Diss. ETH No. 23223

Large scale simulation of cloth and hair with contact

A dissertation submitted to
ETH Zurich

for the Degree of
Doctor of Sciences

presented by
Rasmus Tamstorf
M.Sc. Electrical Engineering, Technical University of Denmark
born June 19, 1972
citizen of Denmark

accepted on the recommendation of
Prof. Dr. Markus Gross, examiner
Prof. Dr. Peter Schröder, co-examiner
Prof. Dr. Eitan Grinspun, co-examiner

2016

Abstract

The goal of this thesis is to develop new methods for robust and scalable simulation of thin objects with contact. This work is motivated by applications in cloth and hair simulation in feature film production, but the results are also applicable in other domains such as in the garment industry, in biology, and in mechanical engineering.

The contributions focus primarily on physically correct contact response and efficient implementation thereof. While collision detection has been studied extensively, collision response remains a challenging problem. This is due in part to the fact that contact constraints are both nonlinear, non-convex, and non-smooth. Furthermore, the response is inherently coupled to the underlying dynamics, which is often neglected in existing methods.

We first develop the necessary details for implementation of the dynamics of orthotropic thin shells undergoing large deformations. This derivation is based on fundamental invariants and symmetry considerations from continuum mechanics and leads to a simple and efficient extension of existing membrane models. We also show how the analytical derivatives for the discrete shell bending model can be computed efficiently to facilitate implicit integration.

The next part of the thesis is based on the observation that nonlinear compliance is critical for collision response involving thin objects. This is illustrated with rods (hair), which exhibit the same fundamental problem as shells, but are cheaper to simulate. Taking the nonlinearity into account, we show that simulations can be run with time steps that are $2-3$ orders of magnitude larger than with existing methods. Based on this observation, we construct a solver that locally adapts the nonlinear treatment based on the current configuration to ensure stable simulations. We then validate our approach by running simulations with thousands of hairs and millions of contacts.

Hair simulation is generally simpler than cloth because the rods can be partitioned into contact groups that can each be handled separately. For shells, all degrees of freedom are inherently coupled, which leads to much larger systems of equations that have to be solved. In pursuit of solving these larger systems of equations, the last part of the thesis investigates the use of algebraic multigrid methods. The first key observation here is that standard multigrid methods (both geometric and

algebraic) fail to perform well for thin shell equations even in the absence of contact. To address this difficulty, we introduce the smoothed aggregation method to cloth simulation. This is an algebraic multigrid method designed specifically for vector valued problems. Used as a preconditioner for CG, smoothed aggregation provides substantial speedups for medium and large size problems compared to a diagonally preconditioned CG method. Compared to geometric multigrid methods, it provides both better convergence rates and increased flexibility. Due to the algebraic nature of the method, it can be used for irregular meshes as well as with adaptive tessellations, which is not practical with geometric multigrid methods.

Zusammenfassung

Das Ziel dieser Dissertation ist die Entwicklung robuster und skalierbarer Methoden zur Simulation von dünnen Objekten mit Kontakten wie beispielsweise Haare und Kleider. Motiviert wurde die Arbeit durch die Filmindustrie, die Resultate können aber zum Beispiel auch in der Biologie oder im Maschinenbau angewendet werden.

Der primäre Beitrag dieser Dissertation ist die physikalisch korrekte Kollisionsbehandlung und deren effiziente Implementierung. Während das Problem der Kollisionserkennung verbreitet erforscht wurde, ist die korrekte Kollisionsbehandlung ein schwieriges Problem, weil die Kontaktbedingungen nicht-linear, nicht-konvex und nicht glatt sind. Ausserdem vernachlässigen existierende Methoden oft die Abhängigkeit der Kollisionsantwort von den unterliegenden Dynamiken.

Wir leiten zuerst die nötigen Gleichungen zur Implementierung der Dynamiken von orthotropen dünne Schalen, die deformiert werden, her. Diese Herleitung basiert auf Invarianz- und Symmetrieannahmen aus der Kontinuumsmechanik und liefert eine einfache und effiziente Erweiterung von existierenden Membranmodellen. Wir zeigen auch, wie die analytischen Ableitungen für diskrete Schalenmodelle effizient berechnet werden können.

