thank you to Nathan Litke for recruiting me to work at Rhythm and Hues studios and for spearheading our efforts to deign a practical texture mapping tool, which lead to our work presented in Chapter 3. Thanks to Liliya Kharyvech for her sage advice, the hours spent helping to render images, and for her work on variational integrators, used in Chapters 2 and 4.

My time at Caltech has brought me great friends. From my earliest days here, thanks to Mike Deceglie, Emily Warren, Caitlin Murphy, Jessica Pfeilsticker, Keenan Crane, Luke Boosey, Lauren Edgar, and Lisa Mauger. Thank you to the Caltech Alpine Club and all of the friends with whom I spent magnificent times in the mountains: Pratyush Tiwary, Jon Weissman, Tucker Jones, Nick Stadie, Joel Scheingross, Lauren Montemayor, Matt Shaner, Josh Zahl, Kedron Silsbee, Hamik Mukelyan, Danielle Bower, Dan Bower, Bill Kells, Will Raff, Nate Villaume, Chris Borstad, Chirranjeevi Gopal, Elisabeth Krause, Ana Brown, David Ayala, Tim Drayna, Sid Creutz, Greg Sadowy, Sebastian Kopf, Fabian Schmidt and Stephen Becker.

Thank you to my perennial friends: Sam Salem, Jeff Cantle, Shawn Bird, Marisa Tice, Kellie Banfield, and Dorota Korta. Thank you to my family for their unending support and inspiration: Jennifer Sanan, Kiri Sanan, Angie Sanan Hunt, and Ryan Hunt. Thank you to Clara Turner for her love and cheer; I couldn't have done it without you.

Lastly, thank you to my parents, Maeve O'Regan and David Sanan. In 1989 we emigrated from Cape Town, South Africa and by way of Texas ended up over 10,000 miles away in the San Francisco Bay Area. The decision to move so far from friends and family was largely motivated by the opportunities the U.S. held for my sister and me. I have lived a charmed life thus far, and I have you to thank for it.

Abstract

We develop new algorithms which combine the rigorous theory of mathematical elasticity with the geometric underpinnings and computational attractiveness of modern tools in geometry processing. We develop a simple elastic energy based on the Biot strain measure, which improves on state-of-the-art methods in geometry processing. We use this energy within a constrained optimization problem to, for the first time, provide surface parameterization tools which guarantee injectivity and bounded distortion, are user-directable, and which scale to large meshes. With the help of some new generalizations in the computation of matrix functions and their derivative, we extend our methods to a large class of hyperelastic stored energy functions quadratic in piecewise analytic strain measures, including the Hencky (logarithmic) strain, opening up a wide range of possibilities for robust and efficient nonlinear elastic simulation and geometry processing by elastic analogy.

Contents

List of Figures

Notation

We base our notation on that in the mathematical elasticity literature, specifically the textbook by Marsden and Hughes [39]. Boldface symbols, such as $\mathbf{C}$, represent tensors (of rank 2, except in the case of the volume form $\mathbf{dV}$). Calligraphic face symbols, such as $\mathcal{B}$, represent manifolds. Capital letters usually represent matrices.

$\langle \cdot, \cdot \rangle$	An inner product, usually a trace inner product $\langle \mathbf{A}, \mathbf{B} \rangle = \mathrm{tr}(\mathbf{A}^T \mathbf{B})$
$\mathcal{B}$	A manifold defining a body
$\mathbf{C}$	The Cauchy-Green strain tensor
$\mathbf{C}(X)$	The restriction of $\mathbf{C}$ to $T_X \mathcal{B}$
$\mathbf{dV}$	A volume form on $\mathcal{B}$
ϵ_n	A Seth-Hill strain measure, defined in (1.2)
E_n	A stored energy function quadratic in ϵ_n, defined in (1.3)
$f[x, y, \ldots]$	A divided difference of the function f
$\mathbf{F}$	The deformation gradient, $\mathbf{F} = T\phi$
$\mathbf{g}$	A metric on $\mathcal{S}$
$\mathbf{\Lambda}$	The real non-negative diagonal factor $\mathbf{\Lambda} = \mathrm{diag}(\lambda_1, \ldots \lambda_d)$ in $\mathbf{C} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T$
ϕ	A deformation $\phi : \mathcal{B} \to \mathcal{S}$.
$\mathbf{R}$	The (special) orthogonal component in the nearest rotation decomposition of $\mathbf{F} = \mathbf{R}\mathbf{Y}$
$\mathcal{S}$	A manifold into which $\mathcal{B}$ is mapped
$\mathrm{Sym}(\mathcal{B})$	The space of symmetric (2,0)-tensor fields on $\mathcal{B}$
$\mathrm{Sym}^+(\mathcal{B})$	The space of positive semidefinite, symmetric (2,0)-tensor fields on $\mathcal{B}$.
$\mathrm{Sym}^{++}(\mathcal{B})$	The space of positive definite, symmetric (2,0)-tensor fields on $\mathcal{B}$.
$T_0^2(\mathcal{B})$	The space of (2,0)-tensor fields on $\mathcal{B}$
$\mathbf{U}$	The orthogonal factor in the diagonalization $\mathbf{C} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T$
x	An element of $\mathcal{S}$, also used for a generic point in a space

X	An element of $\mathcal{B}$
$\mathbf{Y}$	The symmetric component in the nearest rotation decomposition of $\mathbf{F} = \mathbf{RY}$

Introduction

Two computational paradigms have emerged for simulation of elastic phenomena. The first, motivated by engineering, seeks to compute accurate solutions to realistic problems on relatively specific geometries. One might be concerned with the strength of a bridge or the deflection of an airplane wing under load, using models that amongst other requirements are able to approximate the behavior of a real material. The second, motivated by geometry processing, animation, and geometric mechanics, seeks, for a more general class of geometries, robust solutions, quickly computable solutions, and solutions with relevant physical invariances. In this arena, one is ultimately concerned only with elastic analogy, not direct elastic simulation; that is, one might only seek to compute a solution that 'looks like' elasticity, or might use elasticity as an analogy for a more geometrically motivated aim, such as "distributing distortion" or being "as close as possible" to an isometry or conformal map. Simulation of an engineering material or even using a discrete model which converges to a continuous model can be of secondary importance. The first paradigm is recommended by its rigorous physical basis; convergence to continuous physical theories allows simulation of real physical systems, physically intuitive behavior can be expected, and existence of solutions can be addressed in terms of continuous theories. The second paradigm is recommended by its basis in geometry, simplicity, ease of computability, and emphasis on preserving key invariances.

This thesis aims to find the intersection of these two approaches. We seek methods with rigorous foundations in both mathematical elasticity and modern differential geometry. We seek methods which are at once computationally expedient, properly invariant, and physically well-founded.

Outline

Chapter 1 presents relevant foundations in mathematical elasticity and geometry. In Chapter 2, we introduce a simple elastic energy and attendant minimization procedure which puts several highly successful 'elasticity-like' algorithms on rigorous footing, producing a widely-useful and simple-to-implement method for a range of simulation and geometry-processing tasks. In Chapter 3 , we extend this simple model to solve the challenging problem of automatically producing high-quality surface parameterizations in practical situations by solving a nonlinear, non-convex, inequality-constrained optimization problem. In Chapter 4, we leverage the theory

of matrix functions to greatly broaden the range of usable constitutive relations in our simple elastic models. In Appendix A we give more general theory on derivatives of matrix functions brought to light by these investigations.

Narratives

We offer three themes that pervade this thesis.

Intersections of Geometry and Mathematical Elasticity

By finding the points of intersection between classical mathematical elasticity and geometry processing, we discover computational methods which are at once geometrically relevant and obey a precise physical model.

Separating geometric and material nonlinearity

Much of the daunting nonlinearity in general elasticity can be mitigated by considering which nonlinearities are 'purely geometric', that is which are functions of invariant quantities under some group, namely SO(n). In Chapter 2, we deal with a materially linear, geometrically linear elastic model, and in Chapter 4 reintroduce material nonlinearity, leveraging all the geometric insight we gained once we stripped it away. A simple, but powerful identity is

$$f(g_1 x g_2) = f(x)\ \forall g_1, g_2 \in G \implies Df(g_1 x g_2)\cdot v = Df(x)\cdot(g^{-1} v g_2^{-1}) \tag{1}$$

which states that for a function f on some space for which the action of a group G, leaving f invariant, is defined, a change of coordinates can be utilized in computing the derivative. That is, f is written in terms of 'geometric' quantities. If $G = \mathrm{SE}(n)$, this is the same as f being 'objective'. While this seems trivial, perhaps, it has real computation consequences. Consider $f(x) = \sqrt{\sum x_i^2}$, mapping a vector to its Euclidean 2-norm. Differentiating directly with respect to the coordinates gives $Df(x)\cdot v = \frac{\sum x_i v_i}{\sqrt{\sum x_i^2}}$, but if we note that f is $O(n)$-invariant, we can use our geometric intuition in any frame (here that the gradient of the function is in the same direction as $||x||_2$ and has unit magnitude) to write down $Df(x)\cdot v = \frac{1}{||x||_2}\langle v, x\rangle$. This same simple invariance argument leads to drastic simplifications in the Newton systems computed in each chapter to follow. We note that the even the simple geometric intuition in the example just presented is not obvious to Automatic Differentiation (AD) approaches, mechanical derivatives taken by hand, or computer algebra systems. Thus, the derivations presented here are of real benefit in minimization of elastic energies.

Elegantly handling inverted configurations

The choice of $\mathrm{SO}(n)$-invariance has real consequences. Mathematical elasticity typically does not distinguish between $\mathrm{SO}(n)$ and $\mathrm{O}(n)$, as the improper rotations contained in $\mathrm{O}(n)$ are unphysical. However, inverted configurations are common in geometry processing - they can arise, for example from a mesh projected to a plane as an initial configuration for generating a surface parameterization. Further, allowing inverted configurations smooths the energy landscape and allows 'untangling' that would otherwise be more difficult if bodies were not allowed to travel through intermediate, inverted configurations. See Figure 3.8 and the movie in Section 2.3.2 for two examples of this untangling. For this to be effective, however, one should not blindly adapt the theory of mathematical elasticity. As we shall show, there is much to be gained from differentiating between a configuration and its inversion, despite the fact that both have the same induced metric; as we proceed, we will use tools such as the *nearest rotation decomposition* (as opposed to the polar decomposition) and the notion of *special conformal distortion* (as opposed to conformal distortion) to quantify these ideas.

Chapter 1

Foundations from Mathematical Elasticity and Geometry

We begin with a brief overview of the relevant foundations of mathematical elasticity and relevant Reimannian geometry. We base our discussions here on the complete treatment in the textbook by Marsden and Hughes[39], which also forms the basis of our notation throughout this thesis. For a complete reference concerning differential geometry in general, see the textbook by Marsden, Abraham and Ratiu [1].

1.1 Configurations

Consider a body described by a manifold $\mathcal{B}$. For our purposes, $\mathcal{B}$ will usually be a submanifold of $\mathbb{R}^n$, with $n = 2, 3$. We will consider mainly *simple bodies*, that is open subsets of $\mathbb{R}^2$ or R^3.

A *configuration* of a body is a mapping $\phi : \mathcal{B} \to \mathcal{S}$, where $\mathcal{S}$ is another manifold (typically $\mathbb{R}^d$ or $\mathbb{R}^{d+1}$). The tangent of ϕ is called the *deformation gradient* and is denoted $\mathbf{F} \doteq T\phi$. The standard term 'deformation gradient' is misleading since $\mathbf{F}$ as just defined is not a gradient. Rather, it is a map $\mathbf{F} : T\mathcal{B} \to T\mathcal{S}$ which defines a multilinear map at at each $X \in \mathcal{B}$, given by the derivative $D\phi$. This $D\phi$ is of course a linear map from $T_X\mathcal{B}$ to $T_x\mathcal{S}$, where $x \doteq \phi(X)$. This can also be interpreted as a two-point tensor field, or even as attaching a $T_x\mathcal{S}$-valued one-form to $T_X\mathcal{B}$ for each point in $X \in \mathcal{B}$. Thus, a better term than 'deformation gradient' might be 'deformation differential', if not 'deformation tangent'.

We shall write $\mathbf{F}(X)$ for the restriction of $\mathbf{F}$ to $T_X\mathcal{B}$, so $F(X) : T_X\mathcal{B} \times T_x\mathcal{S} \to \mathbb{R}$ is the multilinear map just mentioned. We shall often abuse notation slightly by equating $\mathbf{F}$ with the tensor field given by $\mathbf{F}(X)$ at each point.

A *motion* of a body is a smooth family of deformations depending on a (time) parameter t.

1.2 Configuration Space

We will let $\mathcal{C}$ denote the space of all configurations $\phi : \mathcal{B} \to \mathcal{S}$. Important subspaces include the space of all injective configurations, the space of all bounded-conformal-distortion configurations, and the space of all locally injective configurations. Computationally, we must restrict this space to a finite-dimensional subspace, which removes much of the complexity of the infinite-dimensional theory. In this thesis, we use a piecewise linear complete simplicial complex to represent $\mathcal{B}$, so the space of configurations (including degenerate ones) is $\mathbb{R}^{3N}$, where N is the number of 0-simplices (vertices). Note that a set of configurations defines a subset of the space of all possible metrics $\mathbf{C}$ on $\mathcal{B}$.

1.3 Conservation Laws

Conservation of mass and balance of linear and angular momenta lead to Cauchy's Theorem, which asserts the existence of a symmetric (2,0)-tensor σ, the *Cauchy stress tensor*, a tensor field on $\mathcal{S}$ which, when applied to a 1-form $\mathbf{n}$ representing normal directions, gives the traction (in units of force per unit area) acting on infinitesimal planes orthogonal to $\mathbf{n}$ as $\mathbf{t^b} = \sigma^{ab}\mathbf{n}_b$. To be able to discuss stress as a quantity[1] on $\mathcal{B}$, we introduce the *first Piola-Kirchhoff stress tensor*[2] $\mathbf{P}$, which operates on a 1-form on $\mathcal{B}$ to give a vector field on $\mathcal{S}$ corresponding to force per unit *undeformed* area, $\mathbf{T^a} = \mathbf{P}^{aB}\mathbf{N}_B$.

1.4 Constitutive Theory

The continuum theory above cannot be used to solve relevant problems until the stress in the body can be related to the motion, a temperature field over the body (along with heat sources or sinks), or a microstructure described as a fiber bundle over the body (which may introduce additional consistency conditions).

In its most general form, a constitutive function maps a time t and the entire history of the body (and all attached fields) to a stress tensor. A *thermoelastic constitutive function* considers only the current deformation and temperature, ignoring any rate or history effects (or additional fields).

An *elastic* body is one in which $\mathbf{P}(X)$ at a given point $X \in \mathcal{B}$ depends only on the local deformation gradient $\mathbf{F}(\mathbf{X})$. A *stress-strain law* describes this dependence. A body is called *hyperelastic* if there is a *stored energy function* W which again only depends on $X \in \mathcal{B}$ and $\mathbf{F}$ and which satisfies $\mathbf{P} = \rho_{\text{ref}}\,(\partial W/\partial \mathbf{F})$.

If W is objective (independent of choice of coordinate system), then it can only depend on $\mathbf{C} \doteq \mathbf{F}^T\mathbf{F}$, the *(right) Cauchy-Green tensor* $\mathbf{C}$ or induced metric mentioned above[3]. $\mathbf{C}$ describes how lengths and angles

[1] more specifically, a two-point tensor field

[2] Obtained from the Cauchy stress tensor by performing a Piola transform (a 'pullback with a term to account for different infinitesimal areas') on its first leg.

[3] It can be shown that $\mathbf{C} = \phi^*\mathbf{g}$, where $\mathbf{g}$ is the metric on $\mathcal{S}$ [39].

are distorted due to the deformation of the body, so it is intuitively satisfying that W can also only depend on these local, 'geometric' quantities.

Furthermore, if the material is isotropic, the dependence can only be on the invariants (or equivalently on the spectrum) of $\mathbf{C}$.

To complete a hyperelastic theory, a constitutive function is necessary to describe the form of W.

If modeling a material, it is desirable to use a form which accurately models the response of that material. If trying to work with some geometric quantity, it is desirable to choose a form which represents that quantity; see Section 1.5. In all cases simplicity and computational expediency motivate simpler measures. There is an extensive literature on constitutive inequalities, which restrict this form [59].

1.4.1 Hooke's Law and Work-conjugate stress

A simple and common constitutive relation is *Hooke's Law*, that stress varies linearly with (infinitesimal) strain. A common technique to extend the validity of this relation is to assume stress varies linearly with some strain over non-infinitesimal deformations.

A much-discussed point is how to correctly proceed from this assumption to a stored energy function. In a naive sense, since 'rate of work equals force times speed', by integrating a rate of work we should expect an elastic energy which is quadratic in the strain measure. The confusion in being more precise stems from the fact that while above we have unambiguously defined what sorts of objects $\mathbf{P}$ and $\mathbf{C}$ are, Hooke's Law does not specify what stress and strain are outside of an infinitesimal sense.

A key observation is that in the Eulerian setting (with respect to the deformed body $\phi(\mathcal{B}) \subset \mathcal{S}$), there is an unambiguous expression for rate of work per unit volume $\dot{\omega}$ corresponding to a motion $\phi(X,t) = \phi_t(X)$,

$$\dot{\omega} = \rho\sigma : \mathbf{d}$$

where $\mathbf{d} = 2(\phi_t)_*(\dot{\mathbf{C}})$, dots denote derivatives with respect to t, and ρ is a density (with units of mass over volume).

The notion of *work conjugate stress* to a given measure of strain is now useful. Briefly, if ϵ is a strain measure, its conjugate stress $\mathbf{S}$ satisfies

$$\dot{\omega} = \rho\mathbf{S} : \dot{\epsilon} \tag{1.1}$$

The existence and form of conjugate stresses to various strain measures has been well-studied, and here we refer the reader to the literature [22]

For the purposes of this thesis, the existence of a conjugate stress is subsidiary to the existence of a stored energy function quadratic in a given strain measure, or more generally, a norm on the space of Reimannian metrics on $\mathcal{B}$. This is discussed further is Section 1.5.

This is related to the tension between linearized and general developments of mathematical elasticity. Hooke's law is a convenient hypothesis in many situations (and we make extensive use of it in chapter 4), but determining the appropriate stress conjugate to a particular strain measure is not particularly important in the hyperelastic setting.

1.4.2 Tensor Norms

The stored energy functions used later on take the form of norms of tensor fields $A \in T_0^2(\mathcal{B})$, integrated norms of tensors defined on tangent spaces of $\mathcal{B}$.

Throughout this work we shall make repeated use of the (semi-)norm $||\cdot||_{\alpha,\beta}$, where

$$||A||_{\alpha,\beta} = ||\alpha PA + \beta(I-P)A||$$

and P is a projection. The norm on the righthand side should be invariant under the same group as P. While interesting generalizations are possible [38], we will consider only the case where $PA = (\frac{\operatorname{tr} A}{d})I$, which is invariant under the action of $\mathrm{SO}(d)$ for d the dimension of A. Then, in terms of Lamé parameters (one of several choices of a pair of two isotropic material parameters), $\alpha = 3\lambda + 2\mu$ and $\beta = 2\mu$ [38].If PA and $(I-P)A$ commute and are orthogonal wrt an inner product which induces $||\cdot||$, then we can also write

$$||A||_{\alpha,\beta} = (\alpha^2 - \beta^2)||PA|| + \beta^2||(I-P)A||^2$$

1.4.3 The Nearest Rotation Decomposition

As described in Section 1.4, objective and isotropic constitutive theories can only depend on the invariants of $\mathbf{C}$, (or equivalently be symmetric functions of its eigenvalues). These are invariants with respect to action of $\mathrm{O}(n)$, the orthogonal group, and thus obviously also invariants with respect to the action of $\mathrm{SO}(n) \subsetneq \mathrm{O}(n)$, the special orthogonal group.

An important point is that physical elasticity need not be concerned with the distinction between $\mathrm{O}(n)$ and $\mathrm{SO}(n)$, since orientation-reversing deformations are physically impossible.

Viewed another way, the entire theory must be coordinate free, or covariant. However, insisting that it be invariant with respect to $\mathrm{O}(n)$ is not justified. Rather (we shall argue) it is often preferable to relax this invariance to $\mathrm{SO}(n)$ which allows locally 'inverted' configurations to be dealt with elegantly.

One can define the (right) polar decomposition $\mathbf{F} = \tilde{\mathbf{R}}\tilde{\mathbf{Y}}$, with $\tilde{\mathbf{R}} \in \mathrm{O}(n)$ and $\tilde{\mathbf{Y}} \in \mathrm{Sym}^+(\mathcal{B})$. This implies $\mathbf{C} = \tilde{\mathbf{Y}}^2$, or $\mathbf{C}^{1/2} = \tilde{\mathbf{Y}}$ for a the principal branch of the square root matrix function (see Higham's textbook [19] for more on matrix functions).

If, instead, we use a non-principal square root of $\mathbf{C}$, in particular one which uses the *negative* square root

of the smallest eigenvalue of $\mathbf{C}$ (the squared smallest singular value of $\tilde{\mathbf{Y}}$) whenever $\det \mathbf{C} < 0$, we obtain a new decomposition $\mathbf{F} = \mathbf{RY}$, where $\mathbf{R} \in \mathrm{SO}(n)$ and $\mathbf{Y} \in \mathrm{Sym}(\mathcal{B})$. We call this decomposition the *nearest rotation decomposition* and note that it agrees with the polar decomposition when $\det \mathbf{C} \geq 0$.

The symmetric factors $\tilde{\mathbf{Y}}$ and $\mathbf{Y}$ are related by considering their diagonalizations. $\tilde{\mathbf{Y}} = U\tilde{\Sigma}U^T$ and $\mathbf{Y} = U\Sigma U^T$, where $\tilde{\Sigma} = \mathrm{diag}(\sigma_1, ..., \sigma_{n-1}, \sigma_n)$ and $\Sigma = \mathrm{diag}(\sigma_1, ..., \sigma_{n-1}, \mathrm{sgn}(\det \mathbf{F})\sigma_n)$.

1.4.4 Strain Measures

We shall take the term *strain measure* to mean a matrix function[4] of $\mathbf{C}$. Matrix functions are discussed further in Section 4.3, but are simply defined for symmetry operators; if $\mathbf{C}$ can be unitarily diagonalized as $\mathbf{U\Sigma U}^T$, then $f(\mathbf{C}) = \mathbf{U}f(\mathbf{\Sigma})\mathbf{U}^T$, where $f(\mathbf{\Sigma})$ applies a given analytic function $f : \mathbb{C} \to \mathbb{C}$ to the diagonal entries of $\mathbf{\Sigma}$.

A useful and well-studied family of strain measures are the Seth-Hill measures [54, 22], a family of functions of $\mathbf{C}$ depending on a parameter $n \in \mathbb{Z}$, defined as

$$\epsilon_n(\mathbf{C}) = \begin{cases} \frac{1}{n}(\mathbf{C}^{n/2} - \mathbf{I}) & n \neq 0 \\ \frac{1}{2}\log \mathbf{C} & n = 0 \end{cases} \tag{1.2}$$

These are members of a larger class of functions $f(\mathbf{C})$ with the properties $f(1) = 0, f'(1) = 1/2$. Thus, these measures all have the same linearization around $\mathbf{C} = \mathbf{I}$, namely the canonical measure of strain familiar from the study of linear elasticity. For odd n, $\mathbf{C}^{1/2}$ can be interpreted as any square root including $\mathbf{Y}$ or $\tilde{\mathbf{Y}}$, but we advocate the nearest rotation factor $\mathbf{Y}$ as being preferable for the applications to follow and unless otherwise stated, will always make this choice.

We will frequently make use of stored energy functions of the form

$$E_n(\phi) \doteq \frac{1}{2}\int_{\mathcal{B}} ||\epsilon_n(\mathbf{C}_\phi)||^2_{\alpha,\beta}\mathbf{dV} \tag{1.3}$$

where the subscript on $\mathbf{C}_\phi$ makes explicit the dependence on the configuration ϕ.

1.5 Geometric Quantities

We now change course and consider things from a more geometric point of view.

A first, fundamental observation is that objective elastic theories heavily involve minimizing functions of $\mathbf{C}$, where $\mathbf{C}$ is in fact the *induced metric of a mapping*. This can be thought of as the pullback by the deformation ϕ of the metric $\mathbf{g}$ on $\mathcal{S}$.

[4] possibly a non-principal one

The metric tensor is the fundamental object in Reimannian geometry[14], which in turn is used to understand many topics in physics and mathematics. From this point of view, mathematical elasticity can be see as solving a particular optimization problem involving functions of metrics. The deformation is secondary to the metric itself, and the physical basis of a constitutive function is secondary to the metric it induces on the space of metrics itself.

The existence of stress is arguably the fundamental assumption of continuum mechanics, but here it arises merely as the conjugate quantity to some strain measure, which is the important geometric quantity. This is the situation in which forces are conjugate to displacements which are the actual objects of study, most familiar when forces are considered as Lagrange multipliers - their value is unimportant, but the value of their conjugate quantity (displacement) is.

A key observation is that an isometry is necessarily a local minimizer of a stored energy function, under the uncontroversial assumption that the identity configuration corresponds to minimal stored energy. From a physical point of view, these stored energy functions need not make the distinction between $\mathrm{O}(n)$ and $\mathrm{SO}(n)$, as orientation-reversing deformations are not physically plausible. However, we will allow for such local inversions, in which case we should enforce that only orientation-preserving isometries are minimizers of the energy.

Chapter 2

A Simple Geometric Model for Elastic Deformations

The results presented in this chapter are based on the publication *A Simple Geometric Model for Elastic Deformations* by Chao, Pinkall, Sanan and Schröder [9].

Geometry processing is largely concerned with preserving 'shape', a vague term for properties of a body that are invariant under rigid body motions, that is under the action of the special Euclidean group $\mathrm{SE}(n)$. Locally, these properties are characterized by a metric. Given a set of constraints on a body, we cannot in general exactly preserve the metric, so we seek to minimize some notion 'distance from rigidity'. This could be though of as norm on the space of metrics which can be assigned to the body, something that 'looks like elasticity' is one choice of such a norm. If we think of this norm as arising from a metric on the space of metrics, we can also try and compute geodesics through configuration space $\mathcal{C}$.

This is the basis of the methods presented in this chapter, which has been used to great effect in geometry processing. It turns out, however, that this formulation can employ an actual formulation of mathematical elasticity to induce a norm on the space of metrics, and thus satisfies a precise physical analogy which, amongst other benefits such as ease of introduction of physically valid material parameters, asserts the existence of a continuous theory which the method discretizes and a well-defined elastic energy to be minimized.

More specifically, we use the Biot strain measure $\epsilon_1 = \mathbf{C}^{1/2} - \mathbf{I}$ [7], where we interpret the matrix square root as the non-principal matrix function which produces the nearest rotation factor $\mathbf{Y}$. Because $Y - I$ is lower order in the state variables, it is computationally far more attractive. Compare, *e.g.*, the efforts to"reign in" the computational complexity of the standard Green-Lagrange strain undertaken in [31].

Geometrically speaking, this strain measure can be interpreted as distance between the differential of a deformation and the rotation group, an observations which enables very efficient computation of the variations of the energy.

Thus, the strain energy we study here is at once a geometrically useful, easily computable, and robust

"elasticity-like" tool, and simultaneously a valid model of nonlinear mathematical elastic materials.