Der nächste Teil der Arbeit basiert auf der Beobachtung, dass die nicht-lineare Übereinstimmung für die Kollisionsantwort essentiell ist. Wir illustrieren diese Beobachtung mit Kurvenmodellen, welche die gleichen fundamentalen Probleme wie Schalenmodelle zeigen, jedoch einfacher zu simulieren sind. Unter Beachtung der Nichtlinearität zeigen wir, dass unsere Simulationen mit 100-1000 Mal grösseren Zeitschritten funktionieren als existierende Methoden. Wir konstruieren einen Solver, der die Behandlung der Nichtlinearität lokal anpasst, um die Stabilität der Simulation zu garantieren. Wir validieren unsere Methode mit Simulationen von tausenden von Haaren und Millionen von Kontakten.

Die Simulation von Haar ist im Allgemeinen einfacher als die Simulation von Kleidern, weil Kurvenmodelle in Kontaktgruppen aufgeteilt und separat behandelt werden können. Bei Flächenmodellen sind alle Freiheitsgrade voneinander abhängig, was zu viel grösseren Gleichungssystemen führt. Der letzte Teil dieser Arbeit befasst sich mit algebraischen Mehrgitterverfahren zur Lösung solcher Systeme. Existierende algebraische und geometrische Mehrgitterverfahren

funktionieren für Flächenmodelle jedoch schlecht. Diese Arbeit führt deshalb eine glättende Aggregationsmethode zur Kleidersimulation ein. Es handelt sich dabei um ein Mehrgitterverfahren, welches speziell für vektorwertige Probleme konzipiert ist. Das Verfahren konvergiert schneller als geometrische Mehrgitterverfahren und bietet eine höhere Flexibilität. Da es sich um eine algebraische Methode handelt, kann diese auch für irreguläre Netze und adaptive Triangulierungen verwendet werden.

Acknowledgments

First of all, I'd like to thank Joe Marks for suggesting that I pursue a PhD and for making it possible for me to do it as part of my work at Walt Disney Animation Studios. Thanks also to my advisor, Markus Gross, for taking on an external student despite all the challenges it brings. Throughout the process, Eitan Grinspun has been a great collaborator, always a source of good advice, and a wonderful host for my sometimes extended stays at Columbia University. Along with Eitan, Peter Schröder has kindly agreed to be on my committee, which is greatly appreciated.

Much of what I know today about multigrid methods is entirely thanks to Stephen McCormick and Toby Jones. Steve deserves much gratitude for including me in his extended multigrid family and so willingly sharing his wisdom. In a serendipitous sequence of events, this project has gotten Toby his dream job, but in the process I have also gotten a great colleague and someone who has been willing to check my math, find my bugs, and discuss ideas that have occasionally been less than half baked. Many others have helped including my other co-authors, Danny Kaufman, Breannan Smith, and Jean-Marie Aubry along with the numerous artists at Disney who have lent their talents towards the creation of some of the figures in this thesis. Finally, I'd like to thank Alicia Jones for all the encouragement along the way.

Contents

List of Figures

List of Algorithms

List of Tables

List of Symbols

Generic symbols

$\boldsymbol{A}$	Generic 2nd order tensor
$\boldsymbol{B}$	Generic 2nd order tensor
$\boldsymbol{I}$	Identity matrix/tensor (rank 2)
$\mathbb{I}$	Identity tensor (rank 4)
$\mathcal{O}$	Big O notation

Positions, velocities, and configurations

$\boldsymbol{x}$	Position of a deformable point or vertex
$\bar{\boldsymbol{x}}$	Position of point in reference configuration
$\mathbf{q}$	Generalized coordinates of system
$\mathbf{p}$	Generalized momentum of system
$\mathbf{q}^f$	Configuration of free (or deformable) vertices
$\mathbf{q}^s$	Configuration of scripted (or kinematic) vertices
$\boldsymbol{p}^a$	Position of colliding point on object A
$\boldsymbol{p}^b$	Position of colliding point on object B
γ^a	Barycentric coordinates for colliding point on object A
γ^b	Barycentric coordinates for colliding point on object B
γ	Generalized barycentric coordinates for collision point
$\boldsymbol{v}$	Velocity
$\boldsymbol{u}$	Relative (contact) velocity

Continuum mechanics

ψ	Energy density
ϕ	Deformation function
$\boldsymbol{F}$	Deformation gradient
$\boldsymbol{P}$	First Piola-Kirchhoff stress tensor
$\boldsymbol{S}$	Second Piola-Kirchhoff stress tensor
$\boldsymbol{\sigma}$	Cauchy stress tensor

$\boldsymbol{b}$	Body force density
$\boldsymbol{E}$	Green-Lagrange strain tensor
$\boldsymbol{\epsilon}$	Engineering (linear) strain tensor
$\boldsymbol{C}$	Right Cauchy-Green deformation tensor
$\boldsymbol{R}$	Rotation tensor
$\boldsymbol{U}$	Stretch tensor
$\mathbb{C}$	Elasticity tensor
$\hat{\boldsymbol{a}}$	Direction of orthotropy
$\boldsymbol{A}$	Structural tensor for orthotropy
$\boldsymbol{Q}$	Orthogonal tensor
$\mathcal{G}$	Symmetry group
J	Main invariants
I	Principal invariants
L	Natural orthotropic invariants

Energy related quantities

W_{inc}	Incremental potential
W_{kin}	Kinetic energy
W_{pot}	Potential energy
$\boldsymbol{H}_{pot}$	Hessian of potential energy
$\boldsymbol{H}$	Hessian
$W_{elastic}$	Elastic energy
W_{bend}	Bend energy
$W_{stretch}$	Stretch energy (1D)
W_{mem}	Membrane energy (2D)
$W_{gravity}$	Gravitational potential energy

Material parameters

Y	Young's modulus
ν	Poisson's ratio
G	Shear modulus
D	Flexural rigidity
λ	First Lamé parameter
μ	Second Lamé parameter (shear modulus)
α	First orthotropic material parameter
β	Second orthotropic material parameter
ρ	Mass density

$\boldsymbol{M}$	Mass matrix
μ	Friction coefficient
h	Shell thickness

Geometric quantities

$\boldsymbol{e}$	Edge vector
$\boldsymbol{n}$	Face normal
$\boldsymbol{m}$	In-plane edge normal
l	Deformed edge length
$\bar{l}$	Undeformed edge length
s	Edge based strain
$\boldsymbol{s}$	Edge based strain vector
θ	Bend angle
α	Interior angle in triangle
h	Height of triangle
A	Area of triangle
ω	Scalar weight (not a mass)
ε	Minus the Levi-Civita permutation tensor

Contacts and constraints

C	Scalar constraint function
$\boldsymbol{C}$	Vector valued constraint function
$\mathcal{C}$	Set of contacts or collisions
$\mathcal{S}$	Collision stencil
$\hat{\boldsymbol{n}}$	Normalized contact normal
$\hat{\boldsymbol{d}}$	Normalized contact tangent vector
Φ	Orthonormal contact basis
$\mathbf{n}$	Generalized contact normal
$\boldsymbol{N}$	Matrix of generalized contact normals
$\boldsymbol{T}$	Matrix of generalized contact tangents
$\boldsymbol{G}$	Matrix of generalized contact basis
$\boldsymbol{E}$	Selection matrix for linearized friction cone
d	Separation distance
$\boldsymbol{g}$	Gap vector
s	Constraint offset due to scripted objects
$\mathcal{R}$	Set of valid collision responses

Forces

$\boldsymbol{f}$	Force
$\boldsymbol{r}$	Collision response force
$\boldsymbol{r}_n$	Contact force
$\boldsymbol{r}_t$	Friction force
$\mathbf{r}$	Generalized collision response force
$\mathbf{r}_n$	Generalized contact force
$\mathbf{r}_t$	Generalized friction force

Optimization

$\mathcal{L}$	Lagrangian
α	Lagrange multiplier for contact force
β	Lagrange multiplier for friction force
λ	Lagrange multiplier for all constraints
ϕ	Quadratic objective function
χ	Indicator function

Dynamics

t	Time
τ	Time step size
$\mathbf{y}$	State vector
φ	The flow function for a dynamic system
f	Nonlinear residual function
$\boldsymbol{K}$	Generalized stiffness matrix
$\boldsymbol{D}$	Discrete compliance matrix
δ	Configuration update

Throughout the text, the following conventions are used :

- Scalars are denoted by lower case regular letters.
- Vectors are denoted by lower case bold letters.
- Matrices are denoted by upper case bold letters.
- Second-order tensors are denoted by upper case sans serif bold letters.
- Fourth-order tensors are denoted by upper case letters in blackboard font.