2.1 The Elastic Energy

Given a smooth map $\phi : \mathcal{B} \to \mathcal{S}$, $\phi(X) = x$, consider the energy

$$\tfrac{1}{2}\int_{\mathcal{B}} \operatorname{dist}(\mathbf{F}, \mathrm{SO}(n))^2 \, \mathbf{dV} = \tfrac{1}{2}\int_{\mathcal{B}} \min_{\mathbf{R}\in \mathrm{SO}(n)} ||\mathbf{F} - \mathbf{R}||^2 \mathbf{dV}. \tag{2.1}$$

At a point $X \in \mathcal{B}$, the integrand measures the distance between the deformation differential, $\mathbf{F}(X) \doteq D\phi(X)$, and the nearest rotation, thus characterizing how far ϕ is from an isometry.

The norm $||\cdot||$ in (2.1) is any $\mathrm{SO}(n)$-invariant norm on $T^2_0(\mathcal{B})$, but we shall typically consider the (semi-)norm $||\cdot||_{\alpha,\beta}$ discussed in Section 1.4.2.

If we choose $\alpha = 0$ (thus considering an integrated seminorm), the energy penalizes deviation from conformality. If we choose $\beta = 0$ (again considering a seminorm), the energy penalizes deviation from volume preservation.

More generally, we consider

$$E_1(\phi) \doteq \tfrac{1}{2}\int_{\mathcal{B}} \rho ||\mathbf{F}_\phi - \mathbf{R}_\phi||^2_{\alpha,\beta} \mathbf{dV} \tag{2.2}$$

where ρ is a non-negative scalar density field. From Section 1.4.1, note immediately that this equivalent to

$$\frac{1}{2}\int_{\mathcal{B}} \rho ||\tau_1 : \epsilon_1||^2_{\alpha,\beta} \mathbf{dV}$$

2.2 Minimizing the Energy

A minimizer of $E_1(\phi)$ subject to boundary conditions describes a solution to the elastostatics problem. Such a minimizer is characterized by vanishing variations $\delta_\psi E_1(\phi) \doteq \frac{d}{d\epsilon}|_{\epsilon=0} E_1(\phi + \epsilon\psi) = 0$. Let ψ be an arbitrary admissible variation. Then,

$$\delta_\psi E_1(\phi) = \int_{\mathcal{B}} \langle \mathbf{F}_\phi - \mathbf{F}_\psi \mathbf{R}, \mathbf{F}_\phi - \mathbf{R}\rangle = \int_{\mathcal{B}} \langle \mathbf{F}_\psi, \mathbf{F}_\phi - \mathbf{R}_\phi\rangle, \tag{2.3}$$

where $\langle \mathbf{A}, \mathbf{B}\rangle = \mathrm{tr}(\mathbf{A}^T\mathbf{B})$ denotes the standard inner product between linear maps and $\mathbf{R}_\phi \in \mathrm{SO}(n)$ the minimizer of the squared distance (dropping explicit mention of its dependence on $\mathbf{F}_\phi$). We have used $\mathbf{F}_\phi - \mathbf{R}_\phi \perp \delta_\psi \mathbf{R}_\phi$, which follows from $\mathbf{R}$ being critical with respect to the squared distance; thus, no derivatives of $\mathbf{R}$ appear in the gradient of E_1.

This observation is of critical importance to the effectiveness of the method.

Due to the dependence of $\mathbf{R} - \phi$ on $\mathbf{F}_\phi$, the Euler-Lagrange equation is a non-linear Poisson problem

$$\Delta\phi = \operatorname{div} R.$$

The energy Hessian follows from taking a further variation ζ

$$\delta^2_{\psi,\zeta} E_1(\phi) = \int_{\mathcal{B}} \langle \mathbf{F}_\psi, \mathbf{F}_\zeta \rangle - \int_{\mathcal{B}} \langle \mathbf{F}_\psi, \delta_\zeta \mathbf{R}_\phi \rangle.$$

The first term is the standard Laplace-Beltrami operator which does not depend on ϕ, while the second term varies with ϕ through the dependence of $\mathbf{R}_\phi$ on $\mathbf{F}_\phi$. Here variations of $\mathbf{R}_\phi$ *do* enter since they are in general not orthogonal to $\mathbf{F}_\psi$. This observation is also important - consideration of these variations distinguish this method from a corotational method, as discussed further in Section 2.5.1.

Note the extension to elastodynamics in Section 2.5.

2.3 Piecewise Linear Finite Element Discretization

Let $\mathcal{B}$ be a 2- or 3-manifold given as a complete simplicial complex with an embedding defined by positions in $X_i \in \mathcal{S} = \mathbb{R}^d$, $d \in \{2, 3\}$. The deformation function ϕ is given as a piecewise affine mapping from X_i to new positions x_i. The integral over $\mathcal{B}$ becomes a sum over triangles or tetrahedra. The corresponding discrete energy in 2D is the well known piecewise linear Dirichlet energy [49]

$$\tfrac{1}{2} \int_{p_{ijk}} |\mathbf{F}_\phi - \mathbf{R}_\phi|^2 = \tfrac{1}{4} \sum_{X_{ab} \in \{i,j,k\}} \cot \alpha^c_{ab} |x_{ab} - \mathbf{R}^{ijk}_\phi X_{ab}|^2,$$

where α^c_{ab} denotes the angle at c opposite edge X_{ab}; $X_{ab} \doteq X_b - X_a$, the directed edge from X_a to X_b; and x_{ab} its image under $\mathbf{F}_\phi$. Note that $\mathbf{R}^{ijk}_\phi$ is constant per triangle since $\mathbf{F}_\phi$ is. Thus, for computational purposes these tensor fields can be represented with sets of 2×2 or 3×3 matrices.

The corresponding cotan formula for a tetrahedron [41] is

$$\tfrac{1}{2} \int_{X_{ijkl}} |\mathbf{F}_\phi - \mathbf{R}_\phi|^2 = \tfrac{1}{12} \sum_{e_{ab} \in \{i,j,k,l\}} |X_{cd}| \cot \alpha^{cd}_{ab} |x_{ab} - \mathbf{R}^{ijkl}_\phi X_{ab}|^2,$$

with α^{cd}_{ab} the dihedral angle opposite e_{ab}, *i.e.*, at edge e_{cd} and $\mathbf{R}^{ijkl}_\phi$ constant per tetrahedron.

The gradient and Hessian of the discrete energy can be used to minimize the energy using Newton's method.

2.3.1 The First Discrete Variation

Obtaining the gradient is just a matter of differentiating the above expressions with respect to the x_i. For a single triangle $\{i, j, k\}$, the configuration of which defines an affine mapping ϕ, we obtain

$$\delta_i E_1(\phi) = \tfrac{1}{2}(\cot \alpha_{ij}^k x_{ji} + \cot \alpha_{ik}^j x_{ki}) - \mathbf{R}_\phi^{ijk} \tfrac{1}{2}(\cot \alpha_{ij}^k X_{ji} + \cot \alpha_{ik}^j X_{ki})$$

The first term represents a summand in the Laplace-Beltrami operator applied to the x variables, while the second term is the area gradient of the original triangle X_{ijk} with respect to vertex i rotated by $\mathbf{R}_\phi^{ijk}$. As in the continuous setting (Eq. (2.3)), criticality ensures that no derivatives of R^{ijk} appear in the discrete setting[1]. These expressions are summed for all triangles incident to vertex i. The sum of the rotated area gradients is the divergence of the rotations incident on vertex i.

The expression for a tetrahedron is entirely analogous with terms evaluating the Laplacian of x and the *volume* gradient of the tetrahedron X_{ijkl}, with respect to vertex i, rotated by $\mathbf{R}_\phi^{ijkl}$. Fully detailed expressions are found in Appendix B.1.

2.3.2 Computing the Nearest Rotation

For $\mathbf{F}_\phi(X)$ with positive determinant, $\mathbf{R}_\phi(X)$ is most efficiently found by computing the right polar decomposition $\mathbf{F}_\phi = \mathbf{R}_\phi \mathbf{Y}_\phi$ with Newton's algorithm [18, p. 1168] (with warm start). In case of inverted elements ($\det \mathbf{F}_\phi < 0$) the polar decomposition returns an *orientation reversing* orthogonal transformation instead of the closest rotation. In these cases, as well as for singular $\mathbf{F}_\phi$, the closest rotation (or *a* closest rotation for rank 0 or 1 in 3D resp. 0 in 2D) is given as a simple function of the SVD [45, Lemma 1]. The more costly SVD procedure is invoked only for $\mathbf{F}_\phi$ with non-positive determinant. (In practice we use $\det \mathbf{F}_\phi < \epsilon$ which also covers all rank deficient cases.)

Note also that in the 2-dimensional case, the nearest rotation can be computed easily using an explicit expression, as described further in Algorithm 2 in Chapter 3.

Comment on inverted elements The distance $|\mathbf{F}_\phi - \mathbf{R}_\phi|^2$ is always continuous, *even* as the element inverts. (Additionally it is smooth if $\det \mathbf{F}_\phi > 0$ or $\det \mathbf{F}_\phi \leq 0$ and the smallest singular value is unique.) The energy only increases for an inverted element since the nearest rotation is further away. Compare this to fourth order measures such as $|\mathbf{C} - \mathbf{I}I|^2$. In that case a mirrored tetrahedron $\mathbf{F}_\phi = -\mathbf{I}$ has zero strain. This has led to the development of, at times elaborate, measures to deal with inverted elements in elasticity (see, *e.g.*, [24, 53]).

[1] Equivalently, this behavior can be predicted from (1).

Comment on performance One could use an SVD in all cases, but we do not recommend this for performance reasons. Over a broad set of simulations we found that 10% of the entire runtime is spent computing $\mathbf{R}_\phi$. Within the $\mathbf{R}_\phi$ routine only about 1-3% of all calls invoke the SVD, but those few calls consume 10-20% of the time in the $\mathbf{R}_\phi$ routine. Because the polar decomposition (with warm starts) is so much cheaper than the SVD, using the SVD every time would increase overall runtime by 50% or more.

The resulting algorithm is quite robust as demonstrated by a "torture test:" the tetrahedralized Dragon model, scaled by $-3/4$, is used as an initial configuration. In this configuration every tetrahedron is inverted, yet the original shape is quickly recovered without any adverse effects (see the attached movie in the pdf version of this document).

2.3.3 The Second Discrete Variation

A second variation produces the standard cotan-Laplace operator[49] —independent of ϕ—and an additional term which depends on ϕ (through $\mathbf{Y}_\phi$ and $\mathbf{R}_\phi$) and is constant per triangle or tetrahedron

$$\langle \mathbf{F}_\psi, \delta_\zeta \mathbf{R}_\phi \rangle = 4 \left\langle \mathrm{X}(\mathbf{R}_\phi^T \mathbf{F}_\psi), \mathrm{X}(\mathbf{R}_\phi^T \mathbf{F}_\zeta) \right\rangle_W , \tag{2.4}$$

a weighted (by W) scalar product of the anti-symmetric part (X(.) denotes extraction of the anti-symmetric part of a matrix) of the backward rotated differentials of the variations ψ and ζ. In practice this expression amounts to a linear combination of precomputed outer product matrices multiplied by R^T and the weight matrix, which is a simple linear function of $\mathbf{Y}_\phi$, as follows.

2.3.3.1 Computing the Second Variation

To compute the second variation of Eq. (2.1) we need a formula for $\langle \mathbf{F}_\psi, \delta_\zeta \mathbf{R}_\phi \rangle = \langle \mathbf{R}_\phi^T \mathbf{F}_\psi, \mathbf{R}_\phi^T \delta_\zeta \mathbf{R}_\phi \rangle$. Noting that $\mathbf{R}_\phi^T \delta_\zeta \mathbf{R}_\phi$ is anti-symmetric this reduces to $\langle \mathrm{X}(\mathbf{R}_\phi^T \mathbf{F}_\psi), \mathrm{X}(\mathbf{R}_\phi^T \delta_\zeta \mathbf{R}_\phi) \rangle$. $\mathrm{X}(\mathbf{R}_\phi^T \mathbf{F}_\psi)$ can be computed directly from knowledge of the back rotated differential $\mathbf{F}_\psi$ of the variation ψ. All that is needed is an expression for $\mathrm{X}(\mathbf{R}_\phi^T \delta_\zeta \mathbf{R}_\phi)$.

With $\mathbf{F}_\phi = \mathbf{R}_\phi \mathbf{Y}_\phi$ we have $\mathbf{F}_\zeta = \delta_\zeta \mathbf{F}_\phi = \delta_\zeta \mathbf{R}_\phi \mathbf{Y}_\phi + \mathbf{R}_\phi \delta_\zeta \mathbf{Y}_\phi$ and thus $\mathbf{R}_\phi^T \mathbf{F}_\zeta = \mathbf{R}_\phi^T \delta_\zeta \mathbf{R}_\phi \mathbf{Y}_\phi + \delta_\zeta \mathbf{Y}_\phi$. In order to get a useful expression for $\mathbf{R}_\phi^T \delta_\zeta \mathbf{R}_\phi$ we consider the representative of the anti-symmetric part of $\mathbf{R}_\phi^T \mathbf{F}_\zeta$

$$\mathrm{X}(\mathbf{R}_\phi^T \mathbf{F}_\zeta) = \mathrm{X}(\mathbf{R}_\phi^T \delta_\zeta \mathbf{R}_\phi \mathbf{Y}_\phi) = \tfrac{1}{2} \mathrm{X}(\{\mathbf{R}_\phi^T \delta_\zeta \mathbf{R}_\phi, \mathbf{Y}_\phi\}),$$

where we used the symbol $\{., \mathbf{Y}_\phi\}$ for the anti-commutator $\{\mathbf{A}, \mathbf{Y}_\phi\} = \mathbf{A}\mathbf{Y}_\phi + \mathbf{Y}_\phi \mathbf{A}$, dropped $\delta_\zeta \mathbf{Y}_\phi$ since it is symmetric, and exploited the anti-symmetry of $\mathbf{R}_\phi^T \delta_\zeta \mathbf{R}_\phi$ for the second equality.

Hence, if we can invert the anti-commutator on its first argument we get an expression for $\mathbf{R}_\phi^T \delta_\zeta \mathbf{R}_\phi$. Indeed, the anti-commutator $\{., \mathbf{Y}_\phi\}$ has an inverse on the anti-symmetric matrices if $\mathbf{Y}_\phi$ is positive semi-

definite and no worse than rank deficient by one. Denote this inverse by $W := \{., \mathbf{Y}_\phi\}^{-1}$. Treating it as a map between *representatives* of anti-symmetric matrices we get

$$\mathrm{X}(\mathbf{R}_\phi^T \delta_\zeta \mathbf{R}_\phi) = 2W(\mathrm{X}(\mathbf{R}_\phi^T \mathbf{F}_\zeta))$$

In 2D, $W = \mathrm{tr}(\mathbf{Y}_\phi)^{-1}$ and in 3D, $W = (\mathrm{tr}(\mathbf{Y}_\phi)I - \mathbf{Y}_\phi)^{-1}$. The former follows from simply writing out the expressions while the latter is easily checked for diagonal $\mathbf{Y}_\phi$ with at least two positive diagonal entries. Validity for general symmetric $\mathbf{Y}_\phi$ then follows after multiplying the left and right sides with the appropriate coordinate transformations.

This gives the final representation

$$\langle \mathbf{F}_\psi, \delta_\zeta \mathbf{R}_\phi \rangle = 4 \left\langle \mathrm{X}(\mathbf{R}_\phi^T \mathbf{F}_\psi), \mathrm{X}(\mathbf{R}_\phi^T \mathbf{F}_\zeta) \right\rangle_W ,$$

where we treat the intervening W as a scalar product weighting.

Full expressions can be found in the C++ code portions in Appendix B.1, and relevant identities can be found in Appendix C. Note that the variation $\delta_\zeta \mathbf{R}_\phi$ is not defined when W is undefined, so in practice we set it to zero.

2.4 Elasticity Simulation

To get a material model with Lamé parameters we note that the difference $\mathbf{F}_\phi - \mathbf{R}_\phi = \mathbf{R}_\phi(\mathbf{Y}\phi - \mathbf{I})$ can be further decomposed orthogonally into a trace-free part and a multiple of the identity

$$\mathbf{Y} - \mathbf{I} = \left(\mathbf{Y} - \tfrac{\mathrm{tr}(\mathbf{Y})}{n}\mathbf{I}\right) + \left(\tfrac{\mathrm{tr}(\mathbf{Y})}{n} - 1\right)\mathbf{I} =: \bar{\mathbf{Y}} + y\mathbf{I}$$

with $n = 2, 3$ the dimension of $\mathcal{B}$. Each can now be weighted independently with $\alpha \geq 0$ (bulk deformation) and $\beta \geq 0$ (shear deformation). This amounts to using a tensor norm as described in Section 1.4.2, and gives the energy (2.2).

The energy density for an isotropic Hookean material, $\langle \sigma, \epsilon \rangle$ with $\sigma = 2\mu\epsilon + \lambda\,\mathrm{tr}(\epsilon)I$ and Lamé parameters μ and λ maps to $E_1(f)$ for $\epsilon = \epsilon_1 = \mathbf{Y} - \mathbf{I}$, $2\mu = \beta^2$, and $\lambda = (\alpha^2 - \beta^2)/n$. With $E_1(f)$ we have a geometrically non-linear elastic potential energy with standard material parameters and a minimization solver can find static equilibrium configurations of elastic bodies.

Since $\mathbf{Y} - \mathbf{I} = \epsilon_1$, we have precisely recovered a Hookean constitutive relationship in which we presume that a stored energy function quadratic in ϵ_1 exists.

The corresponding gradient and Hessian expressions are included in Appendix B.1.

Figure 2.1 demonstrates deformations and the influence of the α and β parameters.

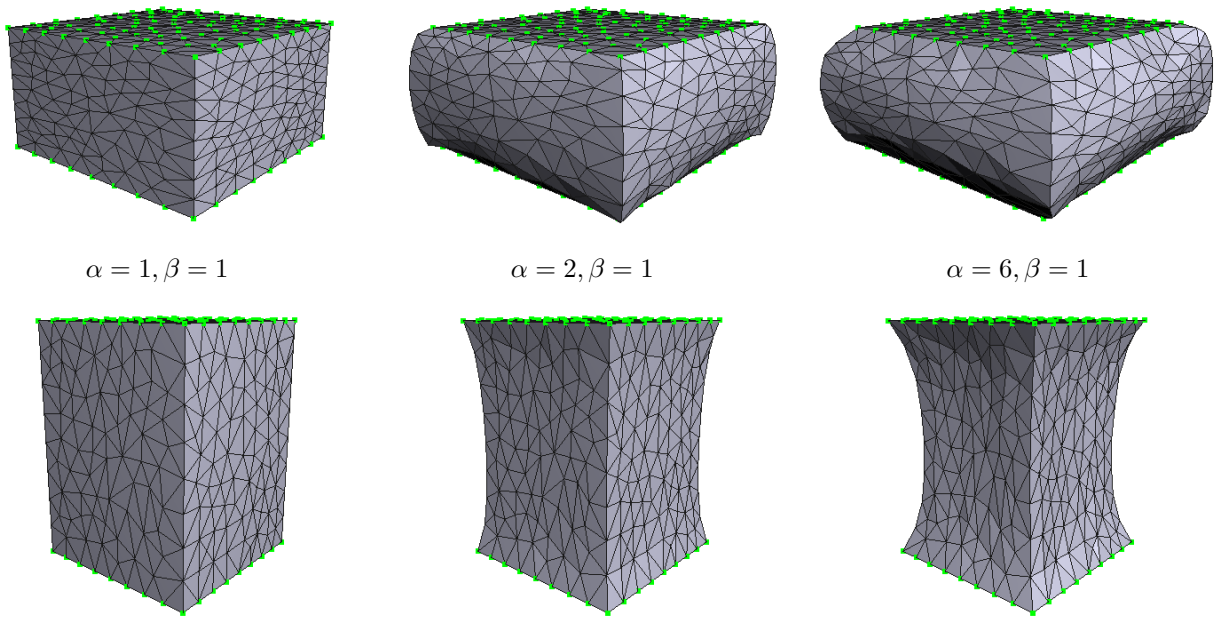


Figure 2.1: Examples of tetrahedral mesh compression (down to 53% height) and expansion (up 135% height) for different α parameters.

2.5 Elastodynamics Problems

Dynamics simulations are a straightforward extension of the static elasticity case and many avenues are possible. For our experiments we implemented the *fully variational integrator* (FVI) of [28], based on the stationary Hamilton-Pontryagin principle. We chose it because it exhibits—like all variational integrators [40]—excellent long term energy stability and exact preservation of momenta, even for large time steps. From a practical point of view an FVI has the significant benefit that it converts the usual non-linear root finding problem in the time stepper into an energy minimization problem. *And* this is achieved with only a minor modification to the energy minimization code we already have!

Assume per vertex momentum (p^k), position (q^k), and velocity variables (v^k) indexed by discrete time $k \geq 0$. Define an ancillary energy

$$\varepsilon(q) := \tfrac{2}{h^2}(q - q^k)^T M(q - q^k) + W(q) - \tfrac{2}{h}(p^k)^T(q - q^k),$$

with h denoting the time step size, M the mass matrix, and $W(.) = E_1(.)$ the elastic potential of the body. Define $q^* = \operatorname{argmin}_q \varepsilon(q)$, then the transition $k \to k+1$ proceeds as

$$\begin{aligned} v^{k+1} &= \tfrac{2}{h}(q^* - q^k) \\ q^{k+1} &= q^k + hv^{k+1} \\ p^{k+1} &= Mv^{k+1} - \tfrac{h}{2}\operatorname{grad} W(q^k + \tfrac{1}{2}hv^{k+1}) \end{aligned}$$

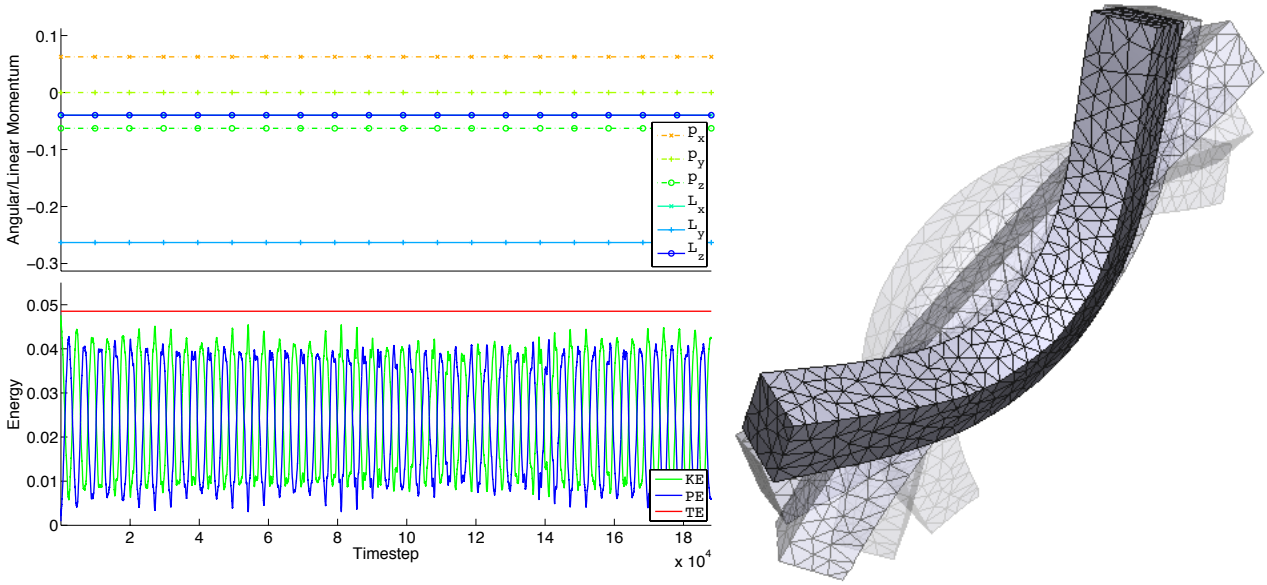


Figure 2.2: Tracking the momentum and energy of an elastic body with some initial motion and bending deformation. Momentum and energy are tracked precisely even for very long simulation runs.

Minimization of $\varepsilon(q)$ requires its gradient and Hessian. Both are simple modifications of the corresponding expressions for $W(q) = E_1(q)$

$$\begin{aligned} \operatorname{grad} \varepsilon(q) &= \operatorname{grad} W(q) + \tfrac{4}{h^2} M(q - q^k) - \tfrac{2}{h} p^k \\ \operatorname{Hess} \varepsilon(q) &= \operatorname{Hess} W(q) + \tfrac{4}{h^2} M. \end{aligned}$$

In this manner the existing minimization code needs only a small modification to serve as a time stepper for dynamics simulations. To complete the system one would likely want to add external forces and damping and we refer the interested reader to [28] for the details. There one also finds the derivation of the energy minimization time stepper.

Figure 2.2 demonstrates the exact momentum conservation and long term stable energy behavior which distinguish these integrators. A more complex example is given in Figure 2.3.

Shape matching approaches to qualitative physics simulations [43] use reasoning reminiscent of our energy. Considering groups of nearby points, they use a shape matching metaphor to determine the best fit rigid transform of the group of points from initial to deformed configuration. The resulting rotation $\tilde{\mathbf{R}}$ arises from the polar decomposition of $\tilde{\mathbf{R}}\tilde{Y} = \mathbf{F}_\phi \mathbf{I}_m$ where $\mathbf{I}_m$ is the inertia tensor of the point set (up to a scale factor). This additional factor of $\mathbf{I}_m$ makes the resulting $\tilde{\mathbf{R}}$ dependent on the shape of the group of points. As a consequence the material behavior may become anisotropic if the groups of points are not isotropic in shape. No continuous energy is given.

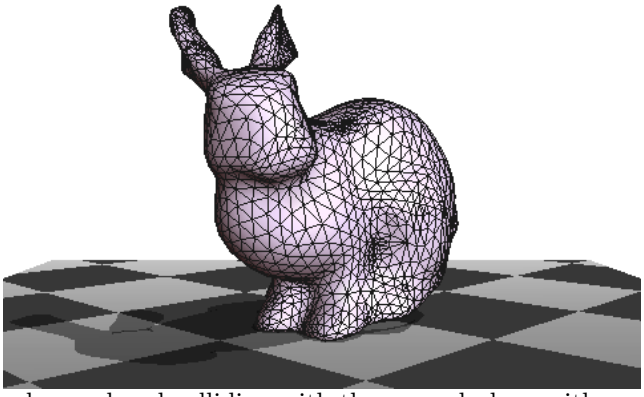


Figure 2.3: A Bunny being dropped and colliding with the ground plane with no damping whatsoever. The energy is perfectly preserved. (See also the embedded movie in the pdf version of this document.)

2.5.1 Relationship to Co-Rotational Methods

Per element rotations applied to $\mathbf{F}_\phi$ appear also in co-rotational methods [51]. These have become popular in computer graphics since [42, 15] as a post facto fix for the disturbing artifacts due to the linearized Green-Lagrange strain $2\epsilon_2 = \mathbf{C} - \mathbf{I} = (\mathbf{F}^T\mathbf{F} - \mathbf{I}) \simeq (\mathbf{F}^T + \mathbf{F})/2 - \mathbf{I}$ when large deformations or rotations occur. We emphasize that this is quite different from our non-linear setting which uses $\mathbf{R}_\phi$ to measure the distance from $\mathbf{F}_\phi$ to an isometry. The following experiment demonstrates that this difference is not mere semantics.