- Generalized vectors are denoted by lower case sans serif bold letters.
- Generalized matrices are denoted by upper case sans serif bold letters.
- Sets are denoted by caligraphic letters.

A hat over a vector denotes a normalized vector. A bar over a quantity indicates that the quantity refers to the undeformed configuration. A dot over a quantity indicates differentiation with respect to time.

The gradient of a scalar multivariate function $f : \mathbb{R}^n \to \mathbb{R}$ is denoted by $\nabla f \in \mathbb{R}^{1\times n}$ and is considered to be a row vector. The Jacobian of a vector valued multivariate function $\mathsf{f} : \mathbb{R}^n \to \mathbb{R}^m$ is also denoted by $\nabla \mathsf{f} \in \mathbb{R}^{m\times n}$, but is now a matrix.

When specifying tensors in coordinate form, we will use a Cartesian basis, which means that the covariant and contravariant form of the tensor have the same components.

CHAPTER 1

Introduction

Cloth and hair simulations are key elements in animated feature films. Not only do they contribute significant aspects of the overall design and appeal of the characters; sometimes they even make up an integral part of the story. One such example is Rapunzel's hair in Disney's animated feature film Tangled, and despite having a less prominent role in the story, Elsa's dress in Disney's Frozen has been a favorite with little girls since the movie came out. By increasing the fidelity and reliability of these simulations, filmmakers can expand the range of costumes and outfits that can be used successfully as part of their story telling.

Accurate cloth simulation is also a topic that is being explored extensively for the fashion and retail industry. If successful, this technology is expected to increase online retail of real garments as well as being able to provide customized tailoring over the internet. Worldwide, more than a trillion dollars is spent annually on apparel, but, as of 2015, it's still only a small percentage of those purchases that are being made online. Thus, there is plenty of room to grow and given the size of the overall market, even a small improvement in the online shopping experience is likely to provide significant business opportunities.

In engineering, cloth simulation is used for testing of body armor, for construction of fiber reinforced composites, and for simulation of airbags and parachutes to name just a few. Thin-walled structures with large deformations also show up in other domains like medicine and biology where they can be used to simulate arteries, various cell structures, and leaf-like fea-

tures. Beyond these domains, classical applications include simulation of airplane fuselages, architectural structures, and car bodies. Accurate simulation of contact is essential here in order to do crash simulations to evaluate safety features.

1.1 Outline

We begin this thesis by reviewing some related work in Chapter 2. Given the immense amount of related work, this chapter is by no means intended to be exhaustive, but rather just a sampling of some of the prior work that has inspired this thesis.

In Chapter 3, a number of basic concepts from continuum mechanics are introduced. The goal is to establish the fundamental principles that govern the construction of hyperelastic constitutive models. Based on these principles, we construct an orthotropic St. Venant-Kirchhoff membrane model. Using a discretization based on constant strain triangles (CST), we furthermore present an efficient way to compute both the associated nodal forces and their derivatives necessary for implicit time integration (Chapter 4).

A thin shell model consists of both a membrane part and a bending part. For bending we use the discrete shells model by Grinspun et al. [2003]. The derivation of the energy Hessian for this model is quite tedious and error prone, but as well shall show, a careful hand derivation can leverage several symmetries in the expressions and achieve more than $3\times$ speedup over a derivation that doesn't leverage these symmetries, and more than $7\times$ speedup over an implementation using automatic differentiation (Chapter 5).

Given the established shell model, Chapter 6 focuses on the unconstrained dynamics for thin shells. We illustrate with the simplest possible example why it is advantageous to phrase time integration as an optimization problem by way of an *incremental potential* as originally suggested by Radovitzky and Ortiz [1999] and later Kharevych et al. [2006]. However, we also show that the resulting non-convex optimization problem can be difficult to solve due to ill-conditioning; especially in finite precision arithmetic.

In Chapter 7, we add an essential component for this thesis : contact handling. In particular, we focus on how to formulate non-penetration and friction constraints for the previously established optimization problem. The formulation provided here is generic as it applies to both hair and cloth simulation and it effectively couples the contact handling with the update of the dynamics equations.