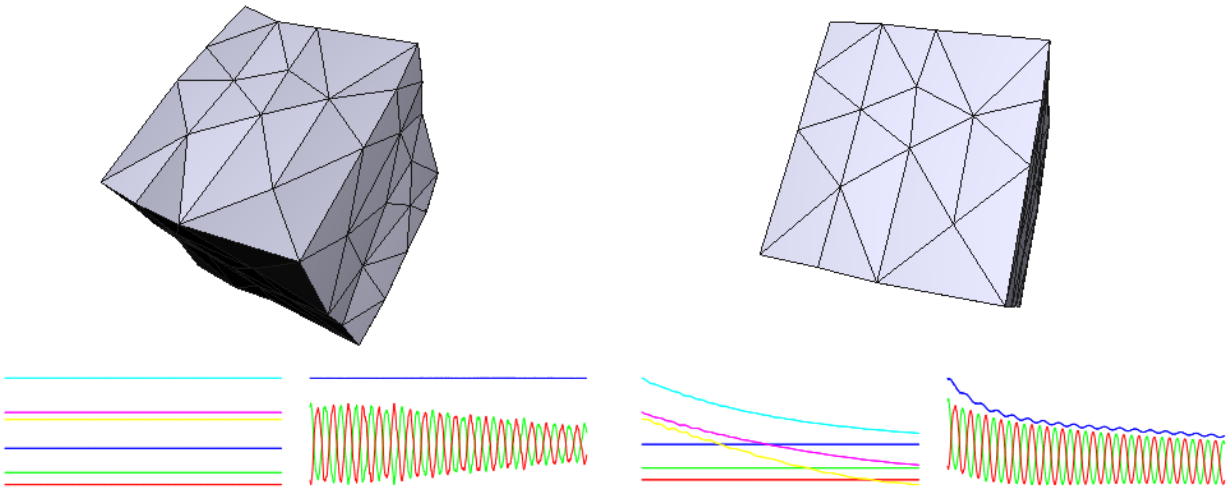


Figure 2.4: Comparison of our non-linear energy (left) with a co-rotational method (right). Linear and angular momenta (6 lines) and energy (kinetic, potential, total) are shown in the plots. For our non-linear energy, momenta are exactly preserved and total energy stays within a small constant band. Fixing rotations at the beginning of each time step (co-rotational) destroys these system invariants. See also the embedded movie in the pdf version of this document.

We will fix the rotations at the beginning of each time step. Now the backward rotated $\mathbf{R}_\phi^T\mathbf{F}_\phi$ at each element gives the strain used in a co-rotational method. The time stepper energy minimization is quadratic

and a single solve moves from $k \to k+1$ (the integration is still a 2^{nd} order accurate implicit method). Figure 2.4 shows what happens in a comparison with the non-linear energy. An elastic body in 3D is subject to some initial momentum and a twist/compression without any external forces. Time step size was identical in both cases. At the bottom of each image are graphs of momenta (left) and energy (right). For this simulation they should stay constant and they do when using the non-linear energy in contrast to the co-rotational method.

Numerical vs. variational damping The numerical damping in the co-rotational method causes not only loss of energy and momentum invariants, but also depends on time step size. This is illustrated in a movie embedded in the pdf version of this document showing side by side runs with stepsizes of 0.01 and 0.002 (the latter subsampled by 5 for display). With the rotation fixed at the beginning of the time step, the state determined for the end of the time step uses the wrong (old) rotation, assigning potential energy to the incremental rotation during the time step. When the rotation is then reset at the beginning of the next time step, this energy is lost. Numerical experiments easily verify this lossage mechanism. As illustrated in the video, smaller time steps can lessen, but never fix this phenomenon. There is always some potential energy, corresponding to a small but finite rotation, which is discarded.

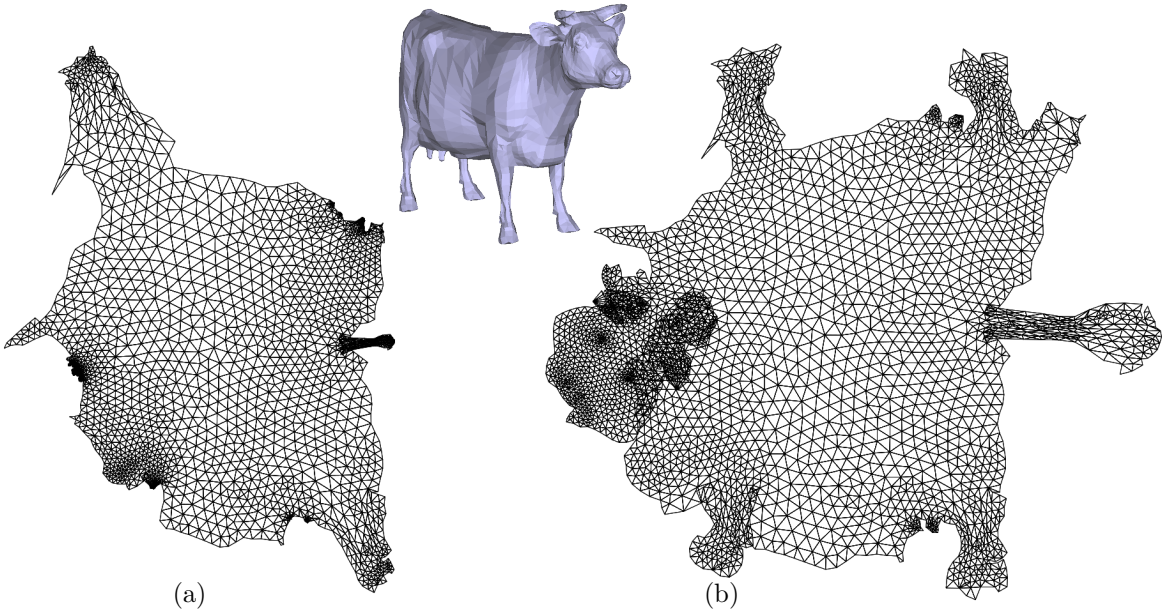
In application practice one generally wishes to have damping, but it should not depend on the time step size. This is easily achieved with variational damping (for details see [28]) as demonstrated by the side by side comparison in the movie embedded in the pdf version of this document. Again 0.01 and 0.002 time step sizes are compared. Linear and angular momentum are conserved, but energy (due to losses in the potential energy) decays at the selected rate even though the time step sizes are different by a factor of 5.

Analysis The increased robustness due to conservation of system invariants in our method comes at a cost: additional linear system solves invoked by the Newton solver and computation of the additional terms in the Hessian (Eq. (2.4)). The latter adds only 3.5% total runtime overhead and we consider it negligible. To compare the methods we count the total number of iterations in the linear algebra solver over the entire run: 3113 for the co-rotational and 7959 for the non-linear energy. One could therefore compare with a co-rotational method run at one third the step size. As demonstrated above (for a reduction in stepsize by a factor of 5), this does not fix the problem of the co-rotational formulation, it only lessens the rate at which system invariants are lost. The inclusion of the additional term in the Hessian (Eq. (2.4)) is essential.

2.6 Surface Parameterization

The 2D case of Eq. (2.1) was studied by Liu *et al.* [37] for purposes of surface parameterization. They treated the energy as a functional depending on both $\mathbf{F}_\phi$ and $\mathbf{R}_\phi$ and optimized each in an alternating

manner, fixing R to find the best $\mathbf{F}_\phi$, updating $\mathbf{R}_\phi$ to be optimal, and repeating until convergence. They also considered distance to the closest similarity and describe a mixed functional (their Eq. 5) which trades off similarity against isometry. However, in that functional the isometry constraint enters to fourth order (their $(a^2 + b^2 - 1)^2$), while we can control the trade-off with only second order. For $(\alpha = 0, \beta = 1)$ we get as-similar-as-possible (ASAP) solutions and for $(\alpha = \beta = 1)$ as-rigid-as-possible (ARAP) (see Fig. 2.5 and compare with Fig. 7 of Liu *et al.*), with $\alpha \in [0, 1]$ controlling the trade-off.



(a) (b)

Figure 2.5: Our method applied in the 2D setting to parameterize a cow model with a given texture connectivity. For $\alpha = 0$, $\beta = 1$ ASAP, *i.e.*, discrete conformal maps result (a), and for $\alpha = \beta = 1$ ARAP maps similar to Liu *et al.* result (b).

Comparison The alternating minimization approach of Liu *et al.* has the advantage of only using a constant system matrix, the Laplace-Beltrami operator, which may therefore be pre-factored. In our Newton solver the Hessian changes and thus cannot be pre-factored. But the number of iterations are far fewer for us than in their alternating solver because of the quadratic convergence of Newton versus their linear convergence (see Fig. 2.6 for data from the 3D case; the results are similar in 2D). We also require no specific isometry constraints (of higher order) to manage the trade-off between similarities and isometries. Instead the trade-off is a natural consequence of the split of Y into trace-free and multiple-of-the-identity parts.

As will be described further in Chapter 4, Liu's 'alternating solver' is special case of a Gauss-Newton solver for a nonlinear least squares problem, which uses gradient information to build an approximate Hessian.

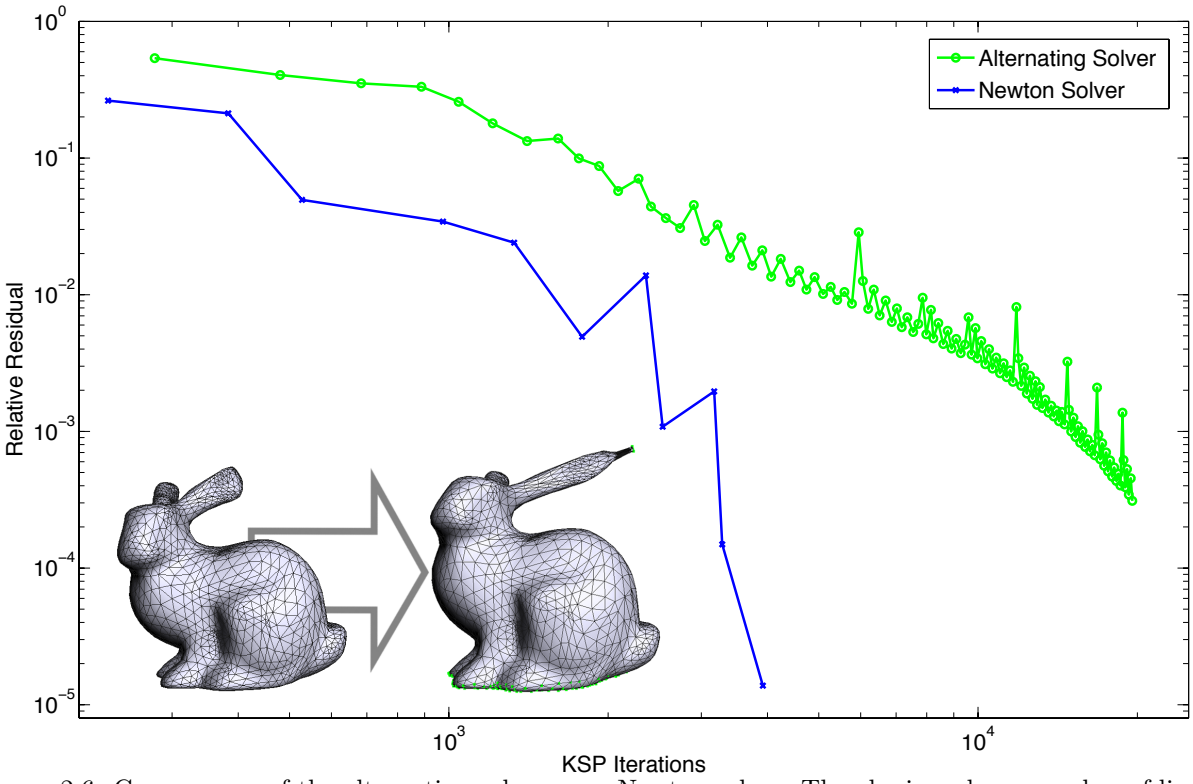


Figure 2.6: Convergence of the alternating solver resp. Newton solver. The abscissa shows number of linear system iterations as a common measure of time.

2.7 Surface Modeling

For the case of triangle meshes in $\mathbb{R}^3$ Sorkine and Alexa [57] model a discrete deformation energy which, as in the case of Liu *et al.*, is a function of $\mathbf{F}_\phi$ and $\mathbf{R}_\phi$, using the same alternating optimization approach. Here the rotations are chosen at vertices to be nearest to the cross covariance of original and deformed edges summed over the incident edges with 2D cotan weights. (Our $\mathbf{R}_\phi$ results if the sum is taken over edges incident to a triangle.) As a consequence they incorporate the extrinsic curvature of the surface and their discrete energy is a mix of membrane and bending terms. No continuous energy is given and it is unclear how the bending contribution depends on the geometry or how its relative weighting with respect to the membrane energy is determined.

If we modify their energy slightly by summing over all edges incident to the triangles incident to a given vertex, *i.e.*, we also add the "rim edges" to their "spoke edges" sum, we arrive at an energy which can be analyzed. In that case the rotation $\mathbf{R}_\phi^i$ at vertex i is closest to $\mathbf{F}_\phi$ integrated over the entire 1-ring of triangles of a given vertex. Using a Taylor series argument one can then show that the resulting continuous energy differs from ours by a bending term weighted by r^2 where r is the radius of integration over which

$\mathbf{R}_\phi$ was chosen to be optimal

$$\tilde{E} = \int_{\mathcal{B}} \left[|\mathbf{F}_\phi - \mathbf{R}_\phi| + r^2 \langle d\mathbf{R}_\phi, d\mathbf{R}_\phi \mathbf{Y}_\phi \rangle \right] \mathbf{dV}.$$

As a consequence the bending contribution is dependent on the mesh element size (r) and goes to zero as the mesh is refined. (The vanishing of the bending term is not just a concern in the refinement limit, but is easily noticeable for meshes with strongly differing discretization rates in different parts of the surface.)

2.8 Geodesic Interpolation in Shape Space

A metric on the space of shapes may be defined with the aid of the elastic deformation energy. Given two shapes $\mathcal{M}_0$ and $\mathcal{M}_1$, the distance between them is the length of a geodesic

$$d^2(M_0, M_1) = \min_\gamma \int_0^1 \left(\int_{\mathcal{M}_\gamma(t)} |\dot{\mathbf{Y}}|^2 \right) dt$$

with γ ranging over all paths connecting $\mathcal{M}_0$ and $\mathcal{M}_1$. This point of view was pursued by Kilian *et al.* [33] for triangle meshes in $\mathbb{R}^3$. Their model does not derive from an elastic energy, but can be understood as the squared residuals of a point mass and spring model with unit coefficients on all edges independent of how stretched they might be.

We can use our elastic energy to approximate such geodesic paths as follows. Recall that a "point" $\mathcal{M}_t$ on a geodesic path from $\mathcal{M}_0$ to M_1 is a critical point of the functional

$$\mathcal{M}_t = \text{argmin}_{\mathcal{M}_x} (1-t) d^2(\mathcal{M}_0, \mathcal{M}_x) + t d^2(\mathcal{M}_x, \mathcal{M}_1).$$

As t ranges over $[0, 1]$, $\mathcal{M}_t$ moves along the geodesic from $\mathcal{M}_0$ to $\mathcal{M}_1$. We now exploit the fact that the elastic energy agrees with the geodesic distance to second order since $|(\mathbf{F})_t - \mathbf{R}_t|^2 = |\dot{Y}_0|^2 t^2 + \text{h.o.t}$ (for $f_t : M_0 \to M_t$), to replace the actual geodesic distance with our energy as a distance approximation

$$\tilde{\mathcal{M}}_t := \text{argmin}_{\mathcal{M}_x} (1-t) E_1(\mathcal{M}_0 \to \mathcal{M}_x) + t E_1(\mathcal{M}_1 \to \mathcal{M}_x).$$

(Here $E(\mathcal{M}_a \to \mathcal{M}_b)$ denotes the energy of the map from a reference shape M_a to a deformed shape $\mathcal{M}_b$.) This generates intermediate shapes $\tilde{\mathcal{M}}_{t \in [0,1]}$, which are static equilibria (for fixed t) of two weighted non-linear "springs" pulling on the intermediate shape from the beginning resp. end configuration. The corresponding energy minimization problem is no harder than what we already have. Setting up the gradient and Hessian does require twice as many evaluations since they are a convex combination ($(1-t)$ resp. t) of the gradients and Hessians of the individual energies. Figure 2.7 shows snapshots of the resulting trajectory between a

straight and twisted/bent beam. One may now approximate a locally optimal geodesic trajectory arbitrarily

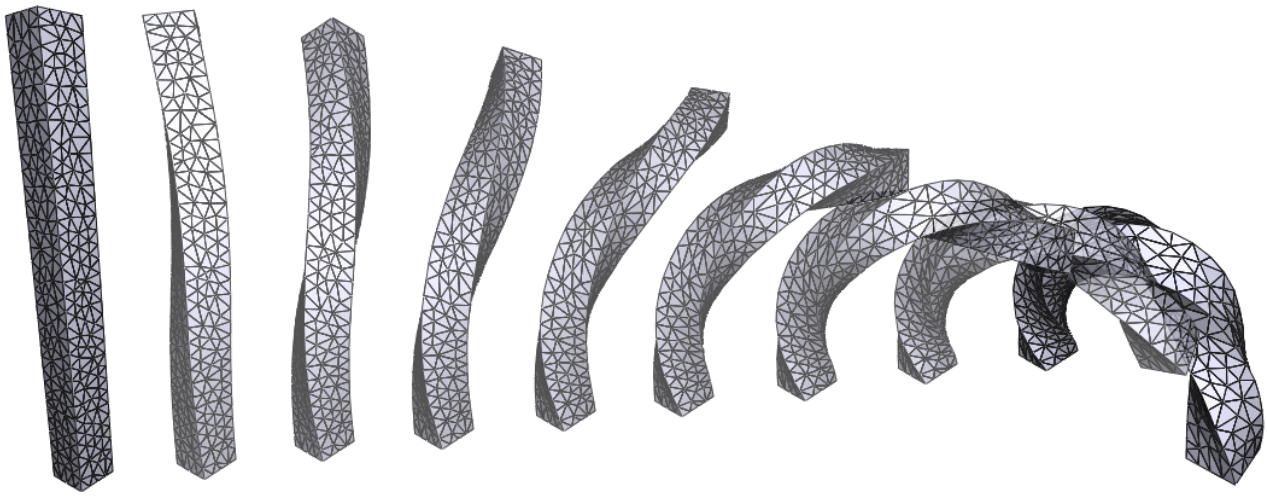


Figure 2.7: Near geodesic interpolation between two shapes using the elastic energy as an approximation of geodesic distance. See also the movie embedded in the pdf version of this document.

closely by minimizing over more and more intermediate shapes (as was done by Kilian *et al.*).

2.9 Summary

Our approach was motivated by the breadth of successful applications of Laplacian-like models for many algorithms which appeal to a physical modeling metaphor. Starting with a smooth model (Eq. (2.1)), which captures the core idea, a careful analysis of the geometric structures underlying it pays off in a number of ways. The biggest practical payoff comes from the additional term in the Hessian due to the rotations. It enables a far more efficient solver (Newton) and is the key to long time stability of our time integrator. The explicit role of rotations as a footpoint for a distance computation also illuminates the issue of inverted elements: one must not accidentally use an orientation reversing orthogonal transformation. (This is of course also true for inverted elements in standard elasticity, though it may not always be as apparent.) Our analysis also enabled us to make the connection with the Biot strain, validating intuitions expressed in a variety of previous work that these models are elasticity-like. In fact, when implemented correctly, we have a first class elasticity model, with textbook parameters, for the large displacement/rotation, small strain regime.

As far as the co-rotational method is concerned our observations can be interpreted as a prescription for improving existing co-rotational simulation systems: add the missing Hessian term and invoke a Newton solver at each time step! Little else would need changing in existing codes to gain the benefit of increased stability. (Or, if one wishes, the ability to take much larger time steps without the usual dissipative effects

of doing so.)

Chapter 3

Bounded-Distortion, Inversion-Free Parameterization

The results presented in this chapter are based on collaborations at Rhythm and Hues Studios with Nathan Litke. Liliya Kharevych helped to render images.

In this chapter, we extend the surface parameterization method in Chapter 2 and present a framework for surface parameterization which, for the first time, provides quality layouts of production meshes [1]with bounded conformal distortion and no inverted elements. This is accomplished through the application of the augmented Lagrangian method for constrained optimization of an elastic energy. We show that intermediate minimization problems to be solved are linearly constrained nonlinear least squares problems with easily computable residual gradients. The system can be sequentially minimized with an easy-to-implement Gauss-Newton and Modified Conjugate Gradient methods, very simple tools which make the method feasible to implement without complicated optimization software. Alternately, more robust and tunable linear solvers and nonlinear optimization tools can be employed. We provide examples of the framework applied to production-scale meshes, showing its suitability to solve many practical problems in texture layout. The method is robust enough to often find a solution from zero initial data, yet can still recover bounded-distortion parameterizations with no local inversions.

Much as we were able to show in Chapter 2 that a geometrically and computationally attractive quantity can be associated with a well-founded elastic theory, we shall be able to relate bounds on what we'll term *special conformal distortion* to an attendant efficient algorithm for an elastic energy that avoids the sorts of 'inversion-prevention fixes' that plague methods which consider conformal distortion. This parallels the notion from the previous chapter that we can vastly simplify existing methods by considering the *nearest rotation* factor $\mathbf{Y}$ as opposed to the polar factor $\tilde{\mathbf{Y}}$.

We present a simple, novel, and effective method of preventing inversion and bounding conformal distortion in surface parameterization, which for the first time can easily be applied to complex meshes with tens

[1]that is, models of considerable size and geometric complexity, here taken from actual meshes used in films

of thousands of degrees of freedom. This improves on the state of the art by simplifying the constrained optimization problem to be solved and showing that it can be approached with simple tools in a way that scales to large problems. We show, perhaps surprisingly, that what is in general a hard problem, namely a nonlinearly inequality-constrained nonlinear non-convex minimization problem, can be attacked by noting its structure: our nonlinear objective is a least squares objective and the residuals therein are indeed only *geometrically* non-linear. We demonstrate the effectiveness of the augmented Lagrangian method for large problems in geometry processing, and show that it can also be implemented as a series of least squares problems, solvable without any complicated optimization machinery. We combine these theoretical and computational advances with a useful set of constraints for practical parameterization, opening an avenue towards practical, interactive automatic texture coordinate layout.

3.1 Conformal Distortion and Special Conformal Distortion

Consider a mapping ϕ which induces a deformation gradient $\mathbf{F}$ on $\mathcal{B}$. If $\mathbf{F}(X)$ has singular values $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n \geq 0$ then the *conformal distortion* is defined as the ratio of the largest and smallest singular values.

Conformal distortion arose in the study of quasi-conformal maps, which are those with bounded conformal distortion. In geometry processing, conformal distortion provides a way to quantify deviation from conformality.

Note that the singular values of $\mathbf{F}$ are also the (non-negative) eigenvalues of the polar factor $\tilde{\mathbf{Y}}$. We will argue that, as opposed to some related methods [35], it is in fact the eigenvalues of the *nearest rotation factor* $\mathbf{Y}$ that we would like to control. To this end, we define a new term, *special conformal distortion*, as the ratio of the largest and smallest eigenvalues of $\mathbf{Y}$. Note that this ratio is negative if and only if the smallest eigenvalue λ_n of $\mathbf{Y}$ is negative, and that it approaches $\pm\infty$ as $\lambda_n \to 0^{\pm}$.

This distinction is closely related to the distinction between the polar and nearest rotation decompositions; it is in a certain sense irrelevant to mathematical elasticity, since any real material could not be inverted. However, If one seeks to relax the set of admissible configurations to allow inverted ones, then a theory which distinguishes between a configuration and its inversion is preferable.

3.2 Surface Parameterization

Surface parameterization is a process of assigning coordinates to a manifold. Often, this means defining a single chart for a topologically disk-shaped domain, but also includes mappings from a manifold to a simpler, but topologically equivalent manifold; a common example is mapping a closed, bounded 2-manifold to a 2-sphere. Texture mapping is the process of parameterizing a 2-dimensional surface.

Nearly four decades since its introduction, texture mapping is still the most widely used method for

adding realism to 3D computer graphics. In commercial entertainment and media production, a texture artist creates a canvas in texture space on which to paint the texture by parameterizing the model of a surface into patches. Each patch maps a region of texture space to the surface, and areas that receive more scrutiny, like the face of a character, are given more texture space through a larger canvas. Control over local distortion is essential to lay out patterns in 2D. Seams between charts create visible artifacts that artists painstakingly avoid by placing them where they're less likely to be noticed, or concealing them by blending multiple textures. Even commercial software that allows textures to be painted on the surface by projecting images into texture space are still constrained by the quality of the parameterization.

While surface parameterization is a well-studied field [16, 55], automatic methods are rarely directly useful to texture painters or other potential users. This is because many automatic methods have the potential to allow unusable parameterizations with local inversions, global lack of injectivity, or excessive distortion. Many seek to produce conformal maps, which are prone to large area distortions. Additionally, parameterizations must almost always satisfy additional constraints or be 'directable', which is not possible with all methods.

'Angle-based' [27, 58] and other novel methods [12] have had great success is generating discrete conformal and quasi-conformal maps, and in particular elegantly allow cone singularities. However, proximity to conformality is only one measure of the quality of a parameterization, and these methods are difficult to adapt to point constraints.

'As Rigid As Possible' methods in 2D and 3D [56, 25] define an energy which penalizes deviation from rigidity, often chosen for its computational simplicity, robustness, and ability to handle badly distorted configurations. These can be extremely efficient and robust [8], yet can get trapped in local minima or, fatally for texture mapping applications, converge to an energy minimum with 'flipped triangles', that is a configuration which is not locally injective. Related elasticity-based methods [30, 10] quantify distortion in a physically intuitive way, but are often computationally demanding, poorly conditioned, and more likely to converge to suboptimal local minima, particularly if infinite energy barriers to inversion are included.

Our method from Chapter 2 can be considered both an elastic model and an 'as-rigid-as-possible' method, and it forms the basis of our method. Its major failing, however, is that it allows for arbitrarily distorted and inverted elements.

Recently, powerful methods in convex optimization have been leveraged to apply bounded distortion constraints to a whole family of parameterization methods [35]. However, due to their generality and reliance on complex existing tools for convex problems, these methods have not yet been scaled to larger systems. Here, we utilize the *augmented Lagrangian method*, a powerful, general, yet simple-to-implement tool for constrained optimization. It does not require any convexification of constraints or complicated machinery, and as we show, can produce practical results on meshes of the scale used in production.

3.3 Elastic Energy

As an elastic model, we modify the elastic energy presented in (2.2) by adding an additional scalar field s controlling 'reference scaling'.

$$E_{1,s}(\phi) \doteq \frac{1}{2}\int_{\mathcal{B}} \rho||\mathbf{Y}_\phi - s\mathbf{I}||^2_{\alpha,\beta}\mathbf{dV} \tag{3.1}$$

Note that the definition of $E_{1,s}$ of course depends on $\mathcal{B}$. Typically $\mathcal{B}$ will be a disk-shaped patch. Multiple patches can be constrained to share boundaries as discussed in Section 3.5.

While the methods presented below for bounding conformal distortion (and preventing element inversion) could be used to augment any other energy, for the purposes of automatic surface parameterization, using the Biot strain energy is ideal in several ways: it is in many ways the computationally fastest model of elasticity, being 'materially linear', and is similar enough to the penalty energies we will ultimately use to allow reuse of many computations. One failing, that is not a particularly accurate description of any real material, is not critical as the parameterization task is more concerned with geometric quantities than material modeling. Its second failing, that it admits local inversions, is remedied by the constrained minimization procedure described in this chapter.

We consider parameterizations of piecewise linear triangulated surfaces defined by a mapping of vertices to the plane, so the integral becomes an area-weighted sum over triangular elements, and the energy can be written as a function of the positions of the vertices in the plane.

Note that while this energy nominally involves working with quantities in 3D, the energy is rotationally invariant and a sum of contributions over elements. This means that computation can proceed in a local frame for each reference element. Put another way, the elastic energy is the same whether or not the reference mesh is a conforming model or simply the same elements, each transformed in an arbitrary normal-preserving way $S \in \mathrm{SO}(3)$ to the plane. This view makes it obvious that the reference triangles need not come directly from a conforming mesh, as illustrated in Figure 3.1. The parameter s can be interpreted as a scaling on these reference elements. See Figure 3.7 for an example of adjusting this parameter, and its affect on the relative 'average length distortion' $||\mathbf{Y}^2||_{\mathrm{F}}$.

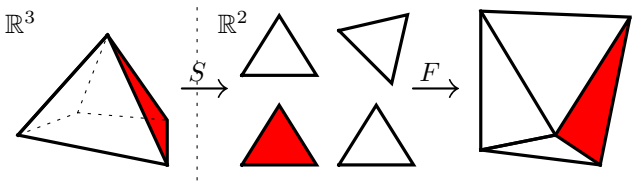
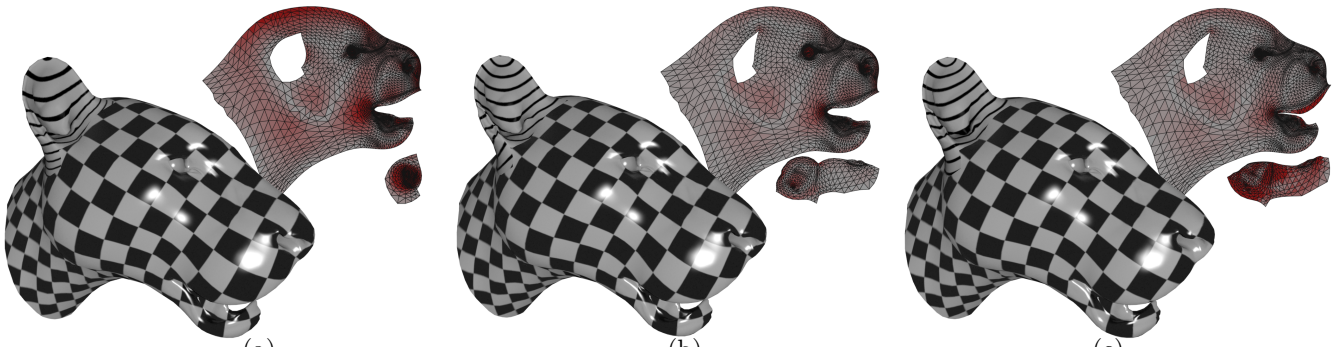


Figure 3.1: The elastic energy is rotationally invariant and defined as a sum over individual elements, so the energy can be computed with respect to a planar, non-conforming reference configuration derived from a mesh by rigidly transforming each element to the plane.

Finally, note that as a weighted sum of squared norms of nonlinear residuals, $E_{1,s}$ is a nonlinear least



(a) (b) (c)

Figure 3.2: Parameterizations of a model that minimize distortion in different ways. A near-conformal parameterization has no visible seams along the edge of the ear but incurs more area distortion (a). An 'As Rigid As Possible' parameterization reduces stretching but has a visible seam (b). Our method produces a bounded-distortion parameterization with a variable angle of rotation along the seam that guarantees it is invisible (c). The color in the parameterization domain corresponds to average length distortion, which in our example an artist has concentrated in less important regions to allot more texture space across the tiger's face.

squares objective with structure to be exploited.

3.4 Linearly Constrained Optimization

The energy (3.1) can be minimized with a variant of Newton's method or most simply with a Gauss-Newton method, advantageous for its implementational simplicity and the fact that it guarantees descent [46]. The energy (3.1) is a nonlinear least squares objective, so a quasi-Newton system can be computed from the gradients of the residuals alone. As we show in Section 3.7, this method can also be used sequentially to solve the more complex nonlinearly constrained problem.

To solve the constrained linear least squares systems that arise, we use the Modified Conjugate Gradient method (MCG) [3]. This allows linear constraints to be described as directions in which the solution is prescribed. These directions are then projected out of update steps during the iterative solve, which is otherwise a standard linear Conjugate Gradient solver. This method offers many practical advantages. As an iterative method it is well-suited to to 'warm-starting' common in interactive applications, and is practical for the large sparse systems that arise in geometry processing. Secondly, it is simple to modify an existing Conjugate Gradient implementation to project onto a linear subspace. Lastly, while we do not fully address the issue of *global* non-injectivity in this work [2], the method allows constraints to be updated easily, allowing the integration of a constraint-based collision resolution method. Indeed, this capability was a key motivation for introducing MCG to the graphics community [5]. The method was described earlier in the numerical linear algebra community as a 'Projected Conjugate Gradient Method' [46, 50].

[2]Though see Figure 3.2 for an excellent practical solution!

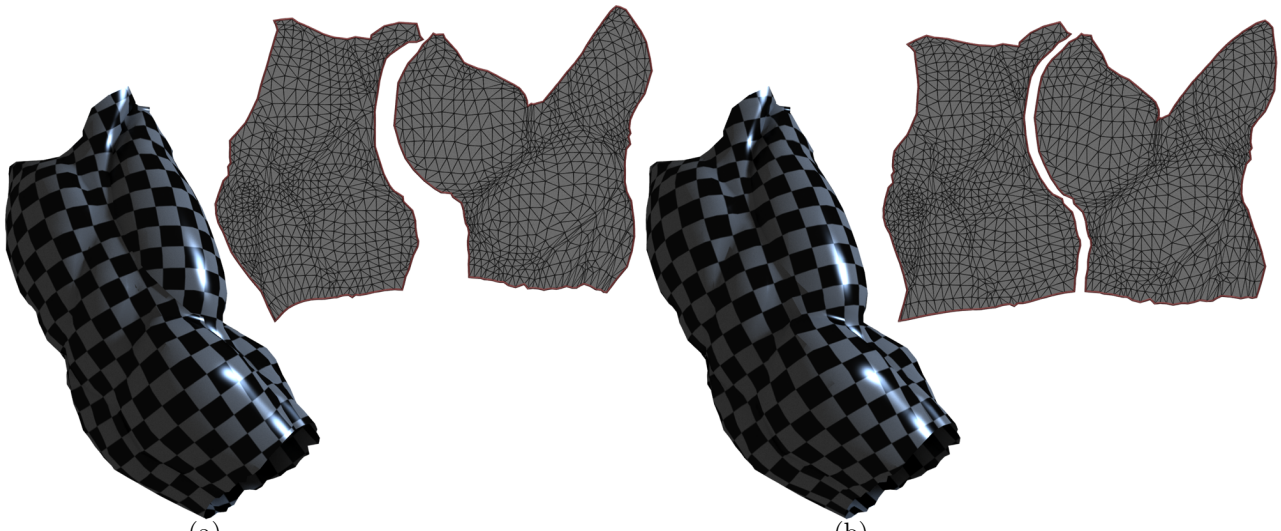
3.5 Seam Constraints

In many applications, it is desirable to align one set of nodes perfectly with another, relating them by an affine transformation, thus facilitating painting across seams and arbitrary divisions of patches to better use real estate in the 'uv space' used by texture artists. The MCG framework handles these constraints; if two sets of points $\{l_1, ..., l_k\}, \{r_1, ..., r_k\} \subset \mathbb{R}^2$ are to be related by an affine transformation $\mathcal{A}$, this amounts to $2k$ linear constraints of the form

$$l_i - \mathcal{A}r_i = 0, \quad i = 1, ..., k.$$

This method works equally well to describe a linear relation between two sets of edges, as shown in Figure 3.3. If L is a linear operator, then such a constraint is described by $2(k-1)$ constraint directions

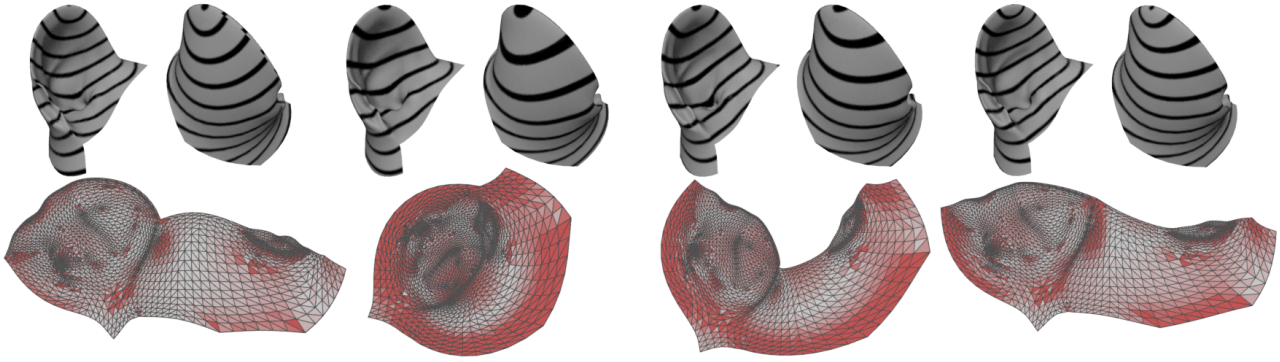
$$(l_{i+1} - l_i) - L(r_{i+1} - r_i) = 0, \quad i = 1, \ldots, k-1$$



(a) (b)

Figure 3.3: Models are cut into patches for topological reasons, or simply to better utilize texture space real estate. Here, an arm cut into two patches is parameterized without any seam constraints (a), and by imposing that both pairs of seams be related by a translation, hence enforcing the natural tube topology of the shape. This allows the mesh to be seamlessly textured.

The example in Figure 3.4 demonstrates the need for variable-angle seam constraints, which we can incorporate with an ad hoc method, but which remain to be treated properly. Achieving this capability is important for layout of complex, segmented models, as it would allow for an entire model to be parameterized as a single system, with perfectly matching seams everywhere.



(a) (b) (c) (d)

Figure 3.4: For a cone-like shape, such as the tiger's ear, it can be difficult to determine a priori what the cone angle should be. Here, we show the tiger's ear with no seam constraints (a), a seam angle of 0° (b), a seam angle of 90° (c), and an angle of 190.25° optimized by an ad hoc automatic method (d). An interesting extension of this work would be to enforce variable-angle seam constraints within the Augmented Lagrangian framework.

3.6 Bounded Distortion and Inversion Prevention Constraints

A useful and easily computed local measure of distortion is *conformal distortion* $\frac{\sigma_1}{\sigma_n}$, the ratio of the singular values of the deformation gradient $\mathbf{F}$ (the square roots of the singular values of $\mathbf{C}$), discussed above in Section 3.1. In the 2-dimensional case, we are aided greatly by the fact that there are only two singular values and as such the conformal distortion can be easily calculated. There are many fascinating links to complex analysis, which we only mention in passing here; see the thesis by Crane for many techniques related to this insight [11].

For a parameterization to be useful, this distortion should be bounded and elements should not be inverted. If we identify $\mathbf{F}$ with its matrix representation $F \in \mathbb{R}^{2\times 2}$, decompose it as

$$A = \begin{bmatrix} \gamma_1 + \delta_1 & \delta_2 - \gamma_2 \\ \delta_2 + \gamma_2 & \gamma_1 - \delta_1 \end{bmatrix} \tag{3.2}$$

and consider the 2-vectors $\gamma = [\gamma_1, \gamma_2]$ and $\delta = [\delta_1, \delta_2]$, then the singular values are $\sigma^{\pm} = ||\gamma| \pm |\delta||$. $|\cdot|$ denotes the usual Euclidean 2-norm, and the requirement that the conformal distortion be less than or equal to C can be written in terms of $|\gamma|$ and $|\delta|$. An element is inverted when $\det(F) < 0$, equivalent to $|\delta| > |\gamma|$. This structure forms part of the basis for the recent work by Lipman [35], which applies these and other constraints to a wide range of parameterization schemes, solves the resulting constrained minimization problems by convexifying a set of constraints, introducing slack variables, and employing general tools for second-order cone programs (SOCPs). To bound distortion *and* prevent inversion, however, requires an extra convexification step.

However, in our framework, we can encapsulate both requirements with a single constant on *special*

conformal distortion.

The key observation is that (real) eigenvalues of $\mathbf{Y}$ are given by $|\gamma| \pm |\delta|$. These are equal in absolute value to the singular values of $\mathbf{Y}$ (which are equal to the singular values of $\mathbf{F}$).

The ratio of the eigenvalues of $\mathbf{Y}$ is given by $\frac{|\gamma|+|\delta|}{|\gamma|-|\delta|}$, similar to conformal distortion except that the denominator may now be negative, which is the case precisely when $\det \mathbf{Y} < 0$. With some rearranging, the expression that this expression be bounded above by C and below by 0 is equivalent to

$$\tilde{C}|\gamma| - |\delta| \geq 0 \tag{3.3}$$

where $\tilde{C} \doteq \frac{C-1}{C+1} \in [0, 1]$.

Note that $C = 1$ ($\tilde{C} = 0$) is the requirement that there the map be conformal[3], and that $C = +\infty$ ($\tilde{C} = 1$) corresponds to the constraint that the element not be inverted but can have unbounded conformal distortion.

Thus, we use a constraint function (which can be thought of as a scalar field over $\mathcal{B}$)

$$c(\phi) \doteq \tilde{C}|\gamma_i| - |\delta_i| \geq 0$$

where δ_i corresponds to decomposing F_i as in (3.2).

Noting that our elastic energies are nonlinear least squares objectives, the minimization problem to be solved is a nonlinearly inequality constrained nonlinear least squares problem

$$\text{minimize } E_1(\phi) \quad \text{s.t.} \quad c(\phi) \geq 0.$$

The two factors γ and δ in (3.2) are in fact the norms of the holomorphic and antiholomorphic parts of the map $\mathbb{C} \to \mathbb{C}$ induced by $\mathbf{F}$. For our purposes, it's more useful to notice that these factors are, in the coordinate frame described by $\mathbf{R}$, the norms of the diagonal and trace-free parts of $\mathbf{Y}$, that is exactly the terms we weight independently when dealing with isotropic elasticity. We will use this notation to make explicit the connections with the other work in this thesis, and in particular to show that similar mechanisms to those in Chapter 2 can be used to form derivatives of the constraint functions as described further in Section 3.8. These factors are invariants in the sense described in (1), so we can expect to be able to compute their derivatives efficiently.

Further observe that the constraint functions are objective functions of $\mathbf{C}$ and thus (integrated) fit the definition of an isotropic hyperelastic stored energy function W as discussed in Section 1.4. This allows us to leverage much of the intuition and computational efficiency we gained from the elastic analogy.

[3]Being conformal in this sense is a very strict requirement for a piecewise linear mapping of a surface, and is only possible for very specific surfaces, such as developable ones.

3.7 Augmented Lagrangian Method

A general method for solving problems of this type is the *augmented Lagrangian method*. Very briefly, the method approaches the equality constrained minimization problem

$$\text{minimize } f(x) \quad \text{s.t.} \quad b(x) = 0$$

by maintaining estimated Lagrange multipliers λ. The algorithm proceeds by approximately minimizing the augmented Lagrangian function

$$L_A(x) = f(x, \lambda) + \frac{\mu}{2}||b(x)||_2^2 - \langle \lambda, b(x) \rangle$$

with respect to x, using the constraint residuals $b(x)$ at the computed minimum to obtain a better estimate of λ, and repeating to convergence. Unlike a quadratic penalty method, convergence to a local minimum can be obtained for bounded penalty parameter μ. The method can be extended to inequality constraints $c(x) \geq 0$ by introduction and successive explicit optimization of slack variables.

Estimated Lagrange multipliers are typically updated according to the notion that the gradient of the penalty term should give an estimate of the update required to a Lagrange multiplier. Thus, a typical update rule is

$$\lambda \leftarrow \lambda - \mu c(x^*)$$

where x^* is the computed minimizer from the previous subproblem. What might be a nice hand-waving elastic analogy in the general case turns out to be a rigorous elastic analogy here! That is, we 'attach a rubber band' (a hyperelastic stored every function) to each element and use the direction of the force it produces to impose a constraint force (a Lagrange multiplier). That this intuition should lead to a convergent method is of course not obvious, and for more detail we refer to Chapter 17 of the textbook by Nocedal and Wright [46].

To see how to handle inequality constraints, consider the problem

$$\text{minimize } f(x) \quad \text{s.t.} \quad c(x) \geq 0$$

and introduce slack variables s to convert the problem to one with equality constraints and positivity constraints on the slack variables,

$$\text{minimize } f(x) \quad \text{s.t. } c(x) - s = 0, s \geq 0$$

The Augmented Lagrangian problem at each step is now to minimize

$$f(x,\lambda) + \frac{\mu}{2}||c(x) - s||_2^2 - \langle \lambda, c(x) - s\rangle$$

over all x and s over the set where $s \geq 0$ (the positive orthant). This function is convex and quadratic in s and the minimizer $s^*(x)$ can be computed explicitly as $s^*(x) = \max(0, c(x) - \lambda/\mu)$.

This gives a simple expression which extends the augmented Lagrangian method to inequality constraints without any explicit slack variables, namely to minimize

$$f(x) + \frac{\mu}{2}|\min(0, c(x) - \lambda/\mu)|_2^2 - \frac{1}{2\mu}|\lambda|_2^2$$

with an adjusted update rule,

$$\lambda \leftarrow \max(0, \lambda - \mu c(x))$$

The method can be thought of both as a quadratic penalty method with accelerated convergence or as a Lagrangian method which iteratively fixes and improves an estimate of the Lagrange multipliers. Each of these viewpoints allows the use of the MCG method, which is attractive from the viewpoint of simple implementation; a pure quadratic penalty method becomes arbitrarily poorly conditioned as the convergence tolerance shrinks, and the pure Lagrangian approach is indefinite.

The augmented Lagrangian function in our setting is

$$\mathcal{L}_A(\phi, \lambda; \mu) = E_1(\phi) + \frac{\mu}{2}|\min(0, c(\phi) - \lambda/\mu)|_2^2 - \frac{1}{2\mu}|\lambda|_2^2$$

Where

$$c(\phi) = \left(\tilde{C}||\tfrac{\operatorname{tr}\mathbf{Y}_\phi}{d}|| - ||\bar{\mathbf{Y}}_\phi||\right)$$

This formulation, combined with the ease of computing the derivative of $c(\phi)$ in the discrete setting, becomes particularly attractive in this setting as it remains a nonlinear least squares problem with respect to deformed vertex positions, allowing the use of a Gauss-Newton method requiring only the first derivative of $c(\phi)$.

The augmented terms are easily added to our existing framework by including extra terms in the objective, gradient, and (quasi-)Hessian computations. The Gauss-Newton method is guaranteed to produce a descent direction, and while in the unconstrained case full steps always decrease the energy, we require a linesearch to ensure descent at each step that penalty terms enter.

3.8 Assembling Gauss-Newton and Newton Systems

Expressions for computing the Newton system for E for a triangulated surface are presented in Chapter 2. By ignoring terms involving the derivatives of $\mathbf{R}$, a Gauss Newton system is recovered.

It remains to describe how to compute first and second derivatives of the expression

$$P(\phi) \doteq \frac{\mu}{2} |\min(0, c(\phi) - \lambda/\mu)|_2^2 \doteq \frac{\mu}{2} |\tilde{c}(\phi)|_2^2$$

over a triangulated surface. We have $\delta_\psi P(\phi) = \mu \tilde{c}(\phi) \delta_\psi c(\phi)$ and $\delta^2_{\psi,\xi} P(\phi) = \mu \left(\delta_\psi c(\phi) \delta_\xi c(\phi) + \tilde{c}(\phi) \delta^2_{\psi,\xi} c(\phi) \right)$. Further, $c(\phi)$ is constant on each triangle, so we compute its first two derivatives when $\mathbf{F}_\phi$ is given locally by the affine mapping of a reference triangle to the plane.

We can consider each term separately, as before adding subscripts to indicate to which deformation or variation thereof particular terms correspond to.

Note the identities

$$\left\| \frac{\operatorname{tr} \mathbf{Y}_\phi}{d} \right\| = \frac{1}{\sqrt{d}} \operatorname{tr} \mathbf{Y}_\phi = \frac{1}{\sqrt{d}} \langle \mathbf{Y}_\phi, \mathbf{I} \rangle = \frac{1}{\sqrt{d}} \langle \mathbf{F}_\phi, \mathbf{R}_\phi \rangle$$

Thus we have

$$\delta_\psi \left\| \frac{\operatorname{tr} \mathbf{Y}}{d} \right\| = \frac{1}{\sqrt{d}} \langle \delta_\psi \mathbf{F}_\phi, \mathbf{R}_\phi \rangle$$

Where we have noted that $\langle \mathbf{F}_\phi, \delta_\psi \mathbf{R}_\phi \rangle = 0$ [4]. The second derivative is

$$\delta^2_{\psi,\xi} \left\| \frac{\operatorname{tr} \mathbf{Y}}{d} \right\| = \frac{1}{\sqrt{d}} \langle \delta_\psi \mathbf{F}_\phi, \delta_\xi \mathbf{R}_\phi \rangle$$

The trace-free factor can be rewritten $||\bar{\mathbf{Y}}_\phi|| = ||\mathbf{F}_\phi - \frac{\operatorname{tr} \mathbf{Y}_\phi}{d} \mathbf{R}_\phi||$. Using the orthogonality of $\delta_\psi \mathbf{R}_\phi - \frac{\operatorname{tr} \mathbf{Y}_\phi}{d} \mathbf{R}_\phi$ and $\mathbf{F}_\phi$, we have

$$\delta_\psi ||\bar{\mathbf{Y}}_\phi|| = \frac{1}{||\bar{\mathbf{Y}}_\phi||} \left(\langle \delta_\psi \mathbf{F}_\phi, \mathbf{F}_\phi \rangle - \frac{\operatorname{tr} \mathbf{Y}_\phi}{d} \langle \delta_\psi \mathbf{F}_\phi, \mathbf{R}_\phi \rangle \right)$$

and

$$\delta^2_{\psi,\xi} ||\bar{\mathbf{Y}}_\phi|| = \frac{1}{||\bar{\mathbf{Y}}_\phi||} \left(\langle \delta_\psi \mathbf{F}_\phi, \delta_\xi \mathbf{F}_\phi \rangle - \delta_\psi ||\bar{\mathbf{Y}}_\phi|| \delta_\xi ||\bar{\mathbf{Y}}_\phi|| - \frac{1}{d} \langle \delta_\psi \mathbf{F}_\phi, \mathbf{R}_\phi \rangle \langle \delta_\psi \mathbf{F}_\phi, \mathbf{R}_\phi \rangle - \frac{\operatorname{tr} \mathbf{Y}}{d} \langle \delta_\psi \mathbf{F}_\phi, \delta_\xi \mathbf{R}_\phi \rangle \right)$$

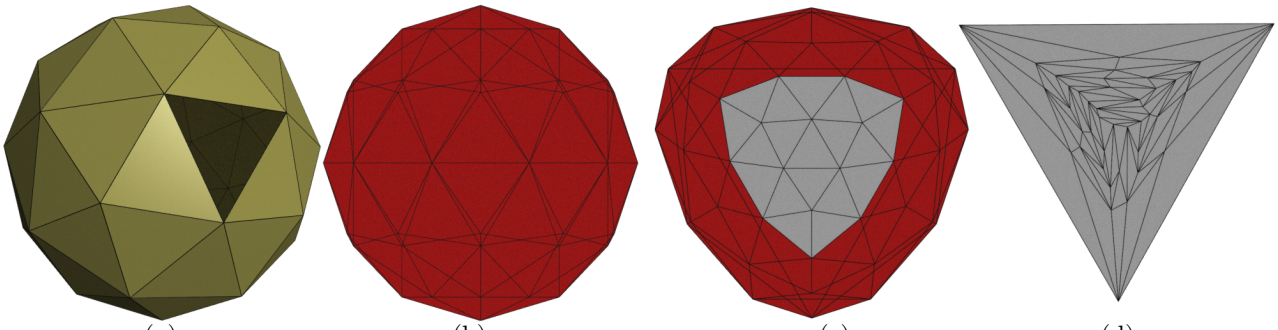
In all the cases above, we can use the convenient expression of $\mathbf{F}_\phi$ on a given triangle as a sum of tensor products of volume gradients and deformed vertex positions to quickly construct the required systems. See Appendix B.2 for the details.

[4] an orthogonality that follows alternately from the equivalent definition of $\mathbf{R}_\phi \doteq \operatorname{argmin}_{\mathbf{R} \in \mathrm{SO}(d)} ||\mathbf{F}_\phi - \mathbf{R}||^2$), from the invariance noted in (1), or by noting that $\langle \mathbf{F}, \delta \mathbf{R} \rangle = \langle \mathbf{Y}, \mathbf{R}^T \delta \mathbf{R} \rangle = 0$ by the orthogonality of symmetric and antisymmetric matrices under the trace inner product

Algorithm 1 Constrained Minimization

procedure ALSOLVE(x_0)
  $x \leftarrow x_0$
  $L_{\text{prev}} \leftarrow \mathcal{L}_A(x, \lambda; \mu)$
  $\lambda \leftarrow \vec{0}$
  $\mu \leftarrow 0$
  for $i = 1, ..., i_{\max}$ **do**
    $x \leftarrow \text{argmin}_x \mathcal{L}_A(x, \lambda; \mu)$ ▷ Gauss-Newton- or Newton-based inner solver
    $L_{\text{prev}} \leftarrow L$; $L \leftarrow \mathcal{L}_A(x, \lambda; \mu)$
    $c_{\text{active}} \leftarrow \min(c(x), 0)$
    if $c_{\text{active}} = \vec{0}$ **then**
      break
    $\triangle_{\text{abs}} \leftarrow |L - L_{\text{prev}}|$; $\triangle_{\text{rel}} \leftarrow \triangle_{\text{abs}} / |L_{\text{prev}}|$
    if $||c_{\text{active}}||_2 < \varepsilon_1 \wedge (\triangle_{\text{abs}} < \varepsilon_2 \vee \triangle_{\text{rel}} < \varepsilon_3)$ **then**
      break
    $\lambda \leftarrow \max(0, \lambda - \mu c(x))$
    $\mu \leftarrow$ UPDATEMU(μ, i)
  return x

function UPDATEMU(μ, i)
  if i $= 1$ **then**
    return μ_0 ▷ Typically, $\mu_0 = 0$
  if i $= 2$ **then**
    return μ_1 ▷ Typically , $\mu_1 \in [0.1, 100]$
  if $i \mod M = 0$ **then** ▷ Typically, $M \in \{3, ..., 10\}$
    return $\tau\mu$ ▷ Typically, $\tau \in [10, 500]$
  else
    return μ



(a) (b) (c) (d)

Figure 3.5: Our constrained parameterization method can solve this pathological icosahedron missing one face (a), which is projected to the plane (b), relaxed to an unconstrained elastic minimum (c), and then subjected to our constrained minimization procedure, producing a parameterization with conformal distortion bounded by 7 and no inversions (d).

3.9 Implementation

Our approach tackles what is in general an extremely computationally challenging problem, namely a non-linearly inequality-constrained, nonlinear, non-convex optimization problem[44], robustly and with with the option of using comparatively rudimentary tools. Indeed, the code used to generate most of the examples in this chapter uses the template library Eigen[17] to define matrix and vector types of hand-coded MCG and Gauss-Newton solvers.

Algorithm 1 gives a rough outline of the constrained optimization procedure, and our choice of update rule for μ.

Ghost materials Our constraints are a useful tool for preventing global self-intersection, as problematic areas can be triangulated and assigned material parameters $\alpha = \beta = 0$. These 'ghost' elements are invisible to the elastic energy yet still subject to the bounded distortion constraints, so serve the intuition that holes and concavities in a shape should be allowed to deform freely, unless they move toward inversion or distortion beyond a certain amount. This technique is used in Figure 3.2 to keep the tiger's mouth open.