The generic formulation can be expensive to solve. In Chapter 8, we make the key observation that thin objects like hair and cloth in some cases have no resistance (up to third order) to impacts perpendicular to their extent. While *linear* compliance as introduced by Otaduy et al. [2009] does couple dynamics and contact handling, the first-order model may therefore be insufficient. In particular, we show that high speed impact with a linear compliance model can lead to excessive deformation, which in turn often leads to instability. By accommodating an adaptively nonlinear compliance model, we show stable hair simulations with time steps that are $2-3$ orders of magnitude larger than otherwise possible.

Extending these results from hair to cloth is nontrivial because the shell equations are inherently much more tightly coupled than the rod equations. This means that we have to solve a single large coupled system instead of multiple smaller problems. In order to solve large problems, it is often beneficial to use multigrid methods, which is the focus of the last part of the thesis. Ultimately, we need to solve large non-convex and non-smooth optimization problems with inequality constraints. However, they can be recast into a sequence of convex quadratic programs (QPs) with inequality constraints. Using an active set strategy the QPs in turn can be solved as a sequence of quadratic problems equality constraints. This latter class of problems will be our focus in the last part of the thesis.

In Chapter 10, we present the basic principles of multigrid theory and, in Chapter 11, we apply this theory to cloth simulation. The first key observation here is that the standard multigrid methods (both geometric and algebraic) fail to perform well for thin shell equations even in the absence of contact. To address this difficulty we introduce the smoothed aggregation method to cloth simulation. This is an algebraic multigrid method designed specifically for vector valued problems. Used as a preconditioner for CG, smoothed aggregation provides substantial speedups for medium and large size problems compared to a diagonally preconditioned CG method. Compared to geometric multigrid methods, it provides both better convergence rates and increased flexibility. Due to the algebraic nature of the method, it can be used for irregular meshes as well as adaptive tessellations, which is not practical with geometric multigrid methods.

The extension to non-convex problems and/or inequality constraints is left as future work.

1.2 Contributions

- A practical method for implementing a geometrically nonlinear St. Venant-Kirchhoff material model for orthotropic materials with large displacements and large strains using a constant strain triangle discretization.
- An efficient way of computing analytical derivatives for bending models involving the dihedral angle between two triangles.
- An adaptive method for handling the nonlinearity inherent to collision response for thin objects.
- A prefiltering method for handling equality constraints in a block-sparse linear system without changing block size.
- An algebraic multigrid method for solving large linear systems of equations from cloth simulations including basic collision response.

1.3 Publications

This thesis is based on the following peer-reviewed publications :

Rasmus Tamstorf and Eitan Grinspun. **Discrete bending forces and their Jacobians**. *Graphical Models*, 75(6):362–370, November 2013. doi:10.1016/j.gmod.2013.07.001

Danny Kaufman, Rasmus Tamstorf, Breannan Smith, Jean-Marie Aubry, and Eitan Grinspun. **Adaptive Nonlinearity for Collisions in Complex Rod Assemblies**. *ACM Trans. Graph*, 33(4), July 2014. doi:10.1145/2601097.2601100

Rasmus Tamstorf, Toby Jones, and Stephen F. McCormick. **Smoothed Aggregation Multigrid for Cloth Simulation**. *ACM Trans. Graph.*, 34(6), November 2015. ISSN 0730-0301. doi:10.1145/2816795.2818081

The first of these papers was selected for ACM's Computing Reviews' Best of 2013 list[1].

The material in chapters 2 and 3 is a summary of known material by other authors, while the material in Chapter 4 is unpublished. Chapter 5 follows the first paper closely, [Tamstorf and Grinspun, 2013]. Chapter 6 is mostly known material but with motivating examples showing the applicability for

[1] http://computingreviews.com/recommend/bestof/notableitems_2013.cfm

this thesis. Also, it adds a critical observation that ill-conditioning in the systems that have to be solved is unavoidable when taking large time steps and is independent of any problems stemming from the thickness of cloth or the choice of discretization. The presentation in Chapter 7 extends the material in the second paper, [Kaufman et al., 2014], to clarify the presentation and to make the connection to [Kaufman et al., 2008] more clear. The remainder of the second paper, [Kaufman et al., 2014], is covered in Chapter 8. The remaining chapters follow the third paper closely, [Tamstorf et al., 2015].