The Nearest Rotation Decomposition The factor R in the decomposition $F = RY$ can very efficiently be computed using Algorithm 2.

Algorithm 2 Nearest Rotation

1: **procedure** NEARESTROT(F)
2: $x \leftarrow F_{11} + F_{22}$
3: $y \leftarrow F_{12} - F_{21}$
4: **if** $x = 0 \wedge y = 0$ **then**
5: **return** $I^{2\times 2}$
6: **return** $\frac{1}{\sqrt{x^2+y^2}} \begin{bmatrix} x & y \\ -y & x \end{bmatrix}$

Convergence For many of the examples presented here, we have elected to use linearly convergent Gauss-Newton methods. The advantages are ease of implementation and robustness in the initial steps, as we are guaranteed descent directions from our Gauss-Newton solver. For most applications, only a few iterations are required to produce a usable parameterization. Our method reliably computes good approximate constrained minima, yet is not well-suited for computing these minima to arbitrary numerical accuracy. However, the regime of slow convergence for our solvers typically corresponds to the region of quadratic convergence for Newton's method, which presents an obvious remedy if rapid convergence to small tolerance is required. See Figure 3.6, which shows the behavior of the solver with a tight convergence tolerance on the inner Gauss-Newton loop.

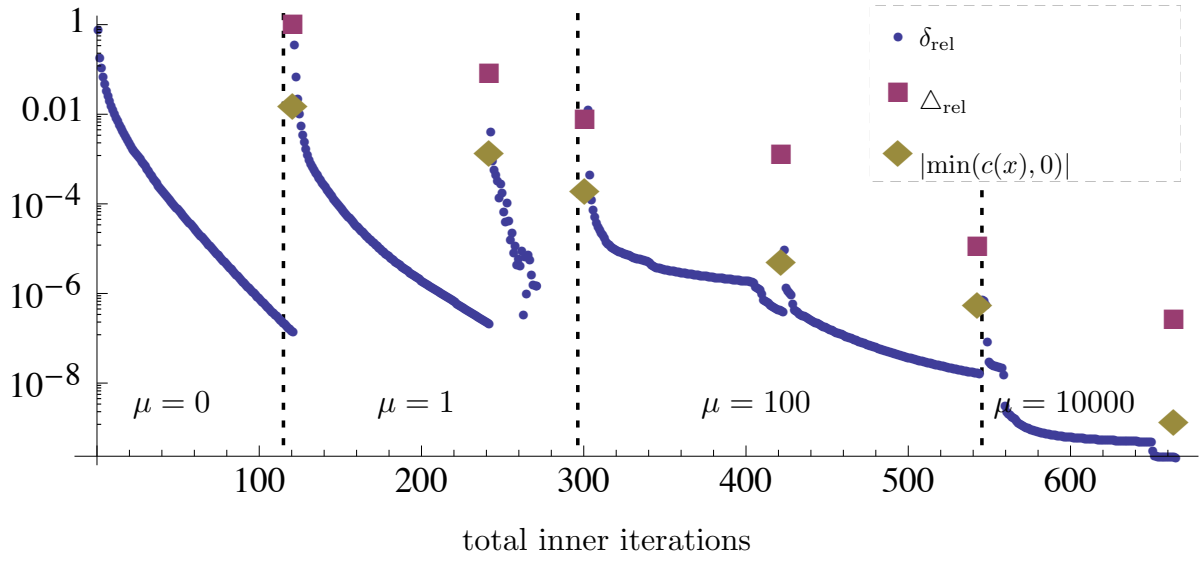


Figure 3.6: The constrained problem in Figure 3.2(c) was solved with 120 maximum inner iterations allotted, and very tight absolute and relative tolerances of 10^{-15} for convergence of the inner Gauss-Newton loop. We graph δ_{rel}, the relative change in the augmented Lagrangian value in the inner loop, along with $\triangle_{\text{rel}}$, the change in this value for the outer loop and the norm of the active constraints. Note the rapid convergence at the outset of each outer iteration leading to slow, linear convergence before λ and μ are updated. In most situations, this initial rapid convergence is enough to produce useful parameterizations, and fewer inner iterations are required.

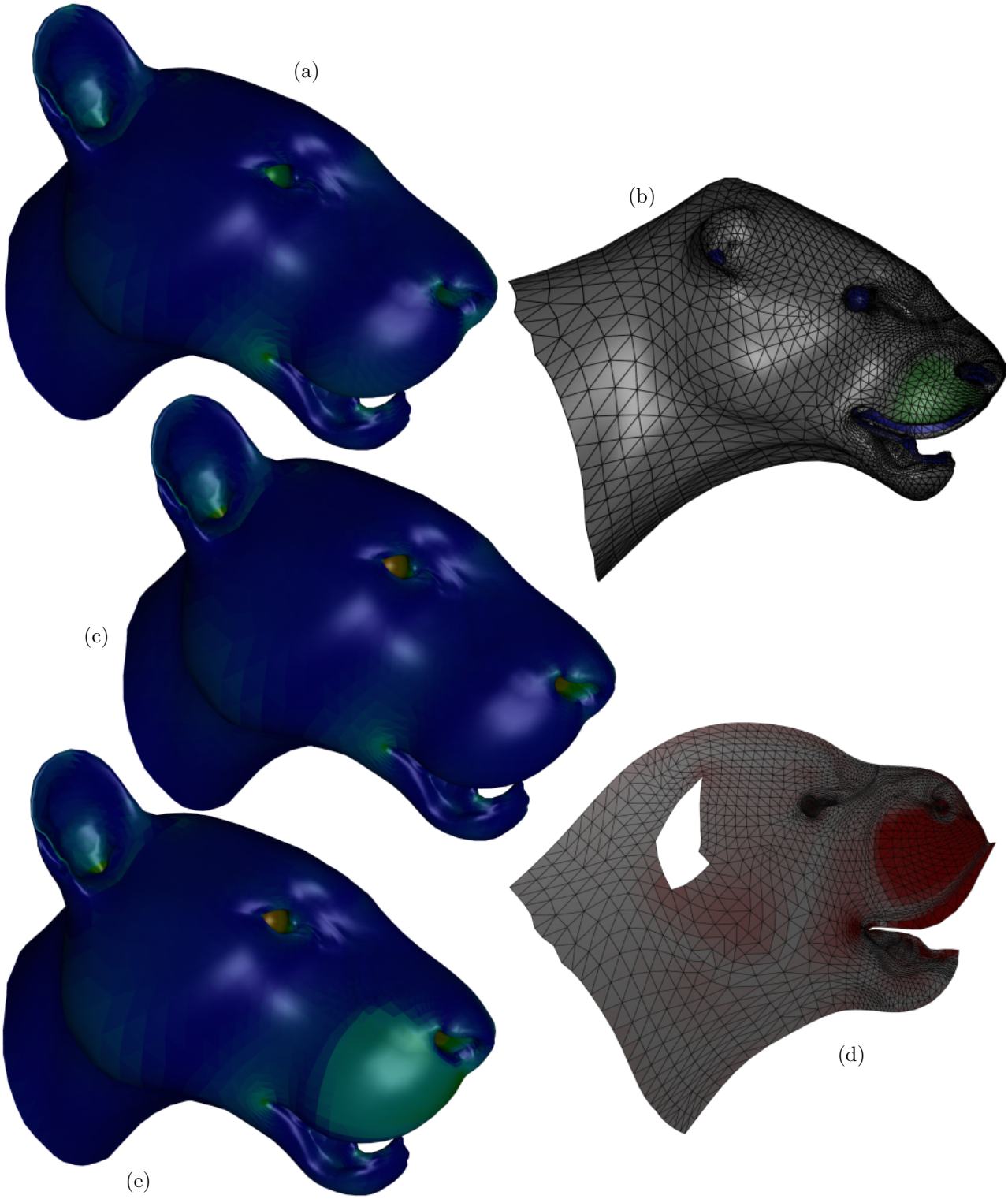


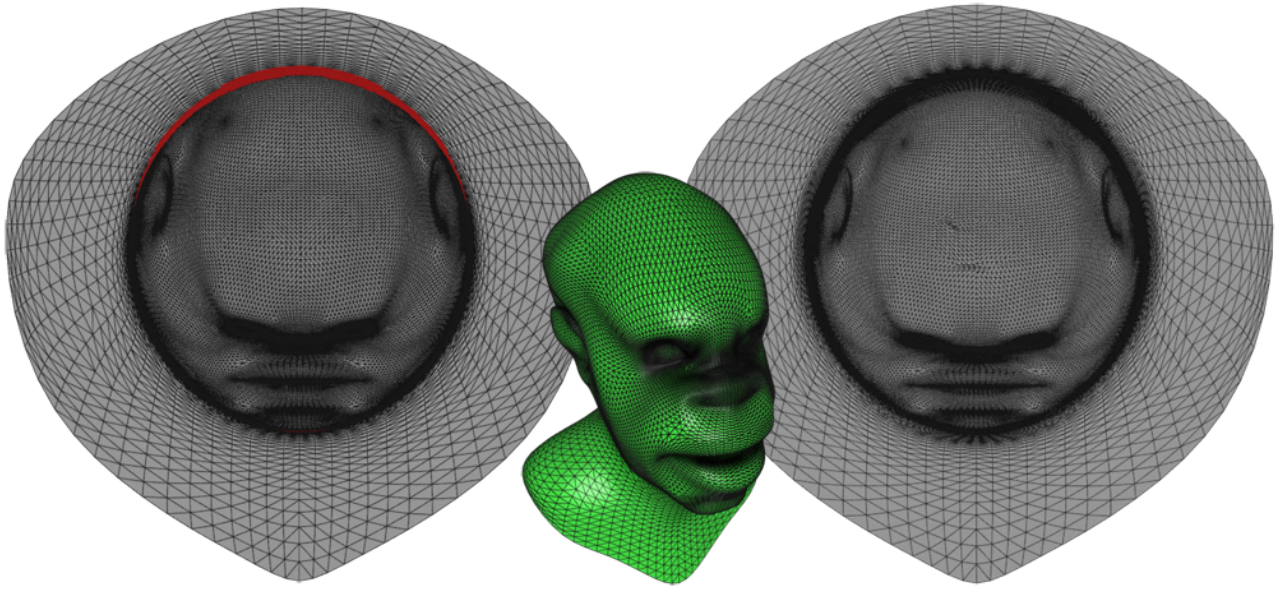
Figure 3.7: Pseudo-color plots of average length distortion at the unconstrained minimum (a) and our parameterization from Figure 3.2 (c) show a reduction in distortion across the tiger's face, achieved by locally decreasing α in less important regions colored blue in (b). Even more texture space is given around the muzzle (d), (e) by locally increasing s in the green region. Triangles added to close the mouth (not shown) with $\alpha = \beta = 0$ prevent the boundary from self-intersecting in (a).

3.10 Results

We tested our method with various production models, ranging in size from a few thousand to a few tens of thousands of degrees of freedom. Our test implementation typically produced usable parameterizations in a few seconds, up to a few minutes for the largest meshes.

A low-dimensional but challenging example is shown in Figure 3.5. Figure 3.7 demonstrates how easily enforceable bounded distortion constraints allow manipulation of the s parameter in the elastic energy, and also how our framework allows the development of a usable tool for texture mapping, here prototyped as a plugin for the widely-used Houdini software.

To examine the limits of the method, we parameterized the model shown in Figure 3.8 as a single patch, requiring many iterations but recovering a valid configuration, even with no initial parameterization and a highly inverted unconstrained minimum energy configuration. This example highlights the robustness of the unconstrained materially linear elastic energy minimization to initial conditions as well as the power of the augmented Lagrangian method to enforce complicated constraints.



(a) (b)

Figure 3.8: We parameterize a difficult mesh with disk topology by computing a bounded-distortion energy minimum, $C = 10, \alpha = \beta = 1$ (so this is a constrained 'As Rigid As Possible' parameterization) with no other constraints beyond those defining the rigid body modes. There is no initial parameterization; the initial configuration consists of every vertex positioned at the origin. The mesh has 43586 degrees of freedom and 43520 elements. The unconstrained minimum (a) has 9941 inverted elements, and the constrained minimum (b) has none.

3.11 Summary

We have demonstrated the great effectiveness of incorporating geometrically attractive energies and constraints into a simple, yet general constrained optimization framework. Indeed, our bounded distortion penalty terms can be interpreted as a fictitious elastic energy which incrementally 'stiffens' until the constraints are satisfied. This automatically lends invariance properties and physical intuitiveness to the procedure. The combination of a robust materially linear elastic model with an accelerated penalty method allows for the first time a parameterization method which is unlikely to get caught in local minima yet can produce a uniformly high-quality texture map with no local inversions.

The work is part of a goal to produce an interactive method for automating many of the tedious tasks involved with generating texture coordinates for 3D models. Production-sized meshes can be processed, and artists can interact with the method in a physically-intuitive way, prescribing spatial constraints and material parameters.

This goal has been elusive for many reasons, many of which have been addressed by the examples in this chapter. By using an elastic energy with intuitive parameters, distortion can be distributed according to scalar fields on a surface describing overall importance ($\alpha + \beta$), relative importance of area and conformal distortion (the ratio of α and β), and desired scaling (s). The constrained optimization framework, in particular the barrier against inverted or overly distorted elements, is essential as a single flipped triangle renders a parameterization useless. Previous methods all either allowed inverted elements, produced purely conformal maps which are of limited useful for texture painting due to their uncontrolled area distortion, or relied on SOCP solvers, thus requiring the program to be convexified and not scaling well to large meshes. We manage to overcome all these limitations, as the tiger head example used repeatedly in this chapter shows. That mesh is one used in actual film production and the fact that we are able to successfully parameterize it represents a step forward in this field.

An intriguing avenue of future work is to incorporate further constraints into the augmented Lagrangian framework. In particular, seam alignment constraints with undetermined angle would be very useful for parameterization.

Chapter 4

Generalized Quadratic Strain Energies

Amongst Seth-Hill elastic strain measures, the Biot and Green-Lagrange measures, ϵ_1 and ϵ_2 respectively, have been used to great effect in graphics applications. Here, we describe an elastic energy, with tunable isotropic material parameters, based on an arbitrary piecewise analytic strain function. In particular, we champion the logarithmic or Hencky measure of strain, ϵ_0 as a basis for numerous applications in elasticity modeling and geometry processing. This energy is quadratic in the logarithm of the metric $\mathbf{C}$ induced by a deformation ϕ, represented by a positive definite matrix at each point on $\mathcal{B}$. We develop a new, general procedure for stable numerical computation of the derivatives of matrix functions. Notably, while we focus on the matrix logarithm, our novel analysis allows for computation of any number of derivatives of arbitrary analytic matrix functions of symmetric positive definite matrices, and also provides a new way to go about numerically stabilizing divided differences of general analytic functions, thus allowing for a wide range of future constitutive modeling possibilities. By recognizing the form of the relevant class of strain energies as nonlinear least-squares problems, we implement a Gauss-Newton solver to complement a Newton solver, and note that this formulation extends existing robust minimization techniques to the class of strain energies quadratic in a given strain measure. We show that, in addition to its mathematical attractiveness, the logarithmic strain-based energy possesses desirable properties in practice. Many advantages stem from an infinite energy barrier against element inversion. Interactive elasticity simulations cannot degenerate. Given any admissible starting point, the minimum of the elastic energy of a surface parameterization is guaranteed to have no flipped triangles. Arbitrary initial velocities are possible in elastodynamics simulation without fear of inversion. Furthermore, the measure is reversible once a density factor is taken into account, and is thus also well-suited to shape space interpolation by elastic analogy.

4.1 Motivation

In general, the elastic energy associated with a deformation depends on all the complexities of materials. However, full descriptions of the responses of real materials are complicated and difficult to obtain, making them impractical for geometry processing both in terms of the computational cost required and the loss of geometric simplicity. Thus, it is common practice to work with elastic energies based on assuming a linear relationship between local *stress* (describing forces on arbitrary infinitesimal plane elements internal to the body) and some measure of *strain*, which characterizes local distortion. Briefly, these strain measures are tensor fields defined as functions of the tensor $\mathbf{C}$, the metric induced by a deformation.

In computer graphics applications, strain energies based on the Lagrange-Green strain $\mathbf{C} - \mathbf{I}$[1] or Biot strain $\mathbf{C}^{1/2} - \mathbf{I}$ have been frequently employed. These measures are two members of a one-parameter family of strain measures, well-known in the mechanics literature as *Seth-Hill strain measures*[54, 22]. Here, we champion the *logarithmic strain* or *Hencky strain* $(1/2)\log\mathbf{C}$ for use in geometry processing. Many authors, including Hill [21], have argued for this strain measure's optimality amongst Seth-Hill strains; we will argue that it is also optimal for many purposes in computer graphics, in particular due to its 'completeness' with respect to element inversions; under this measure, inverted elements are inadmissable configurations and there is an infinite energy barrier protecting against inversion.

Implementations require working with matrix logarithms, however, at some computational cost and implementation effort. We outline practical methods for computing the required matrix logarithms and their derivatives, and also extend the idea of an 'alternating solver' (closely related to a corotational scheme) or 'local/global' approach [36] in which strains are linearized around a local rotation field by describing its equivalence in previous cases with the *Gauss-Newton method* and applying this method to minimize our logarithmic strain energy. This complements standard minimization tools based on Newton's method.

In fact, our methods extend to include any piecewise analytic strain measure, which opens up a wide range of material models at a modest cost.

In this chapter we demonstrate the applicability of the logarithmic strain measure to various applications in elasticity and computational geometry and present methods for minimizing an energy quadratic in this measure. In doing so, we note the applicability of the Gauss-Newton method to this and related problems. Finally, we derive new expressions for arbitrary derivatives of matrix functions and introduce a new, general method for stabilizing divided differences of analytic functions.

[1] Note that it is common, and we argue incorrect, to use the polar factor $\tilde{\mathbf{Y}}$ instead of the nearest rotation factor $\mathbf{Y}$. This does not make any difference when $\det\mathbf{F} > 0$ as is the focus in this chapter, but is crucial when dealing with inverted configurations as we do in Chapters 2 and 3.

4.2 Optimality of the Logarithmic Strain Measure

In several useful senses, the logarithmic strain ϵ_0 is the optimal measure in the class of Seth-Hill strain measures discussed in Section 1.4.4.

Firstly, it is 'complete' in the sense that it maps $\mathrm{Sym}^{++}(\mathcal{B})$ bijectively to $\mathrm{Sym}(\mathcal{B})$. It is the only measure which becomes infinite as a body is compressed to zero volume and also becomes infinite as a body is stretched to infinite volume. In applications this is very useful as it prevents any inversion of elements in a discrete representation of a body yet retains plausible behavior for large stretching.

Further, it is the only measure in its class which is reversible, in the sense that $\epsilon_0(\mathbf{C}) = -\epsilon_0(\mathbf{C}^{-1})$, so, accounting for a density factor, the strain energy associated with a transformation is equal to the strain energy associated with its inverse. This property has been recognized for its usefulness in image registration applications [48], where, as in shape interpolation, there is no clear notion of which of a pair of states should be considered 'reference' or 'deformed'.

In addition to its mathematical attractiveness, there is some evidence that the logarithmic strain models the behavior of real materials over a wider range of deformation than other strain measures might. One study of vulcanized rubber suggests a good agreement with the logarithmic model over an approximate range of linear stretches $0.7 < \lambda < 1.3$ [2].

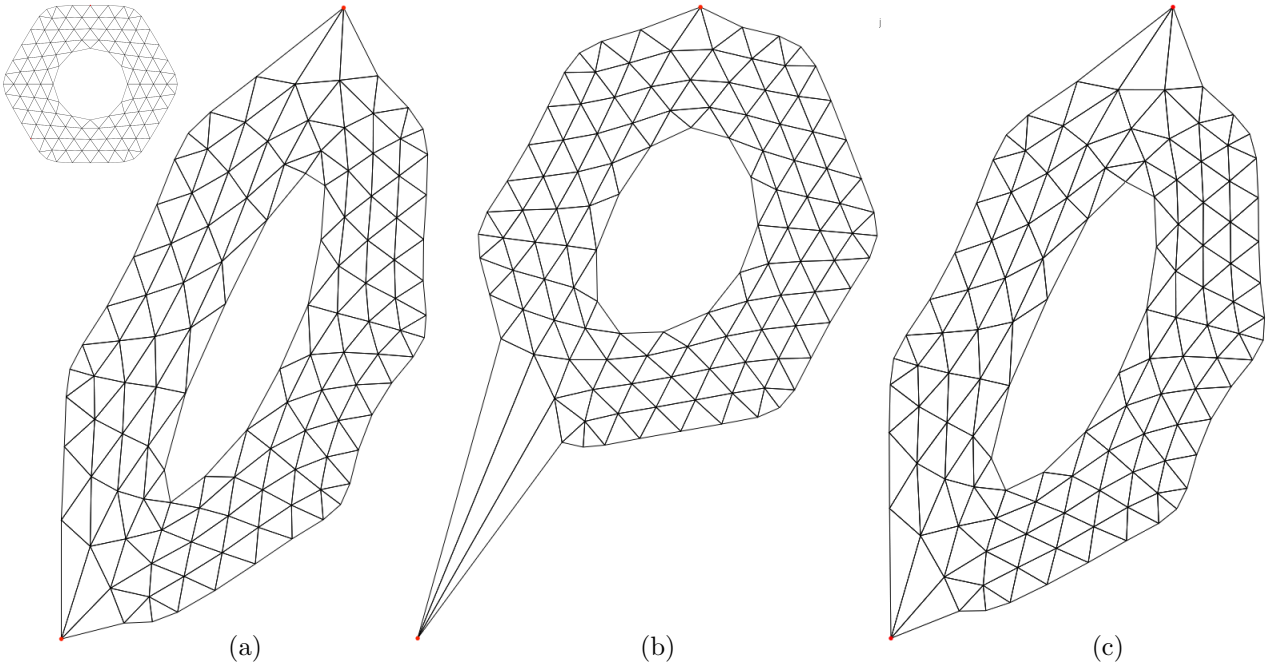
4.2.1 A hybrid strain energy

One potentially undesirable feature of the Hencky strain energy E_0 is its material non convexity for large strains; t $\log^2(x)$ is not convex for $x > e$ (see Figure 4.2). This can lead to unintuitive local energy minima, with distortion concentrated into small regions, and may also not reflect the behavior of real materials well, some of which even exhibit 'strain hardening' and may be better modeled at large deformations by exponential strain functions [23]. Fortunately, these limitations can be overcome easily, as our formulation admits *piecewise* analytic strain functions f. For example, a useful strain function is based on the function

$$f_{\mathrm{loglin}}(x) = \begin{cases} \frac{1}{2}\log(x) & 0 < x \leq e \\ \sqrt{ex} - \frac{1}{2} & x \geq e \end{cases} \tag{4.1}$$

This is C^1 function[2] for which $f^2_{\mathrm{loglin}}(x)$ is convex. See Figure 4.1 for a comparison of the Biot, Hencky, and 'LogLin' hybrid strain.

[2]Hence might show reduced order convergence in pathological cases when strains are close to e, which could remedied by choosing a C^2 function instead.



(a) (b) (c)

Figure 4.1: The same simple elastostatics problem is solved by minimizing three different strain energies, with an initial configuration (upper left) obtained by scaling the reference shape by a factor of 2 and fixing two points (red). The Biot model (a) converges to a nicely symmetric minimum, yet can admit inverted elements and is materially linear. The Hencky (logarithmic) model (b) converges to a local minimum which, while physically plausible, is undesirable from a geometry-processing point of view, due to its asymmetry and distance from a global minimizer. The hybrid model (c) uses a strain energy (Equation (4.1)) which is materially convex (that is, convex if $\mathbf{R}_\phi$ is constant), equivalent to a scaled Biot model for stretches greater than e, and is able to recover a more symmetric minimum while preserving all the other benefits of the logarithmic strain energy.

4.3 Matrix Functions

We consider a class of matrix functions defined as an extension of analytic functions by replacing a real argument with a square matrix and making use of the available Taylor series. For a thorough treatment, see the textbook by Higham[19]. In particular, we consider the matrix function corresponding to the natural logarithm on real numbers, $\log(x)$, which is analytic for $x \in (0, \infty)$. Consider a symmetric positive definite matrix $\mathbf{A}$, which admits a diagonalization $\mathbf{A} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T$, where $\mathbf{U} \in \mathrm{SO}(d)$ and $\mathbf{\Lambda}$ is a diagonal matrix with real, positive eigenvalues $\lambda_1, ..., \lambda_d$ on the diagonal. In this case, for example, the matrix logarithm can be simply written (and computed) as

$$\log(\mathbf{A}) = \mathbf{U} \log(\mathbf{\Lambda}) \mathbf{U}^T \tag{4.2}$$

where the logarithm of the diagonal matrix $\mathbf{\Lambda}$ is computed by taking logarithms of the diagonal elements.

4.3.1 The Derivative of a Matrix Function

If we presume $\mathbf{A} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T$ to be diagonalizable, we can derive the following expression for the derivative.

$$Df(\mathbf{A}) \cdot \mathbf{V} = \mathbf{U} \left[\Theta \odot (\mathbf{U}^T \mathbf{V} \mathbf{U}) \right] \mathbf{U}^T \tag{4.3}$$

where $\odot$ denotes an elementwise product [3] and Θ is a symmetric array of divided differences.

$$\Theta_{ij} = f[\lambda_i, \lambda_j] \doteq \begin{cases} \frac{f(\lambda_i) - f(\lambda_j)}{\lambda_i - \lambda_j} & \lambda_i \neq \lambda_j \\ f'(\lambda_i) & \lambda_i = \lambda_j \end{cases}$$

Note that Θ is not a tensor.

While divided differences are a familiar concept, their presence in derivatives of matrix functions contrasts with their usual appearances either as as approximate derivatives or in interpolation in two major ways. Firstly, they arise in *exact* expressions for derivatives, not approximations, so they must be evaluated to arbitrary precision. Further, divided differences with arbitrarily close arguments (as opposed to some potentially small but fixed increment) must be evaluated, again to machine precision.

For formulæ such as (4.3) to be useful in a computational setting, a general and stable method for computing the required divided differences to machine precision should be available. We present a simple and general method for computing these in Appendix A.3.1, which extends work such as that by Kahan and Fateman [26], who present the results for first divided differences of useful classes of functions.

[3] also known as a Schur product or Hadamard product

4.3.2 General Derivatives

Appendix A further considers expressions for arbitrary derivatives of matrix functions, and gives (4.3) as a special case. In this thesis we have focused on unitarily invariant norms, which leads to the relatively simple expressions for energy gradients and Hessians, which only require computation of *first* derivatives of matrix functions. However, for non-isotropic materials (hence an arbitrary norm appearing in the elastic energy) we cannot leverage this invariance and as such would need second derivatives of matrix functions to compute a Newton system. Similarly, we have focused on hyperelastic constitutive theories, in which stress can only depend on $\mathbf{C}$ (which is diagonalizable), but more general approaches might relax this to general analytic functions of $\mathbf{F}$ which may be degenerate; for this reason (and for expository purposes), we include expressions for arbitrary matrices.

4.4 The Elastic Energy

We will consider the elastic energy

$$E_f(\phi) \doteq \frac{1}{2}\int_{\mathcal{B}} ||f(\mathbf{C})||^2_{\alpha,\beta}\mathbf{dV} \tag{4.4}$$

where f is a matrix function induced by a piecewise analytic function. .

Of particular interest is the energy $E_0 \doteq \frac{1}{8}\int_{\mathcal{B}} ||\log(\mathbf{C})||^2_{\alpha,\beta}\mathbf{dV}$. [4]

See Figure 4.2 for an illustration of how the energy E_0 corresponds to stretching in a 1-dimensional element.

4.5 First and Second Variations of the Elastic Energy

We consider a variation δ_ψ in a deformation ϕ, and compute the variation in $E_f(\phi)$ for $\alpha = \beta = 1$

$$\delta_\psi E_f(\phi) = \int_{\mathcal{B}} \langle f(\mathbf{C}_\phi), \delta_\psi f(\mathbf{C}_\phi)\rangle \mathbf{dV}$$

Using (4.3), we have

$$\int_{\mathcal{B}} \langle \mathbf{U}f(\mathbf{\Lambda})\mathbf{U}^T, \mathbf{U}(\Theta \odot (\mathbf{U}^T\delta_\psi\mathbf{C}\mathbf{U}))\mathbf{U}^T\rangle \mathbf{dV},$$

where we have used the diagonalization $\mathbf{C} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T$, and where $\Theta_{ij} = f[\lambda_i, \lambda_j]$.

We require that the inner product is the standard trace inner product, in which case we have the identity

$$\langle A_1 \odot A_2, A_3\rangle = \sum_{i,j\in 1,\ldots,d} (A_1)_{ij}(A_2)_{ij}(A_3)_{ij},$$

[4]Note that the factor of $\frac{1}{8}$ arises because of the factor of $\frac{1}{2}$ in the definition of ϵ_0, which ensures that all Seth-Hill strain measures have the same linearization - see Section 1.4.4.

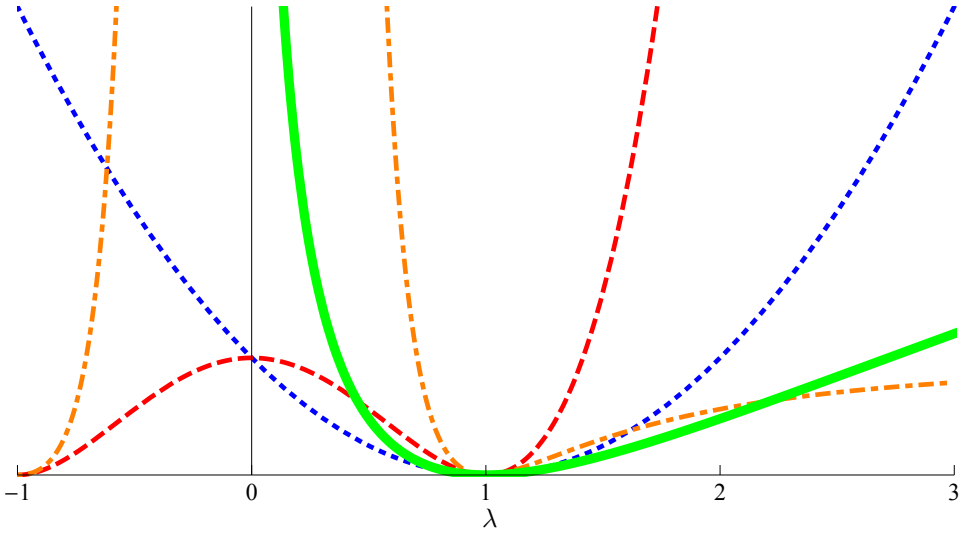


Figure 4.2: Illustration of energy in a one-dimensional element, stretched by a factor of λ. The logarithmic strain energy ($n = 0$) in green is compared to models using the Biot ($n = 1$, blue, dotted), Green-Lagrange($n = 2$, red, dashed), and Almansi ($n = -2$, orange, dot-dashed) strains . The logarithmic strain energy is unique here in that it provides an infinite energy barrier to inversion as well as growing without bound with λ, albeit with a loss of convexity for $\lambda > e$.

where we treat $\mathbf{A}_i, i = 1, 2, 3$ as matrices which may or may or not be representations of tensors.

From this identity it's apparent that only the diagonal entries of Θ_f, namely $(\Theta_f)_{ii} = f'(\lambda_i)$ enter, so the expression is equivalent to

$$\delta_\psi E_f(\phi) = \int_{\mathcal{B}} \langle p(\Lambda), \mathbf{U}^T(\delta_\psi \mathbf{C})\mathbf{U}\rangle \mathbf{dV} = \int_{\mathcal{B}} \langle p(\mathbf{C}), \delta_\psi \mathbf{C}\rangle \mathbf{dV},$$

where $p(x) \doteq f(x)f'(x)$. Finally, note that $\delta\mathbf{C} = \mathbf{F}^T\delta_\psi\mathbf{F} + \delta_\psi\mathbf{F}^T\mathbf{F}$ and that $p(\mathbf{C}) = p(\mathbf{C})^T$ to obtain

$$\delta_\psi E_f(\phi) = 2\int_{\mathcal{B}} \langle p(\mathbf{C}_\phi), \mathbf{F}_\phi^T\delta_\psi\mathbf{F}_\phi\rangle \mathbf{dV} = 2\int_{\mathcal{B}} \langle \mathbf{F}_\phi p(\mathbf{C}_\phi), \delta_\psi\mathbf{F}_\phi\rangle \mathbf{dV}. \tag{4.5}$$

Using this expression, we proceed to compute a second variation, again for clarity under the simplifying assumption $\alpha = \beta = 1$. We obtain

$$\delta^2_{\psi,\xi} E_f(\phi) = 2\int_{\mathcal{B}} \langle \delta_\xi\mathbf{F}_\phi, \delta_\psi\mathbf{F}_\phi\rangle_{p(\mathbf{C}_\phi)} \mathbf{dV} + 2\int_{\mathcal{B}} \langle \delta_\xi p(\mathbf{C}_\phi), \mathbf{F}_\phi^T\delta_\psi\mathbf{F}_\phi\rangle \mathbf{dV} \tag{4.6}$$

which expands to

$$\delta^2_{\psi,\xi} E_f(\phi) = 2\int_{\mathcal{B}} \langle \delta_\xi\mathbf{F}_\phi, \delta_\psi\mathbf{F}_\phi\rangle_{p(\mathbf{C}_\phi)} \mathbf{dV} + 2\int_{\mathcal{B}} \langle \Theta_p \odot \mathbf{U}_\phi^T(\mathbf{F}^T\delta_\xi\mathbf{F}_\phi + \delta_\xi\mathbf{F}_\phi^T\mathbf{F}_\phi)\mathbf{U}_\phi, \mathbf{U}_\phi\mathbf{F}_\phi^T\delta_\psi\mathbf{F}_\phi\rangle \mathbf{dV}$$

4.6 Isotropic Material Parameters

If $\alpha \neq \beta$, we carry out similar manipulations to those in Section 4.5 to obtain corresponding expressions for the first and second variations of the energy. The first variation is given by

$$\delta_\psi E_f(\phi) = \int_{\mathcal{B}} \langle (\alpha^2 - \beta^2) \tfrac{\operatorname{tr} f(\mathbf{C})}{d} f'(\mathbf{C}) + \beta^2 p(\mathbf{C}), \delta_\psi F_\phi \rangle \mathbf{dV} \tag{4.7}$$

and the second variation by

$$\begin{aligned}
\delta^2_{\psi,\xi} = {} & 2\beta^2 \int_{\mathcal{B}} \langle \delta_\xi \mathbf{F}_\phi, \delta_\psi \mathbf{F}_\phi \rangle_{p(\mathbf{C}_\phi)} \mathbf{dV} + \\
& 2\beta^2 \int_{\mathcal{B}} \langle \delta_\xi p(\mathbf{C}_\phi), \mathbf{F}_\phi^T \delta_\psi \mathbf{F}_\phi \rangle \mathbf{dV} + \\
& 2(\alpha^2 - \beta^2) \tfrac{\operatorname{tr} f(\mathbf{C}_\phi)}{d} \int_{\mathcal{B}} \langle \delta_\xi \mathbf{F}_\phi, \delta_\psi \mathbf{F}_\phi \rangle_{f'(\mathbf{C}_\phi)} \mathbf{dV} + \\
& 2(\alpha^2 - \beta^2) \tfrac{\operatorname{tr} f(\mathbf{C}_\phi)}{d} \int_{\mathcal{B}} \langle \delta_\xi f'(\mathbf{C}_\phi), \mathbf{F}_\phi^T \delta_\psi \mathbf{F}_\phi \rangle \mathbf{dV} + \\
& 2(\alpha^2 - \beta^2) \tfrac{\operatorname{tr} f(\mathbf{C}_\phi)}{d} \int_{\mathcal{B}} \langle f'(\mathbf{C}_\phi), \mathbf{F}_\phi^T \delta_\xi \mathbf{F}_\phi \rangle \langle f'(\mathbf{C}_\phi), \mathbf{F}_\phi^T \delta_\psi \mathbf{F}_\phi \rangle \mathbf{dV}
\end{aligned} \tag{4.8}$$

Figure 4.3 shows a simple example of the effect of varying these parameters.

Note that the expressions here only require f to have two derivatives [5].

4.6.1 Discrete Variations

We restrict our attention to ϕ being an affine transformation of a reference d-simplex (a triangle $\triangle = \triangle_{ijk}$ for $d = 2$ and a tetrahedron $\triangle = \triangle_{ijkl}$ for $d = 3$) to a deformed configuration, as described by mappings of reference to deformed vertices.

In this case, as in Section 2.3.1, the deformation gradient over a d-simplex is given by

$$\mathbf{F}_\phi = \frac{1}{\text{vol}} \sum_{s \in \triangle} X_s \otimes \text{volGrad}_s,$$

A sum of outer products of deformed vertex positions and reference element volume gradients. With this observation, the expressions in Section 4.6 can be used to assemble the gradient and Hessian of the energy. Partial sample code is provided in Appendix B.3.

For simplicity we have written a volume-weighted sum, when more accurately we use a *mass*-weighted sum. This becomes important in the shape interpolation application, discussed further in Section 4.8.2,

[5] but be aware that this depends critically on only evaluating matrix functions of diagonalizable arguments, which is always the case for the examples in this chapter, but might not be so for extensions, as discussed in Section 4.3.2. In general, computing $f(A)$ can require d derivatives, where d is the dimension of A, and computing the nth derivative raises this number to the nth power.

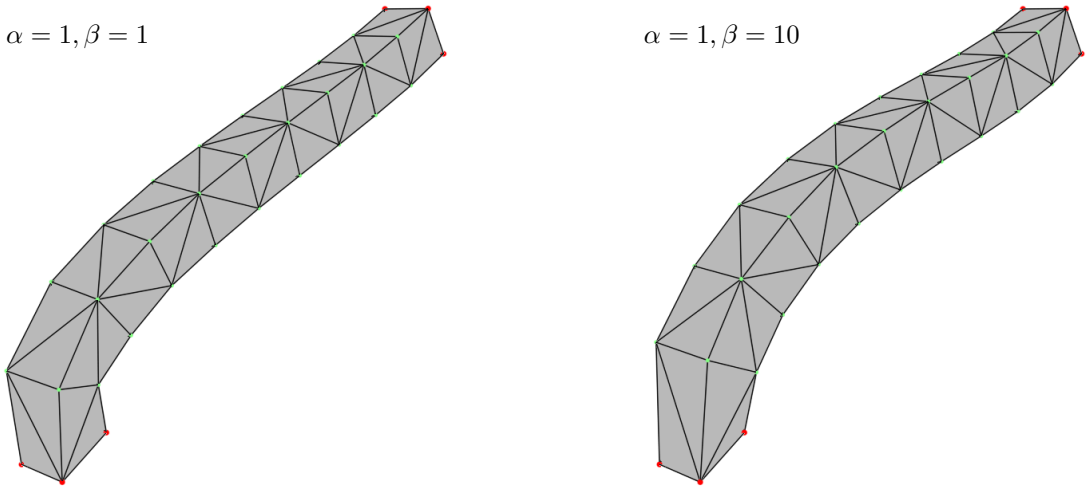


Figure 4.3: The same elastostatics simulation with different material parameters. The material parameters α and β control the initially straight bar's resistance to volume change and isochoric deformation, respectively.

where we include a density factor to ensure that multiple reference states represent bodies with the same mass.

Minimizing the discrete energy is a nonlinear least squares problem, and this additional structure can be exploited computationally.

4.7 Gauss-Newton Minimization

Using the expressions derived above, the elastic energy can be minimized using standard tools based on Newton's method [6, 4].

For many applications, the additional computational complexity in computing the full Hessian may not be warranted, especially when the solution is outside the domain of rapid convergence and one must rely on the global convergence properties of a trust-region solver. Fortunately, a practical quasi-Newton method is well-known in the numerical optimization for minimizing objectives of the form

$$S(x) = \frac{1}{2} \sum_i \left(r_i(x)\right)^2 \tag{4.9}$$

Indeed, this is a nonlinear least squares problem which can be solved with the *Gauss-Newton method*.

The Gauss-Newton method uses an approximate Hessian computed by neglecting terms involving $\nabla^2 r_i(x)$,

replacing the residuals $r_i(x)$ with their linear approximations.

$$\nabla^2 S(x) \approx \sum_i \left(\nabla r_i^T(x) \nabla r_i(x)\right)$$

The main advantages of the Gauss-Newton method over Newton's method are firstly that it only requires knowledge of the gradient of the residuals and secondly that it always produces a descent search direction [46]. See Figure 4.4 for comparisons of the Gauss-Newton and Newton methods in two simple scenarios.

This robustness and relative ease of implementation has led to specific cases of this method to be employed in elastic energy minimization contexts before, under different names and derivations such as 'local/global minimization' [6][36].

In our case, using a Gauss-Newton solver amounts to neglecting the second term in (4.6) and the corresponding terms in (4.8). See Appendix B.3.

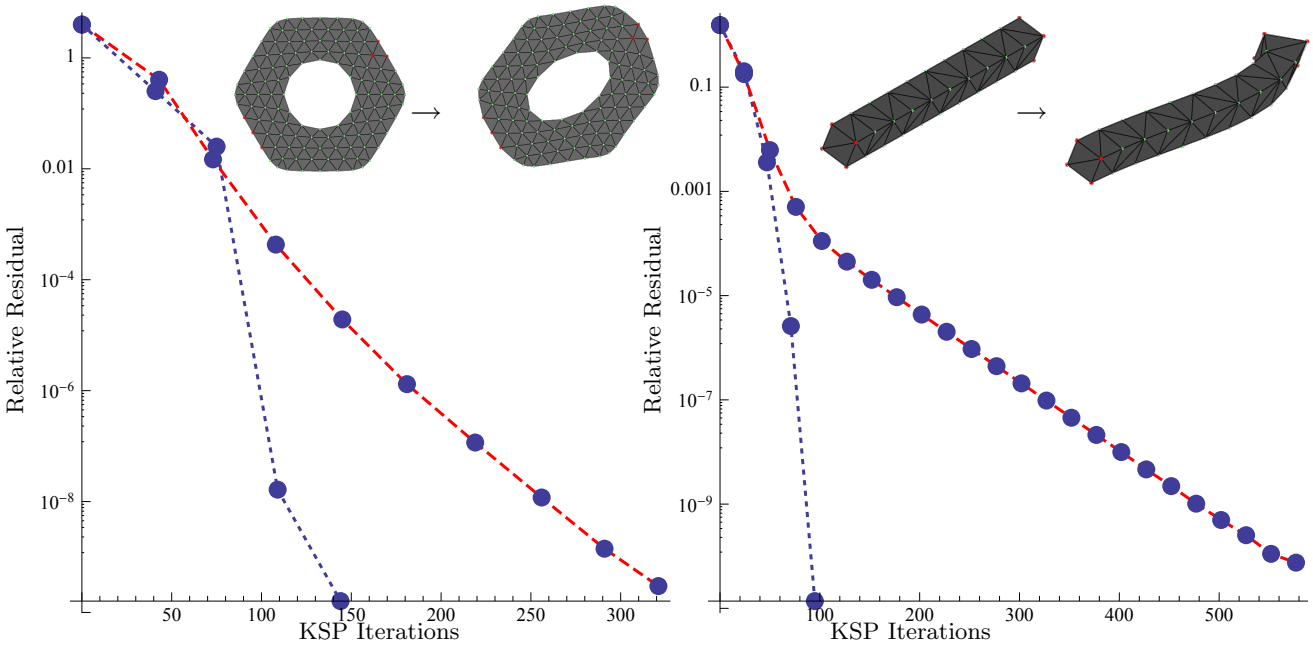


Figure 4.4: Comparisons of the cumulative number of linear solver inner iterations required to minimize the energy in two simple elastostatics scenarios, using the PETSc KSP linear solver. In the planar case on the left, using Newton's method (blue, dotted) requires fewer iterations but the additional complexity required may make the Gauss-Newton method (red, dashed) the attractive choice. For the volumetric mesh on the right, the energy landscape is complicated enough to make the Gauss-Newton method less effective.

[6]Formulating the energy with the Biot strain, $n = 1$, it can be shown that neglecting terms involving the derivatives of the polar factor R by 'freezing rotations' gives rise to the same Hessian approximation as a Gauss-Newton method.

4.8 Implementation and Examples

Implementing an algorithm to minimize the elastic energy requires repeated assembly of a global gradient vector and (approximate) Hessian matrix, solution of a linear system, and iteration until convergence. This is accomplished with a standard iteration over elements, computing local contributions and adding them to the global data structures.

4.8.1 Computing Matrix Functions and Derivatives

To compute the elastic energy and its derivatives based on the expressions in Section 4.5 requires fast diagonalization of small positive semidefinite matrices defining $\mathbf{C}$ locally. We use a modern hybrid diagonalization technique, specialized for $d = 2, 3$ [34].

We must also compute first divided differences of scalar functions [7]. Divided differences are notoriously difficult to work with, yet are required for computation of the first and second derivatives of the elastic energy. The main difficulty in computing with divided differences is that as their arguments become arbitrarily close, a naive computation exhibits an arbitrarily severe cancellation error, and in the elastic setting where a typical use case often has minuscule deformation throughout most of a large body, local stretches λ_i all very close to 1 are commonly encountered. Fortunately, we introduce a new, general stabilization technique, which extends existing schemes for the first derivatives of the exponential and logarithm [26] to arbitrary divided differences of general analytic functions, and apply it to the particular case of the matrix logarithm. See Appendix A for details, in particular Section A.3.1.

4.8.2 Elastostatics

We can use our minimization method of choice to solve elastostatics problems. Figure 4.5 and the embedded video there show an interactive elasticity simulation of the response of a planar body to given fixed node positions, highlighting the usefulness of our energy's barrier against inversion.

4.8.3 Elastodynamics

As described in Section 4.8.3, a simple modification to the energy [29][8] allows us to implement a Fully Variational Integrator (FVI), using essentially the same code to generate elastodynamics simulations with excellent conservation properties. Figure 4.6 and the embedded video demonstrate how our energy provides robustness against element inversion when high initial compressive velocities are prescribed on the boundary of a body.

[7] And for situations with less invariance, require second order divided differences or even divided differences of matrix functions, which are discussed in Appendix A.

[8] Note that our E becomes their W.

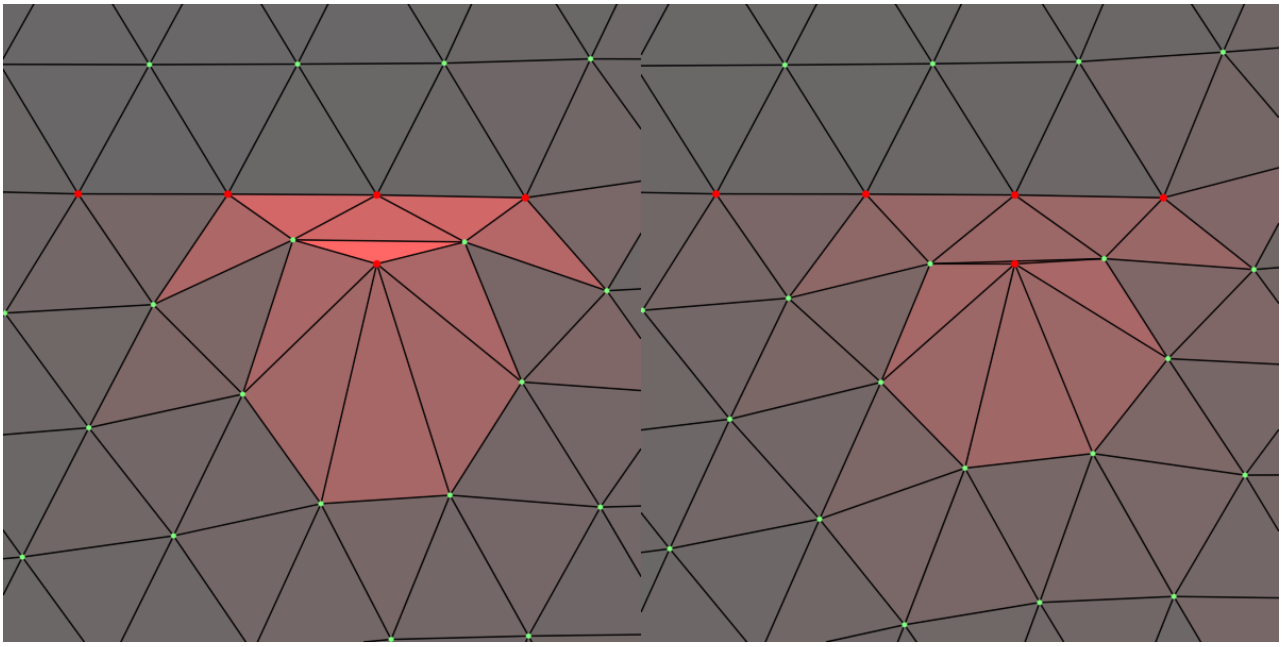


Figure 4.5: A frame from an interactive elasticity simulation comparing logarithmic and Biot strain-based energies, demonstrating the former's usefulness in avoiding degenerate configurations. Elements are colored according to the norm of the local strain. See the embedded movie in the pdf version of this document.

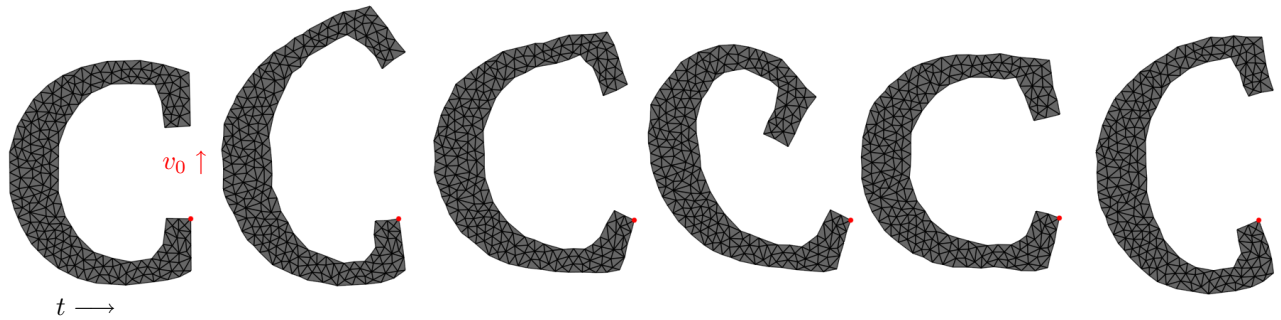
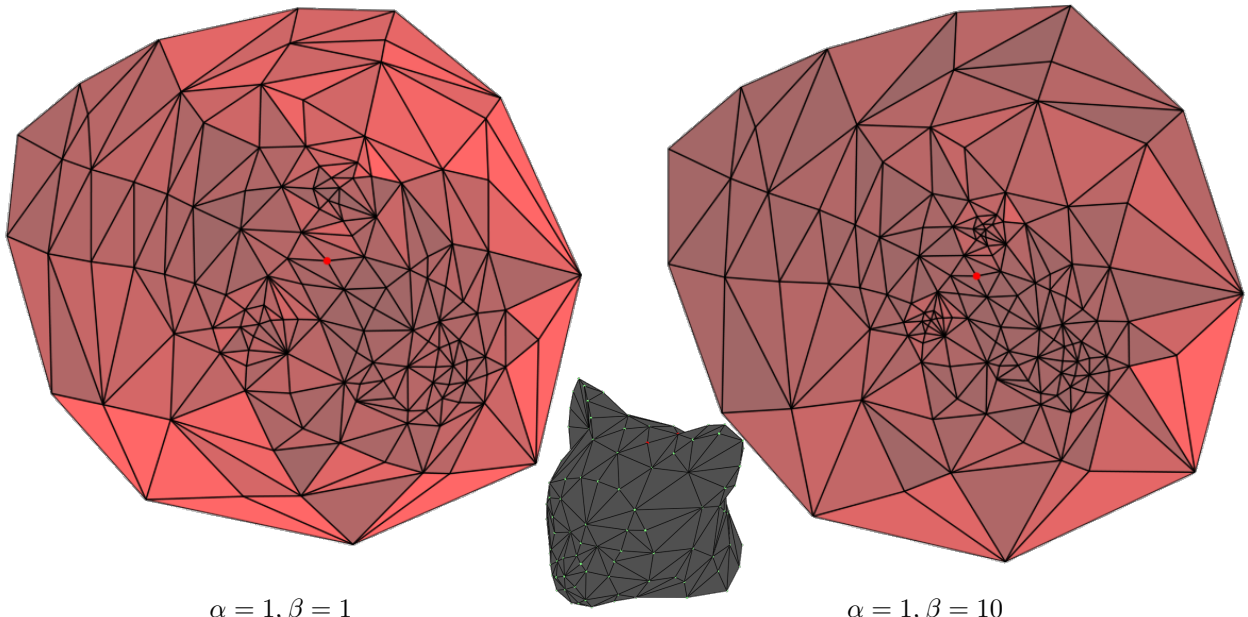


Figure 4.6: Frames from a dynamics simulation of a high initial velocity on portion of the boundary of an elastic object. The logarithmic strain energy and FVI allow simulation of an undamped compressive wave without inversion. Using the Biot strain energy , for example, limits the initial velocity permitted (regardless of choice of time step) if inverted elements are to be avoided; in this example elements would invert almost immediately. See the embedded movie in the pdf version of this document, which shows the simulation at two framerates.

4.8.4 Surface Parameterization

We can parameterize a surface by minimizing the intrinsic elastic energy required to map it to the plane. Compared to other methods, such as the methods in Chapters 2 and 3, minimization of the logarithmic strain energy has the advantage of guaranteeing a proper parameterization with no local inversions and a gradual 'hardening' against extreme compression than the method of Chapter 3 lacks. However, an admissable initial parameterization is required, as the logarithmic model does not assign energy to inverted elements. See Figure 4.7 for an example.



$\alpha = 1, \beta = 1$ $\alpha = 1, \beta = 10$

Figure 4.7: We map the surface of a topological disk to the plane by minimizing our elastic energy with two different sets of isotropic material parameters. By virtue of the energy barrier against inversion, given an admissible initial parameterization, no ad-hoc 'flipped triangle repair' stage is required. Redness corresponds to quasi conformal distortion.

4.8.5 Shape Interpolation

Shape interpolation is achieved by minimizing a weighted sum of energies with respect to multiple reference states. See Figure 4.8 for an example. As noted above, we include a density factor in the energy to account for differing reference volumes of discrete elements. With this, the energy is truly reversible; it is symmetric with respect to which of two states is chosen as the reference. Thus, no additional symmetrization is required to extend this technique to minimize the length of a path connecting an arbitrary number of intermediate states between two given endpoints, for example in the technique employed by Kilian et al. [32]. While the elastic energy only agrees to second order with true geodesic distance on the space of possible deformations, it is the optimal one amongst the class based on Seth-Hill strain measures [52].

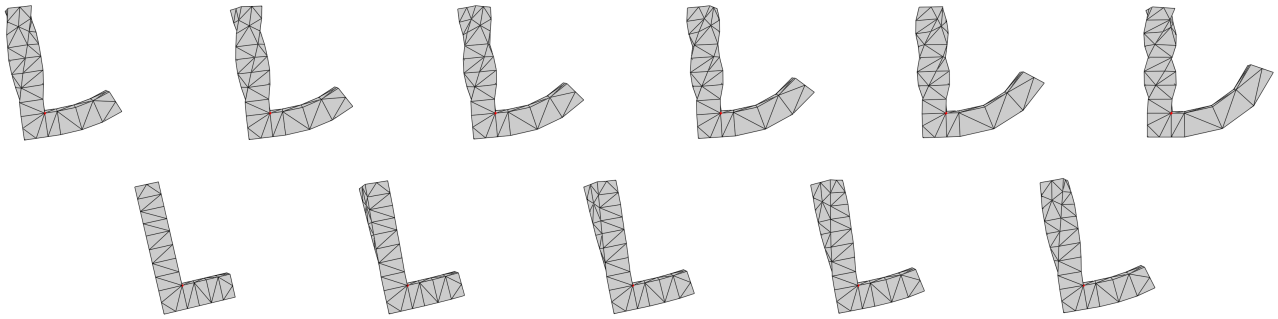


Figure 4.8: The poses between two reference states, shown in the upper left and lower right, are generated by minimizing a weighted sum of elastic energies. As such, the interpolated poses are guaranteed to have no inverted elements. Further, since the energy is reversible, the interpolation between two states shown here can be extended without further symmetrization to minimization between an arbitrary number of states along a path in shape space.

4.9 Discussion

We have shown how employing the logarithmic strain measure lends many desirable properties to a wide range of algorithms in elastic simulation and geometry processing. In doing so we have extended and unified existing methods based on energies quadratic in Seth-Hill strain measures. Additional complexity is introduced by use of the matrix logarithm, but we have developed efficient formulations for computing this function and its derivatives by use of a diagonalization and stable evaluation of divided differences. Of particular note is that by utilizing the general theory of analytic matrix functions, the matrix function f is by no means limited to the logarithm, and thus the methods here can be used to greatly broaden material models used for elasticity-based geometry processing.

Bibliography

[1] R Abraham, Jerrold E. Marsden, and R. Ratiu. *Manifolds, Tensor Analysis, and Applications*. Springer-Verlag, New York, 2nd edition, 1988.

[2] L. Anand. On H. Hencky's approximate strain-energy function for moderate deformations. *J. Appl. Mech. ASME*, 46:78–82, 1979.

[3] Uri M Ascher and Eddy Boxerman. On the modified conjugate gradient method in cloth simulation. *The Visual Computer*, 19(7-8):526–531, 2003.

[4] Satish Balay, Jed Brown, , Kris Buschelman, Victor Eijkhout, William D. Gropp, Dinesh Kaushik, Matthew G. Knepley, Lois Curfman McInnes, Barry F. Smith, and Hong Zhang. PETSc users manual. Technical Report ANL-95/11 - Revision 3.2, Argonne National Laboratory, 2011.

[5] David Baraff and Andrew Witkin. Large Steps in Cloth Simulation. *ACM Transactions on Graphics*, 13:43–54, 1998.

[6] Steve Benson, Lois Curfman McInnes, Jorge Moré, Todd Munson, and Jason Sarich. TAO user manual (revision 1.10.1). Technical Report ANL/MCS-TM-242, Math. and Comp. Sc. Division, Argonne Nat. Lab., 2010. http://www.mcs.anl.gov/tao.

[7] M. A. Biot. Theory of Elasticity with Large Displacements and Rotations. In *Proc. Fifth Int. Cong. Appl. Math.*, pages 117–122. John Wiley & Sons, 1938.

[8] Mario Botsch, Mark Pauly, Markus Gross, and Leif Kobbelt. Primo: coupled prisms for intuitive surface modeling. In *Proceedings of the fourth Eurographics symposium on Geometry processing*, SGP '06, pages 11–20, Aire-la-Ville, Switzerland, Switzerland, 2006. Eurographics Association.

[9] Isaac Chao, Ulrich Pinkall, Patrick Sanan, and Peter Schröder. A simple geometric model for elastic deformations. *ACM Trans. Graph.*, 29:38:1–38:6, July 2010.

[10] U. Clarenz, N. Litke, and M. Rumpf. Axioms and variational problems in surface parameterization. *Computer Aided Geometric Design*, 21(8):727–749, 2004.

[11] Keenan Crane. *Conformal Geometry Processing*. PhD thesis, California Institute of Technology, 2013.

[12] Keenan Crane, Ulrich Pinkall, and Peter Schröder. Spin transformations of discrete surfaces. *ACM Trans. Graph.*, 40, 2011.

[13] Philip I. Davies and Nicholas J. Higham. A Schur-Parlett Algorithm for Computing Matrix Functions. *SIAM Journal on Matrix Analysis and Applications*, 25(2):464, 2003.

[14] M.P. do Carmo. *Riemannian Geometry*. Mathematics (Boston, Mass.). Birkhauser Verlag AG, 1992.

[15] Olaf Etzmuß, Michael Keckeisen, and Wolfgang Straßer. A Fast Finite Element Solution for Cloth Modeling. In *Proc. Pac. Graph.*, pages 244–251, 2003.

[16] Michael S Floater and Kai Hormann. Surface Parameterization : a Tutorial and Survey. In *Advances in Multiresolution for Geometric Modelling*, pages 157–186. Springer, 2005.

[17] Gaël Guennebaud, Benoît Jacob, et al. Eigen v3. http://eigen.tuxfamily.org, 2010.

[18] Nicholas J. Higham. Computing the Polar Decomposition—With Applications. *SIAM J. Sci. Stat. Comp.*, 7(4):1160–1174, 1986.

[19] Nicholas J. Higham. *Functions of Matrices: Theory and Computation*. SIAM, Philadelphia, 2008.

[20] Nicholas J. Higham and Awad H. Al-Mohy. *Computing matrix functions*, volume 19. Cambridge University Press, May 2010.

[21] R. Hill. Constitutive Inequalities for Isotropic Elastic Solids Under Finite Strain. *Proc. Royal Soc. A: Math., Phys. and Eng. Sc.*, 314(1519):457–472, January 1970.

[22] Rodney Hill. On Constitutive Inequalities for Simple Materials-I. *J. Mech. Phys. Solids*, 16:229–242, 1968.

[23] Cornelius O. Horgan and Michael G. Smayda. The importance of the second strain invariant in the constitutive modeling of elastomers and soft biomaterials. *Mechanics of Materials*, 51:43–52, August 2012.

[24] Geoffrey Irving, Joseph Teran, and Ron Fedkiw. Invertible Finite Elements for Robust Simulation of Large Deformations. In *Proc. Symp. Comp. Anim.*, pages 131–140, 2004.

[25] Alec Jacobson, Ilya Baran, Ladislav Kavan, Jovan Popović, and Olga Sorkine. Fast automatic skinning transformations. *ACM Transactions on Graphics (proceedings of ACM SIGGRAPH)*, 30(4):77:1–77:10, 2012.

[26] W. Kahan and R. J. Fateman. Symbolic computation of divided differences. *SIGSAM Bull.*, 33:7–28, June 1999.

[27] Liliya Kharevych, Boris A. Springborn, and Peter Schröder. Discrete Conformal Mappings via Circle Patterns. *ACM Transactions on Graphics*, 25(2):412–438, 2006.

[28] Liliya Kharevych, Weiwei, Yiying Tong, Eva Kanso, Jerrold Marsden, Peter Schröder, and Mathieu Desbrun. Geometric, Variational Integrators for Computer Animation. In *Proc. Symp. Comp. Anim.*, pages 43–52, 2006.

[29] Lilya Kharevych, Peter Schröder, Y. Tong, E. Kanso, Jerrold E. Marsden, and Mathieu Desbrun. Geometric , Variational Integrators for Computer Animation. *Eurographics / ACM SIGGRAPH Symp. on Comp. Anim.*, 2006.

[30] Andrei Khodakovsky, Nathan Litke, and Peter Schröder. Globally smooth parameterizations with low distortion. *ACM Trans. Graph.*, 22(3):350–357, July 2003.

[31] Ryo Kikuuwe, Hiroaki Tabuchi, and Motoji Yamamoto. An Edge-based Computationally Efficient Formulation of Saint Venant-Kirchhoff Tetrahedral Finite Elements. *ACM Trans. Graph.*, 28(1):1–13, 2009.

[32] Martin Kilian, Niloy J. Mitra, and Helmut Pottmann. Geometric modeling in shape space. *ACM Trans. Graph.*, 26, July 2007.

[33] Martin Kilian, Niloy J. Mitra, and Helmut Pottmann. Geometric Modeling in Shape Space. *ACM Trans. Graph.*, 26(3):#64, 1–8, 2007.

[34] Joachim Kopp. Efficient numerical diagonalization of hermitian 3×3 matrices. *Int. J. Mod. Phys. C*, 19:523–548, 2008.

[35] Yaron Lipman. Bounded distortion mapping spaces for triangular meshes. *ACM Transactions on Graphics*, 31(4):1–13, July 2012.

[36] Ligang Liu, Lei Zhang, Yin Xu, Craig Gotsman, and Steven J. Gortler. A Local/Global Approach to Mesh Parameterization. *Computer Graphics Forum*, 27(5):1495–1504, July 2008.

[37] Ligang Liu, Lei Zhang, Yin Xu, Craig Gotsman, and Steven J. Gortler. A Local/Global Approach to Mesh Parameterization. *Comp. Graph. Forum*, 27(5):1495–1504, 2008.

[38] Rolf Mahnken. Anisotropy in geometrically non-linear elasticity with generalized Seth-Hill strain tensors projected to invariant subspaces. *Comm. Num. Meth. Eng.*, 21(8):405–418, March 2005.

[39] Jerrold E. Marsden and Thomas J. R. Hughes. *Mathematical foundations of elasticity*. Dover, New York, New York, USA, 1994.

[40] Jerrold E. Marsden and Matthew West. Discrete Mechanics and Variational Integrators. *Acta Numerica*, 10:357–514, 2001.

[41] Mark Meyer, Mathieu Desbrun, Peter Schröder, and Alan H. Barr. Discrete Differential-Geometry Operators for Triangulated 2-Manifolds. In H. C. Hege and K. Polthier, editors, *Visualization and Mathematics III*, Mathematics and Visualization, pages 113–134. Springer, 2002.

[42] Matthias Müller, Julie Dorsey, Leonard McMillan, Robert Jagnow, and Barbara Cutler. Stable Real-Time Deformations. In *Proc. Symp. Comp. Anim.*, pages 49–54, 2002.

[43] Matthias Müller, Bruno Heidelberger, Matthias Teschner, and Markus Gross. Meshless Deformations Based on Shape Matching. *ACM Trans. Graph.*, 24(3):471–478, 2005.

[44] Katta J. Murty and Santosh N. Kabadi. Some NP-complete problems in quadratic and nonlinear programming. *Mathematical Programming*, 39(2):117–129, 1987.

[45] Andriy Myronenko and Xubo Song. On the Closed-Form Solution of the Rotation Matrix Arising in Computer Vision Problems. `http://arxiv.org/abs/0904.1613v1`, 4 2009.

[46] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer, New York, New York, USA, 2 edition, 2006.

[47] M. Ortiz, R. a. Radovitzky, and E. a. Repetto. The computation of the exponential and logarithmic mappings and their first and second linearizations. *International Journal for Numerical Methods in Engineering*, 52(12):1431, December 2001.

[48] Xavier Pennec. Left-invariant Riemannian elasticity: a distance on shape diffeomorphisms? In *Proc. of the Int. Workshop on the Math. Found. of Comp. Anat.*, 2006.

[49] Ulrich Pinkall and Konrad Polthier. Computing Discrete Minimal Surfaces and Their Conjugates. *Experiment. Math.*, 2(1):15–36, 1993.

[50] E. Polak and G Ribière. Note sur la convergence de méthodes de directions conjuguèes. *Revue Française dInformatique et de Recherche Opérationnelle*, 16:35–43, 1969.

[51] C.C̃. Rankin and F.Ã. Brogan. An Element Independent Corotational Procedure for the Treatment of Large Rotations. *ASME J. Press. Valve Techn.*, 108(2):165–174, 1986.

[52] P Rougée. A Geometrical Interpretation of the Tensorial Characteristics of Finite Strains. *Sol. Mech. and App.*, 82:109–122, 2002.

[53] Rüdiger Schmedding and Matthias Teschner. Inversion Handling for Stable Deformable Modeling. *Vis. Comp.*, 24(7-9 (CGI 2008 Special Issue)):625–633, 2008.

[54] B.R. Seth. Generalised strain measure with application to physical problems. In *Second Order Effects in Elasticity, Plasticity and Fluid Dynamics*, pages 162–171. Macmillam, 1964.

[55] Alla Sheffer, Emil Praun, and Kenneth Rose. Mesh Parameterization Methods and Their Applications. *Foundations and Trends in Computer Graphics and Vision*, 2(2):105–171, 2006.

[56] Olga Sorkine and Marc Alexa. As-rigid-as-possible surface modeling. In *Proceedings of the fifth Eurographics symposium on Geometry processing*, pages 109–116, Aire-la-Ville, Switzerland, Switzerland, 2007. Eurographics Association.

[57] Olga Sorkine and Marc Alexa. As-rigid-as-possible Surface Modeling. In *Proc. Symp. Geom. Proc.*, pages 109–116, 2007.

[58] Boris Springborn, Peter Schröder, and Ulrich Pinkall. Conformal Equivalence of Triangle Meshes. *ACM Transactions on Graphics*, 27(3), 2008.

[59] C. Truesdell and W. Noll. *The Nonlinear Field Theories of Mechanics*. Springer-Verlag, 1965.

[60] Charles F. Van Loan. A study of the matrix exponential. *Numerical Analysis Report No. 10, University of Manchester, Manchester, UK, August 1975. Reissued as MIMS EPrint 2006.397, Manchester Institute for Mathematical Sciences, The University of Manchester, UK, November 2006.*, 1975.

Appendix A

Arbitrary Order Derivatives of General Matrix Functions

We present simple expressions for arbitrary order derivatives of matrix functions and specialize them to some cases useful in Chapter 4.

Let $f : \mathbb{R} \to \mathbb{R}$ be a piecewise analytic function. f can be extended to a function $f : \mathbb{C}^{d\times d} \to \mathbb{C}^{d\times d}$ on square matrices, replacing a scalar Taylor series by one of matrices. This extension is compatible with any similarity transformation, in that the identity

$$Xf(A)X^{-1} = f(XAX^{-1}) \tag{A.1}$$

holds for any invertible $X \in \mathbb{C}^{d\times d}$. Thus, if A has Jordan normal form $A = ZJZ^{-1}$, then $f(A) = Zf(J)Z^{-1}$ and we may define f by its operation on elementary Jordan blocks. See the textbook by Higham [19] for alternative equivalent definitions.

Working from Van Loan's maxim, "Anything that the Jordan Decomposition can do, the Schur Decomposition can do better!" [60], we also note his equivalent expression for an upper triangle square matrix. If $T = QAQ^H$ has the form

$$T = \begin{bmatrix} \lambda_1 & t_{12} & \cdots & t_{1d} \\ & \lambda_2 & \cdots & t_{2d} \\ & & \ddots & \vdots \\ & & & \lambda_d \end{bmatrix}, \tag{A.2}$$

then

$$(f(T))_{ij} = \sum_{i=s_1<s_2<\cdots<s_m=j} t_{s_1 s_2} t_{s_2 s_3} \cdots t_{s_{m-1} s_m} f[\lambda_{s_1}, \lambda_{s_2}, \ldots, \lambda_{s_m}] \tag{A.3}$$

We let a zeroth order divided difference just be a function evaluation, $f[x] \doteq f(x)$. A first order divided

difference is defined

$$f[x,y] = f[y,x] \doteq \begin{cases} \frac{f[x]-f[y])}{x-y} & x \neq y \\ f'(x) & x = y \end{cases}$$

Noting that the expression is symmetric with regard to its arguments and using the convention $f[x] = f(x)$, higher order divided differences are defined recursively. Without loss of generality, by permutation of the arguments either $x_k \neq x_{k+1}$ or $x_{k+1} = x_k = \cdots = x_2 = x_1$:

$$f[x_1, x_2, ..., x_{k-1}, x_k, x_{k+1}] = \begin{cases} \frac{f[x_1,x_2,...,x_{k-1},x_{k+1}]-f[x_1,x_2,...,x_{k-1},x_k]}{x_{k+1}-x_k} & x_k \neq x_{k+1} \\ \frac{1}{k!}f^{(k)}(x) & x_{k+1} = x_k = \cdots = x_1 \end{cases}, \quad k \geq 0$$

The expression (A.3) becomes useful for our purposes once it is generalized to *block* upper triangular matrices, which we do with an appropriate generalization of divided differences to matrix arguments, detailed in Section A.1.1

The expression (A.3) is not directly useful for large matrices due the exponential number of terms, but the Schur-Parlett algorithm described by Higham and Davies [13] reduces its computation to $O(n^3 m)$ operations, where m is the size of the largest cluster of eigenvalues. The proof in Section A.1.2 leverages the approach of that algorithm.

A.1 Functions of Block Upper Triangular Matrices

Due to the property (A.1), it suffices to define matrix functions in any suitable basis. For computational purposes, we are free to use any stably-computable and computationally expedient basis, such as that produced by a block Schur decomposition. For theoretical purposes, the Jordan canonical form is useful. To deal with these cases and to allow for simple derivation of expressions for derivatives of matrix functions, we derive a completely general description of matrix functions of block (upper) triangular matrices.

A.1.1 Generalized Divided Differences

We can generalize the definition of a divided difference to matrix arguments as follows. Let X_i be matrices of dimensions $m_i \times n_i$ and $V_{i,i+1}$ be matrices of dimension $n_i \times m_{i+1}$

We define $f[X] \doteq f(X)$ and recursively $f[X_1, X_2, ..., X_m]_{V_{1,2},V_{2,3},...,V_{m-1,m}}$ as the solution C to the equation

$$X_1 C - C X_m = f[X_1, X_2, ..., X_{m-1}]_{V_{1,2},V_{2,3},...,V_{m-1,m}} V_{m-1,m} - V_{1,2} f[X_2, X_3 ..., X_m]_{V_{2,3},V_{3,4},...,V_{m-1,m}} \quad \text{(A.4)}$$

with the additional property that C is continuous with respect to all the X_i and $V_{i,i+1}$. Note that unlike

scalar divided differences, generalized divided differences are not invariant (or even in general defined) under permutations of their arguments. As in the scalar case, we can equivalently define $f[X,Y]_V$ as the solution C to $XC - CY = f[X]V + Vf[Y]$ and $f[X,Y,Z]_{V,W} = (f[X,\cdot]_V)[Y,Z]_W$, and so on.

Although this has the form of a Sylvester equation, which for a general right hand side only has a unique solution here when X_1 and X_m have disjoint spectra, the special structure of the right hand side guarantees a solution, which we can write out explicitly for an analytic function $f(A) = \sum a_i A^k$ as

$$C = \sum a_i \sum_{\sum \alpha_j = k-m+1} X_1^{\alpha_1} V_{12} X_2^{\alpha_2} V_{23} \cdots V_{m-1,m} X_m^{\alpha_m}$$

which we can show converges as each term is bounded by the respective symmetrized term in the (convergent) series for the derivative of a polynomial function, which sum to give the series for $Df(A) \cdot (V_{12}, ..., V_{m-1,m})$.

Further, this solution is the only one which satisfies the continuity condition.

Some special cases are important for the sequel. Firstly, the generalization of the scalar expression

$$f[\overbrace{x, x, ..., x}^{k+1 \text{ copies}}] = \frac{1}{k!} f^{(k)}(x)$$

is

$$\sum_{\sigma \in \mathbb{S}_k} f[\overbrace{X, X, \ldots, X}^{k+1 \text{ copies}}]_{V_{\sigma_1}, ..., V_{\sigma_k}} = D^{(k)} f(X) \cdot (V_1, V_2, ..., V_k).$$

Secondly, if the spectra of X_1 and X_m are distinct (as they would be for the type of block Schur decomposition employed by the Schur-Parlett algorithm), then divided differences can be computed with standard methods for the solution of Sylvester equations.

A.1.2 Main Theorem

Theorem If T is a block upper triangular matrix with nonzero blocks T_{ij}, then

$$(f(T))_{ij} = \sum_{i = s_1 < s_2 < \cdots < s_m = j} f[T_{s_1 s_1}, T_{s_2 s_2}, \ldots, T_{s_m s_m}]_{T_{s_1 s_2}, T_{s_2 s_3}, ..., T_{m-1,m}} \tag{A.5}$$

using the definition of generalized divided differences from the previous section.

Proof Let $F \doteq f(T)$. The diagonal blocks of F are given by

$$F_{ii} = f(T_{ii}) = f[T_{ii}]$$

As can be seen by directly from the series form of f.

A basic property of matrix functions is that they commute with their arguments,

$$TF - FT = 0$$

Also, a matrix function is a continuous function of its argument, and in particular a continuous function of each T_{ij}.

Decompose $T = D + N$, where D is block diagonal and N is block-nilpotent (block upper triangular with zero block diagonal), obtaining

$$DF - FD = FN - NF$$

Considering the ij block of this equation, we obtain a well-known recurrence which gives a Sylvester equation for F_{ij} with a righthand side which only depends on values of F on higher superdiagonals (thus allowing a recursive algorithm for computing F).

$$T_{ii}F_{ij} - F_{ij}T_{jj} = F_{ii}T_{ij} - T_{ij}F_{jj} + \sum_{k=i+1}^{j-1} F_{ik}T_{kj} - T_{ik}F_{kj} \tag{A.6}$$

When T_{ii} and T_{jj} have distinct eigenvalues, this Sylvester equation uniquely determines F_{ij}. If there are common eigenvalues, then there are in general a family of solutions, but by a continuity argument we chose the one discussed above.

We proceed inductively. When $j = i + 1$, the sum on the right hand side is empty, and we have from definition (A.4) that $F_{i,i+1} = f[T_{ii}, T_{jj}]_{T_{ij}}$, so the proposition holds on the first superdiagonal. More generally, assume that the proposition (A.5) holds on the lth superdiagonal. Then, substituting the known blocks of F into (A.6), we obtain that (A.5) holds on the $l + 1$th super diagonal.

□

A.1.3 Relation to the Schur-Parlett Algorithm

The proof of the previous theorem (A.5) uses the same recursion as the Schur-Parlett algorithm [13], and as such recovers that algorithm in the case when the T_{ii} have distinct spectra, as they do when T is a block Schur decomposition. The theorem above remains true for arbitrary block upper triangular T, albeit with generalized divided differences that may not always be convenient to compute.

F_{ii}, unfortunately, seems to require $O(m_i^4)$ operations to stably compute in general, using a series method.

A.2 Derivatives of Matrix Functions of Block Upper Triangular Matrices

Armed with (A.5) and its attendant fast algorithm we can quickly produce expressions and fast algorithms for computing arbitrary derivatives of matrix functions. Indeed, consider a particular block upper triangular argument

$$B(A,V) \doteq \begin{bmatrix} A & V \\ 0 & A \end{bmatrix}$$

from which we obtain

$$f\left(B(A,V)\right) = \begin{bmatrix} f[A] & f[A,A]_V \\ 0 & f[A] \end{bmatrix} = \begin{bmatrix} f(A) & Df(A)\cdot V \\ 0 & f(A) \end{bmatrix},$$

a well-known identity that is useful for testing purposes, computing derivatives of matrix functions from a general algorithm for matrix functions, and which (as we shall see below) is also of great theoretical usefulness.

With our more general framework, this trivially extends to higher derivatives. Indeed, consider $B(B(A,V),W)$ and compute

$$\begin{aligned} f(B(A,V),W)) &= \begin{bmatrix} f(B(A,V) & f[B(A,V),B(A,V)]_W \\ 0 & f(B(A,V) \end{bmatrix} \\ &= \begin{bmatrix} \begin{bmatrix} f[A] & f[A,A]_V \\ 0 & f[A] \end{bmatrix} & \begin{bmatrix} f[A,A]_W & f[A,A,A]_{V,W} + f[A,A,A]_{W,V} \\ 0 & f[A,A]_W) \end{bmatrix} \\ 0 & \begin{bmatrix} f[A] & f[A,A]_V \\ 0 & f[A]) \end{bmatrix} \end{bmatrix} \\ &= \begin{bmatrix} \begin{bmatrix} f(A) & Df(A)\cdot V \\ 0 & f(A) \end{bmatrix} & \begin{bmatrix} Df(A)\cdot W & D^2f(A)\cdot(V,W) \\ 0 & Df(A)\cdot W) \end{bmatrix} \\ 0 & \begin{bmatrix} f(A) & Df(A)\cdot V \\ 0 & f(A)) \end{bmatrix} \end{bmatrix} \end{aligned}$$

This pattern continues recursively. We can extract $D^{(k)}f(A)\cdot(V_1,V_2,..,V_k)$ as the $(1,k)$ block of $f(B_k(A;V_1,...V_k))$ where $B_k(A;V_1,...V_{k-1},V_k) = B(B_{k-1}(A;V_1,...V_{k-1}),V_k)$ and $B_1(A;V) = B(A,V)$.

Since matrix functions are continuous functions of their arguments, and extraction of a block is a linear function, this construction immediately shows that what to this point have only been defined as Gâteaux (directional) derivatives are in fact Fréchet derivatives.

In the general case, one can use the Schur-Parlett algorithm on a full block matrix and extract the derivatives as sub blocks, which shows that a polynomial time algorithm exists for computing general derivatives of matrix functions, though there is a need to properly adapt the algorithm to take advantage of the highly structured form of B_k.

A.2.1 Derivatives of Diagonalizable Matrices

If $A = U\Sigma U^T$ is diagonalizable, then we can read off derivatives from (A.5), which simplifies greatly, as all blocks are 1×1 and all off-diagonal entries of T are 0. We can recover a well-known expression (Higham [19] Corollary 3.12), noting (A.1) and (1),

$$Df(A) \cdot V = U \left(Df(\Sigma) \cdot (U^T V U^T) \right) U^T$$

We have from (A.5) that $Df(\Sigma) \cdot \tilde{V} = f[\Sigma, \Sigma]_{\tilde{V}}$, which is the solution C to

$$\Sigma C + C\Sigma = f(\Sigma)\tilde{V} + \tilde{V} f(\Sigma)$$

which amounts to computing scalar divided differences of f, giving (4.3).

The second derivative is given, for $\mathbf{A}$ with diagonalization $\mathbf{A} = \mathbf{U}\Lambda\mathbf{U}^T$, as

$$D^2 f(\mathbf{A}) \cdot (\mathbf{V}, \mathbf{W}) = f[A, A, A]_{V,W} + f[A, A, A]_{W,V}$$

which can be written

$$D^2 f(\mathbf{A}) \cdot (\mathbf{V}, \mathbf{W}) = \mathbf{U}\Psi\mathbf{U}^T \tag{A.7}$$

where

$$\Psi_{ij} = \sum_k \Omega_{ijk} \left((\mathbf{U}^T\mathbf{V}\mathbf{U})_{ik} (\mathbf{U}^T\mathbf{W}\mathbf{U})_{kj} + (\mathbf{U}^T\mathbf{W}\mathbf{U})_{ik} (\mathbf{U}^T\mathbf{V}\mathbf{U})_{kj} \right)$$

and Ω is a symmetric three-dimensional array of second order divided differences, $\Omega_{ijk} = f[\lambda_i, \lambda_j, \lambda_k]$.

A.3 Computing Divided Differences

Computing a completely general divided difference is a daunting task. However, many special cases are worth addressing.

When the arguments to a divided difference have distinct spectra, standard methods for solving Sylvester equations may be employed.

Computing a matrix function itself (a 0th order divided difference) for a triangular matrix with equal or

near-equal eigenvalues is daunting. State-of-the-art methods [20] still require $O(n^4)$ operations in this case, using series methods.

In the case of diagonalizable matrices, divided differences can be computed, as in Section A.2.1, by computing arrays of divided differences of the scalar function f. This is useful in the setting of covariant hyperelasticity, but even this special case requires care, as we discuss in the following section.

A.3.1 Stabilizing Arbitrary Divided Differences of Scalar Analytic Functions

We present a general method of obtaining stable representations of divided differences of analytic functions $f : \mathbb{C} \to \mathbb{C}$. While this method is simple and seems quite powerful, we have not found it in the literature.

We begin with an elementary lemma, that divided differences of analytic functions are themselves multivariate analytic functions.

Lemma Let $f : \mathbb{C}^k \to \mathbb{C}$ be a multivariate analytic function, and $\tilde{f} \doteq f(x_1, \ldots, x_{k-1}, x_k)$. Let $g : \mathbb{C}^{k+1} \to \mathbb{C}$ be defined by $g(x_1, \ldots, x_{k-1}, y, z) = \tilde{f}[y, z]$ (the first divided difference of f with respect to its last argument). Then, g is a multivariate analytic function.

Proof Sketch For brevity, we omit the subscripts on $\tilde{f}$. Formally construct the power series of $\tilde{f}[y, z]$. When $y \neq z$, $\tilde{f}$ is analytic as the ratio of an analytic function and a nonzero analytic function. By formally dividing the series representation of $\tilde{f}(y) - \tilde{f}(z)$ by $y - z$, we obtain a convergent multivariate Taylor series which is equal to $\tilde{f}'(z)$ when $y = z$. $\square$

Corollary All divided differences of univariate analytic functions are themselves multivariate analytic functions.

Consider a multivariate analytic function $g : \mathbb{R}^d \to \mathbb{R}$ which, like our divided differences, is computationally tractable to compute for some arguments, which we will call *good*, but difficult for other, nearby *bad* arguments.

Given a good point $\mathbf{m} \in \mathbb{R}^d$ and a bad point $\mathbf{x}$ s.t. $|\mathbf{m} - \mathbf{x}|$ is sufficiently small, we can compute $g(\mathbf{x})$ to arbitrary precision with a Taylor series centered at $\mathbf{m}$.

If we restrict our attention to divided differences, note that a point is good if all pairs of arguments are equal or well-spaced, and bad if any two arguments are similar (relative to the magnitude of function the divided difference is based on) but unequal, due to floating-point cancellation errors.

In many cases of interest, by choosing m to respect the symmetry of the divided difference, the form of the resulting series is a known and easily computable form. For instance, when computing first divided differences of the exponential function, it is well-known that expressions involving the sinh function arise

when m is chosen to be the average of the arguments and we let $\mathbf{m} = (m, m)$. Similarly, expressions involving $\tanh^{-1}$ are known to arise when considering the first divided difference of the logarithm.

This technique can be generalized with the following procedure. Partition the arguments $x_i$of the divided difference into k clusters with the property that no two elements of distinct clusters are separated by a distance less than a chosen ϵ and such that each element in a non-singleton cluster is less than ϵ away from another element of the cluster. Choose a cluster center within each cluster and let m_i be the cluster center associated with the argument x_i, arranged into a vector $\mathbf{m}$. Using the analyticity of the divided difference function, write it in terms of the m_i and new variables $d_i = x_i - m_i$, in particular as a multivariate Taylor Series about m_i in each argument. Since the cancellation error is large precisely when the d_i are small but non-zero, the resulting series expression can be evaluated accurately with a small number of terms.

A.3.1.1 Example: Stabilized divided differences of the Logarithm

To provide a concrete example, we note the stabilized forms required for computation of the first two derivatives of the matrix logarithm of a diagonalizable matrix, in those cases where it cannot be computed stably from the definition, that is when two or more eigenvalues are poorly separated but unequal. The resulting series can be truncated to give any desired accuracy.

First Divided Difference For $|\lambda_i - \lambda_j| \ll 1$, let $m = (\lambda_i + \lambda_j)/2, d = (\lambda_i - \lambda_j)/2$, and we have the expression

$$\log[\lambda_i, \lambda_j] = (1/m)\frac{\tanh^{-1}(d/m)}{(d/m)} = (1/m)\sum_{j=0}^{\infty}\frac{(d/m)^{2j}}{2j+1}$$

Second Divided Difference, case 1 If $\lambda_i = \lambda_j$ and $|\lambda_i - \lambda_k| \ll 1$, we choose $\mathbf{m} = (\lambda_i, \lambda_i, \lambda_i)$ and let $d = \lambda_k - \lambda_i$ to obtain

$$\log^{[2]}[\lambda_i, \lambda_j, \lambda_j] = \frac{1}{\lambda_i^2}\sum_{k=0}^{\infty}\frac{(-1)^{k+1}}{k+2}\left(\frac{d}{\lambda_i}\right)^k$$

Second Divided Difference, case 2a If $|\lambda_i - \lambda_j| \ll 1$ but λ_k is well separated from λ_i and λ_j, by choosing $\mathbf{m} = (m, m, \lambda_k)$, where $m = (\lambda_i + \lambda_j)/2$ and letting $d = (\lambda_i - \lambda_j)/2$, we obtain a tractable power series, the first two terms of which we compute here as

$$\begin{aligned}\log^{[2]}[\lambda_i, \lambda_j, \lambda_k] = &\frac{(m-\lambda_k) + m\left(\log(\lambda_k) - \log(m)\right)}{(m-\lambda_k)^2 m} \\ &+ \frac{\left((m-\lambda_k)\right)\left(2\lambda_k^2 - 7\lambda_k m + 11m^2\right) + 6m^3(\log(\lambda_k) - \log(m))\big)}{6(m-\lambda_k)^4 m^3}d^2 \\ &+ O(d^3)\end{aligned}$$

Second Divided Difference, case 2b If all three eigenvalues $\lambda_i, \lambda_j, \lambda_k$ are clustered, by choosing $\mathbf{m} = (m, m, m)$ where $m = \frac{\lambda_i+\lambda_j+\lambda_k}{3}$, we obtain a tractable power series in terms of the components of $\mathbf{d} = (\lambda_i, \lambda_j, \lambda_k) - \mathbf{m} = (d_i, d_j, d_k)$. The leading terms are again computed for reference here.

$$\begin{aligned}\log^{[2]}[\lambda_i, \lambda_j, \lambda_k] = & -\frac{1}{2m^2} - \frac{1}{3m^3}(d_i + d_j + d_k) \\ & - \frac{1}{4m^4}(d_i^2 + d_i d_j + d_j^2 + d_j d_k + d_k^2 + d_k d_i) \\ & + O(|\mathbf{d}|^3)\end{aligned}$$

A.4 Series Methods

Though this exposition focuses on spectral formulæ for computing derivatives of matrix functions, an alternative that can be preferable is to differentiate an available series representation for f directly. In particular, this can be useful when the relevant series converges rapidly, or when working with a matrices with significant degenerate subspaces. Series methods also have a place within other methods, as in the block Schur-Parlett method for computing $f(T)$ [13], and are readily adapted to compute derivatives of Taylor series [47]. Methods based on Padé approximants are the state-of-the-art methods for computing first derivatives of matrix exponential and logarithm functions. These methods take advantage of identities specific to these functions, so are not directly generalizable.

Appendix B

Sample Code

We present some pseudo-C++ related to building Newton and Gauss-Newton systems for the methods presented in this thesis. The code is not intended to be complete or optimized, but rather to give a clearer idea of the practical expressions used for computation and to give a guide for those who would like to reimplement the methods.

B.1 Biot Strain Energy Newton System Assembly Pseudocode

The following pseudo-C++ should allow for easy implementation of the methods of Chapter 2. See also Appendix C. The code in this section is adapted from the publication by Chao, Pinkall, Sanan and Schröder [9].

B.1.1 Triangle Meshes

A full implementation computes quantities on a per triangle basis and puts them into the data structures for the minimizer. This implies a loop over the triangle and some mapping from the local vertices into a global array. For clarity this is left out and we treat a triangle all by itself.

Listing B.1: Unadorned Triangle Data Structure

```
class Tri{
  // X dependent (i.e., constant) section
  Vec2 x[4]; // original configuration
  Vec2 AreaGrad[3]; // area grad wrt vertex i=0,1,2
  double we[3]; // edge weights of 2D Laplace
  double area; // area of original triangle
  Mat2x2 Outer[6]; // Precomputed outer products of reference edges

  // x section which needs to be updated as we go along
```

```
  Vec2 q[3]; // current deformed configuration
  Mat2x2 R; // optimal rotation for current q
  Mat2x2 Y; // symmetric factor for current q
  double w; // weight in second part of Hessian
};

Tri::Init(){
  // compute all quantities depending on p alone
  // compute: vol, AreaGrad, we, Outer, R=Id() and store
}

Tri::ComputeAreaGrad( int i ){
  // volume gradient wrt to X_i (i=0,1,2)
  int j = (i+1)%3, k = (i+2)%3;
  // units of m^2
  AreaGrad[i] = (X[k]-X[j]).rotatePiOver2()/2;
}

// used to fill we[] array
double
Tri::ComputeEdgeWeight( int i, int j ){
  // weight in Dirichlet energy for edge ij
  // assumes volume gradients are initialized
  // units of (m^2)^2/m^3 = m
  return -AreaGrad[i].dot(AreaGrad[j])/area;
}

Tri::ComputeOuterMatrices(){
  // for the second term in the Hessian; precompute fixed matrices
  // outer products of opposing edges
  for (int i=0; i < 3; ++i){
    Vec2 Oppi = X[i+2%3]-X[i+1%3];
    for(int j=0; j < 3; ++j){
     Vec2 Oppj = X[j+2%3]-X[j+1%3];
    Outer[3*i+j] = Oppi.Outer(Oppj);
  }
  }
}
```

Listing B.2: Machinery for the First Variations of Biot Strain Energy for Triangles

```
Vec2
Tri::LaplaceQ( int i ){
```

```
  // use precomputed cot*edgelength weights
  int j = (i+1)%3, k = (i+2)%3;
  // e() maps from two index pairs to one of 3 edges (e(i,j)=0,1,2)
  return
    we[e(i,j)]*(x[i]-x[j]) + we[e(i,k)]*(x[i]-x[k]) ;
}

Vec2
Tri::Grad( int i, double alpha, double beta ){
  // gradient of energy wrt to i
  // assumes R and Y are current wrt x
  const double a2 = alpha*alpha, b2 = beta*beta;
  return b2*LaplaceQ(i) - (a2-(a2-b2)*Y.Trace()/2)*R*VolGrad[i];
}
```

Listing B.3: Machinery for the Second Variations of Biot Strain Energy for Triangles

```
Mat2x2
Tri::HessianSecondPart( int i, int j ){
  // computes local stiffness contribution from varying x_i,x_j
  // assumes Outer[] has been initialized
  // assume R is current at time of call
  if( isNotApproximatelyZero(Y.Trace()){
     // member function T() returns transpose
    return R*Outer[3*i+j]*R.T()/(4*Y.Trace()*area);
  }else{
    return Mat2x2::Zero();
  }
}

Mat2x2
Tri::HessianFirstPart( int i, int j ){
  // Laplace part
  // we[] are the original Laplace edge weights indexed by edge
  if( i == j ){ // diagonal entry
    // negative sum of off-diagonal entries in that row
    // e(a,b) maps a,b=0,1,2 to edge number 0,1,2
    return Id(2)*(we[e(i,(i+1)\%3)]+we[e(i,(i+2)\%3)];
  }else{
    return -Id(2)*we[e(i,j)];
  }
}
```

```
Mat2x2
Tri::HessianSecondPartABSplit( int i, int j ){
  // when \alpha \neq \beta this part splits off and is linearly combined
  // assumes R is current at time of Call
  //
  // the outer product between AreaGrad's could be precomputed
  return (R*AreaGrad[i]).Outer(R*AreaGrad[j])/area;
}

Mat2x2
Tri::TotalHessian( int i, int j, double alpha, double beta ){
// For a Gauss-Newton solver, ignore HessianSecondPart
  const double a2 = alpha*alpha, b2 = beta*beta;
  if( alpha == beta ){
    return a2*(HessianFirstPart(i,j)-HessianSecondPart(i,j));
  }else{
    return
      b2*HessianFirstPart(i,j)-
      (a2-(a2-b2)*Y.Trace()/2)*HessianSecondPart(i,j)+
      (a2-b2)/3*HessianSecondPartABSplit(i,j);
  }
}
```

B.1.2 Tetrahedral Meshes

Again, for clarity we only present expressions for gradient and Hessian contributions from individual vertex degrees of freedom, and do not include iterations over all vertices and elements and assembly into global data structures.

Listing B.4: Unadorned Tetrahedron Data Structure

```
class Tet{
  // X dependent (i.e., constant) section
  Vec3 X[4]; // original configuration
  Vec3 VolGrad[4]; // vol grad wrt vertex i=0,1,2,3
  double we[6]; // edge weights of 3D Laplace
  double vol; // volume of original tet
  Mat3x3 Curl[4]; // Precomputed opposing edge outer product sums

  // x section which needs to be updated as we go along
  Vec3 x[4]; // current deformed configuration
  Mat3x3 R; // optimal rotation for current q
  Mat3x3 Y; // symmetric factor for current q
```

```
  Mat3x3 W; // weight matrix in second part of Hessian
};

Tet::Init(){
  // compute all quantities depending on p alone
  // compute: vol, VolGrad, we, Curl, R=Id() and store
}

Tet::ComputeCurlMatrices(){
  // for the second term in the Hessian; precompute fixed matrices
  const int i = 0, j = 1, k = 2, l = 3;
  // six edges
  const Vec3
    pij = p[j]-p[i], plk = p[k]-p[l], pik = p[k]-p[i],
    pjl = p[l]-p[j], pil = p[l]-p[i], pkj = p[j]-p[k];
  // outer products of opposing edges in the needed linear combinations
  // could be optimized further, but is only precomputation, so don't bother
  Curl[i] = pij.Outer(plk) + pik.Outer(pjl) + pil.Outer(pkj);
  Curl[j] = pij.Outer(plk) + pjl.Outer(pik) + pkj.Outer(pil);
  Curl[k] = plk.Outer(pij) + pik.Outer(pjl) + pkj.Outer(pil);
  Curl[l] = plk.Outer(pij) + pjl.Outer(pik) + pil.Outer(pkj);
}
```

Listing B.5: Machinery for the First Variations of Biot Strain Energy for Tetrahedra

```
Tet::ComputeVolGrad( int i ){
  // volume gradient wrt to X_i (i=0,1,2,3)
  int j = (i+1)%4, k = (i+2)%4, l = (i+3)%4;
  // units of m^2
  VolGrad[i] = (X[k]-X[l]).cross(X[j]-X[l])/6;
}

// used to fill we[] array
double
Tet::ComputeEdgeWeight( int i, int j ){
  // weight in Dirichlet energy for edge ij
  // assumes volume gradients are initialized
  // units of (m^2)^2/m^3 = m
  return -VolGrad[i].dot(VolGrad[j])/vol;
}

Vec3
Tet::LaplaceQ( int i ){
```

```
  // use precomputed cot*edgelength weights
  int j = (i+1)%4, k = (i+2)%4, l = (i+3)%4;
  // e() maps from two index pairs to one of 6 edges (e(i,j)=0,1,2,3,4,5)
  return
    we[e(i,j)]*(x[i]-x[j]) + we[e(i,k)]*(x[i]-x[k]) + we[e(i,l)]*(x[i]-x[l]);
}

Vec3
Tet::Grad( int i, double alpha, double beta ){
  // gradient of energy wrt to i
  // assumes R and Y are current wrt x
  const double a2 = alpha*alpha, b2 = beta*beta;
  return b2*LaplaceQ(i) - (a2-(a2-b2)*Y.Trace()/3)*R*VolGrad[i];
}
```

Listing B.6: Machinery for the Second Variations of Biot Strain Energy for Tetrahedra

```
Tet::ComputeW(){
  // assumes Y is current at time of call
  // this is a weighting factor which depends on x (through Y)
  W = (Y.Trace()*Id(3)-Y).Inv(); // Expects W to be set to zero if the inverse does not exist
}

Mat3x3
Tet::HessianSecondPart( int i, int j ){
  // computes local stiffness contribution from varying p_i,p_j
  // assumes Curl[] has been intialized
  // assume R and W are current at time of call
  Mat3x3 RMi = R*Curl[i], RMj = R*Curl[j];
  // member function T() returns transpose
  return RMi*W*RMj.T()/(36*vol);
}

Mat3x3
Tet::HessianFirstPart( int i, int j ){
  // Laplace part
  // we[] are the original Laplace edge weights indexed by edge
  if( i == j ){ // diagonal entry
    // negative sum of off-diagonal entries in that row
    // e(a,b) maps a,b=0,1,2,3 to edge number 0,1,2,3,4,5
    return Id(3)*(we[e(i,(i+1)%4)]+we[e(i,(i+2)%4)]+we[e(i,(i+3)%4)]);
```

```
  }else{
    return -Id(3)*we[e(i,j)];
  }
}

Mat3x3
Tet::HessianSecondPartABSplit( int i, int j ){
  // when \alpha \neq \beta this part splits off and is linearly combined
  // assumes R is current at time of Call
  //
  // the outer product between VolGrad's could be precomputed, but we
  // would have to store 10 of them (symmetric part of 4x4 matrix) and
  // don't want to burn the memory
  return (R*VolGrad[i]).Outer(R*VolGrad[j])/vol;
}

Mat3x3
Tet::TotalHessian( int i, int j, double alpha, double beta ){
// For a Gauss-Newton solver, ignore HessianSecondPart
  const double a2 = alpha*alpha, b2 = beta*beta;
  if( alpha == beta ){
    return a2*(HessianFirstPart(i,j)-HessianSecondPart(i,j));
  }else{
    return
      b2*HessianFirstPart(i,j)-
      (a2-(a2-b2)*Y.Trace()/3)*HessianSecondPart(i,j)+
      (a2-b2)/3*HessianSecondPartABSplit(i,j);
  }
}
```

B.2 Bounded-Distortion Parameterization Augmented Lagrangian System Assembly Pseudocode

We provide additional terms which, when combined with the expressions in Section B.1, allow construction of the Augmented Lagrangian system described in Chapter 3.

Listing B.7: Machinery for the First and Second Variations of Augmented Terms

```
class AugTri : public Tri
{
  double lambda;   // Estimated Lagrange Multiplier
```

```
  double mu;     // Penalty Parameter
  double Ctilde;   // in [0,1], controlling allowable distortion
  double cAdj;     // Shifted constraint function
  double normYbar; // ||Y-Y.trace()/d||_fro
};

double
AugTri::ComputeCAdj
{
  // Assumes Y has been computed
  return fmax(0, Ctilde * (Y.Trace()/sqrt(2)) - normYbar + lambda/mu);
}

Vec2
AugTri::AugGradFirstTerm(int i)
{
  // Assumes R and Y have already been computed
  return (1/sqrt(2)) * R * areaGrad[i];
}

Vec2
AugTri::AugGradSecondTerm(int i)
{
  // Assumes that R and normYbar has been computed
  return (LaplaceQ(i) - (Y.Trace()/2)*R*areaGrad[i])/normYbar;
}

Vec2
AugTri::GradC(int i)
{
  return Ctilde * AugGradFirstTerm(i) - AugGradSecondTerm(i);
}

Vec2
AugTri::AugGrad(int i)
{
  // Assumes that cAdj has already been computed
  return mu * cAdj * gradC(i);
}

Mat2x2
AugTri::AugGaussNewtonTerms(int i, int j)
{
```

```
  if (cAdj == 0) return Mat2x2::Zero();
  return (GradC(i)).Outer(GradC(j))/area;
}

Mat2x2
AugTri::cHess(int i, int j)
{
  // Outer products of areaGrads can be precomputed if desired
  return
    Ctilde * (1/sqrt(2)) * (1/(4*area*trace(Y)))*R*outer[3*i+j]*R.T() +
    -(
      AugGradSecondTerm(i).Outer(AugGradSecondTerm(j))
      AugGradFirstTerm(i).outer(AugGradFirstTerm(j))
      + (1/8) * R * outer[3*i+j]*R.T();
    );
}

Mat2x2
AugTri::AugHess(int i, int j)
{
  // For a Gauss-Newton solver, ignore the cHess contribution
  // This expression amounts to a combination of LaplaceQ blocks,
  //  outer blocks, and areaGrad outer product blocks, and
  //   can be further simplified by noting that structure
  return mu * ( AugGaussNewtonTerms(i,j)  + cAdj * cHess(i,j) )
}
```

B.3 Generalized Quadratic Strain Energies Pseudocode

Here, we present expressions useful for constructing a Newton or Gauss-Newton system to minimize the strain energies discussed in Chapter 4. Here, we template on the dimension of the simplex `D` (2 for triangles, 3 for tetrahedra).

Listing B.8: Unadorned Element Data Structure for Generalized Strain Energies

```
template<int D>
class Element
{
  Vec<D> VolGrad[D+1];
  double vol;
  Mat<D> F;
  Mat<D> C;
```

```
  Mat<D> U;
  DiagonalMat<D> Lambda;
  double a, b;

  // Scalar strain function and its first two derivatives
  double (*f)  (double);
  double (*fp) (double);
  double (*fpp)(double);

  // Various matrix functions of C
  Mat<D> fC, fpC, pC;

  // Arrays of Divided Differences
  Mat<D> Thetap, Thetafp;
}
```

Listing B.9: Machinery for the First and Second Variations of Generalized Strain Energies

```
template<int D>
Vec<D>
Element<D>::Gradient(int i)
{
  return (2/vol)*F*((a^2-b^2)*(fC.Trace()/D)*fpC + b^2*pC)*areaGrad[i];
}

template<int D>
Mat<D>
Element<D>::HessianFirstTerm(int i, int j, Mat<D> gC)
{
  return Mat<D>::Identity() * volGrad[i].T()*gC*volGrad[j];
}

template<int D>
Mat<D>
Element<D>::HessianSecondTerm(int i, int j, Mat<D> Thetag)
{
  // Outer products of volGrad terms can be precomputed if desired
  return (F*U)*(
          Diag(Thetag * (U.T() * volGrad[i].elementWiseProduct(U.T() * volGrad[j]))) +
            Thetag.elementWiseProduct(U.T() * (volGrad[j].Outer(volGrad[i])) * U)
        )*(F*U).T();
}
```

```
template<int D>
Mat<D>
Element<D>::Hessian(int i, int j)
{
  // Omit HessianSecondTerm to obtain Gauss-Newton solver
  // Outer products of volGrad terms can be precomputed if desired
  if(a==b){
    return (2.0/vol) * b^2 * ( HessFirstTerm(i,j,pC) + HessSecondTerm(i,j,pC,Thetap));
  }else{
    const double a2b2 = (a^2-b^2);
    return (2.0/vol) * (
         b^2  *                (HessianFirstTerm(i,j,pC)  + HessSecondianTerm(i,j,Thetap)) +
         a2b2 * fC.Trace()/D * (HessianFirstTerm(i,j,fpC) + HessSecondianTerm(i,j,Thetafp))+
         a2b2 * F * fpC *volGrad[i]).Outer(F * fpC(volGrad[j])
         );
  }
}
```

Appendix C

Matrix Facts

The results presented in this Appendix are based on the publication *A Simple Geometric Model for Elastic Deformations* by Chao, Pinkall, Sanan and Schröder [9].

We collect some identities useful for deriving the expressions for the second discrete variation of the elastic energy in Chapter 2.

Inner products of 2D and 3D matrices are given by $\langle \mathbf{A}, \mathbf{B} \rangle = \mathrm{tr}(\mathbf{A}^T\mathbf{B})$ with the L_2-norm $|\mathbf{A}|^2 = \langle \mathbf{A}, \mathbf{A} \rangle$ used to measure distances between matrices $\mathrm{dist}(\mathbf{A}, \mathbf{B}) = |\mathbf{A} - \mathbf{B}|$. All matrices with positive determinant possess a unique (right) polar decomposition $\mathbf{A} = \mathbf{RY}$ into a rotation $\mathbf{R}$ and a symmetric $\mathbf{Y} = (\mathbf{A}^T\mathbf{A})^{1/2}$. This R is closest to $\mathbf{A}$ in the L_2-norm. This follows from $|\mathbf{RY} - \mathbf{R}|^2 = |\mathbf{Y} - I|^2$ and the L_2 orthogonality of symmetric and anti-symmetric matrices, the latter forming the tangent space to $\mathrm{SO}(n)$ at the identity.

Inner products involving anti-symmetric matrices can be greatly simplified. Consider $\langle \mathbf{B}, \mathbf{A}_X \rangle$ for arbitrary $\mathbf{B}$ and anti-symmetric $\mathbf{A}_X$. The result depends only on the anti-symmetric part of $\mathbf{B}$, $\mathbf{B}_X = \frac{1}{2}(\mathbf{B} - \mathbf{B}^T)$ due to the orthogonality of symmetric and anti-symmetric matrices. In the 2D case, anti-symmetric matrices $\mathbf{A}_X$ are multiples of J, the rotation by $\pi/2$, $\mathbf{A}_X = aJ$. We will denote the extraction of the representative a as $\mathrm{X}(\mathbf{A}_X) = a$ and more generally use $\mathrm{X}(\mathbf{B}) = b$ for extracting the representative of the anti-symmetric part of an arbitrary $\mathbf{B}$. Hence $\langle \mathbf{B}, \mathbf{A}_X \rangle = 2\,\mathrm{X}(\mathbf{B})\,\mathrm{X}(\mathbf{A}_X)$. In 3D anti-symmetric matrices can be written as cross products with suitable vectors a, $\mathbf{A}_X = a\times$. Again we use the notation $\mathrm{X}(v_X) = a$ for "pulling out" the representative, this time a 3-vector. Consequently $\langle \mathbf{B}, v_X \rangle = 2\langle \mathrm{X}(\mathbf{B}), \mathrm{X}(v_X) \rangle$. We summarize this by writing $\langle \mathbf{B}, v_X \rangle = 2\langle \mathrm{X}(\mathbf{B}), \mathrm{X}(v_X) \rangle$ in both the 2D and 3D cases.

The representative of the anti-symmetric part of a map could be computed by first finding the corresponding matrix representation and then "extracting" the representative as above. Alternatively, and more simply, it can be computed directly from the X (domain) and x (range) data as the discrete curl of x.

For a given triangle X_{ijk} we get

$$c_{ijk} = \langle \tfrac{1}{2}(x_i + x_j), X_{ij} \rangle + \langle \tfrac{1}{2}(x_j + x_k), X_{jk} \rangle + \langle \tfrac{1}{2}(x_k + x_i), X_{ki} \rangle$$

as the boundary integral (using Stokes' theorem) of the corresponding piecewise linear interpolant. For maps over 2D triangles we then get

$$\mathrm{X}(v) = \tfrac{c_{ijk}}{2a_{ijk}},$$

i.e., the discrete curl is normalized by the area a_{ijk} of the triangle X_{ijk} and $\mathrm{X}(v)$ becomes its density. For maps over tetrahedra the 3-vector representative of the anti-symmetric part of the map is also given as a density, this time normalized by the tetrahedron volume v_{ijkl}

$$\mathrm{X}(v) = \tfrac{1}{6v_{ijkl}}(c_{kli}X_{ij} + c_{lji}X_{ik} + c_{ijk}X_{il}).$$

This expression uses the fact that discrete curls are divergence free, *i.e.*, they must sum to zero over the boundary of a tetrahedron. Hence four discrete curl values on the faces of a tetrahedron together with the divergence free constraint uniquely fix the discrete curl 3-vector $\mathrm{X}(v)$ associated to the tetrahedron as a whole.