During the course of the PhD program, the following peer-reviewed papers were also published :

David Harmon, Etienne Vouga, Breannan Smith, Rasmus Tamstorf, and Eitan Grinspun. **Asynchronous contact mechanics**. In *ACM SIGGRAPH 2009 papers*, SIGGRAPH '09, pages 87:1–87:12, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-726-4. doi:10.1145/1576246.1531393

Iman Sadeghi, Heather Pritchett, Henrik Wann Jensen, and Rasmus Tamstorf. **An Artist Friendly Hair Shading System**. *ACM Trans. Graph.*, 29 (3):56:1–10, 2010. doi:10.1145/1833349.1778793

Etienne Vouga, David Harmon, Rasmus Tamstorf, and Eitan Grinspun. **Asynchronous variational contact mechanics**. *Computer Methods in Applied Mechanics and Engineering*, 200(25–28):2181–2194, 2011. doi:10.1016/j.cma.2011.03.010

Aleka McAdams, Yongning Zhu, Andrew Selle, Mark Empey, Rasmus Tamstorf, Joseph Teran, and Eftychios Sifakis. **Efficient Elasticity for Character Skinning with Contact and Collisions**. *ACM Trans. Graph.*, 30(4): 37:1–37:12, July 2011. ISSN 0730-0301. doi:10.1145/2010324.1964932

Jeroen van Baar, Steven Poulakos, Wojciech Jarosz, Derek Nowrouzezahrai, Rasmus Tamstorf, and Markus Gross. **Perceptually-Based Compensation of Light Pollution in Display Systems**. In *Symposium on Applied Perception in Graphics and Visualization*, 2011

David Harmon, Etienne Vouga, Breannan Smith, Rasmus Tamstorf, and Eitan Grinspun. **Asynchronous contact mechanics**. *Commun. ACM*, 55(4): 102–109, April 2012. ISSN 0001-0782. doi:10.1145/2133806.2133828

Breannan Smith, Danny M. Kaufman, Etienne Vouga, Rasmus Tamstorf, and Eitan Grinspun. **Reflections on simultaneous impact**. *ACM Trans. Graph.*, 31(4):106:1–106:12, July 2012. ISSN 0730-0301. doi:10.1145/2185520.2185602

Samantha Ainsley, Etienne Vouga, Eitan Grinspun, and Rasmus Tamstorf. **Speculative parallel asynchronous contact mechanics**. *ACM Trans. Graph.*, 31(6):151:1–151:8, November 2012. ISSN 0730-0301. doi:10.1145/2366145.2366170

Eder Miguel, Rasmus Tamstorf, Derek Bradley, Sara C. Schvartzman, Bernhard Thomaszewski, Bernd Bickel, Wojciech Matusik, Steve Marschner, and Miguel A. Otaduy. **Modeling and Estimation of Internal Friction in Cloth**. *ACM Trans. Graph.*, 32(6):212:1–212:10, November 2013. ISSN 0730-0301. doi:10.1145/2508363.2508389

Xiang Ni, Laxmikant Kale, and Rasmus Tamstorf. **Scalable Asynchronous Contact Mechanics using Charm++**. In *IEEE 29th International Parallel and Distributed Processing Symposium (IPDPS 2015)*, May 2015. doi:10.1109/IPDPS.2015.45

Carlos Aliaga, Carol O'Sullivan, Diego Gutierrez, and Rasmus Tamstorf. **Sackcloth or Silk ? The Impact of Appearance vs Dynamics on the Perception of Animated Cloth**. In *ACM Symposium on Applied Perception (SAP '15)*, September 2015. doi:10.1145/2804408.2804412

CHAPTER 2

Related Work

Cloth simulation has been an active area of research in computer graphics for many years. Some of the most popular methods for cloth simulation in computer graphics have been published by [Baraff and Witkin, 1998] and [Bridson et al., 2002]. Numerous improvements to those methods have been proposed, but the fundamental ideas remain largely unchanged. A few exceptions include the work presented by [Harmon et al., 2009] on asynchronous contact mechanics, and the notion of simulating cloth at the yarn level proposed by [Kaldor et al., 2008] and also explored by [Cirio et al., 2014].

The key bottlenecks in the standard methods are collision detection on the one hand, and computation of dynamics and contact response on the other hand. In the following, we primarily focus on dynamics and contact response and briefly review some of the related work.

2.1 Cloth models

To simulate cloth, an appropriate model must first be chosen. A very simple model used in graphics is a mass-spring model based on linear springs. The popularity of this model is due to its simplicity and the fact that it can capture a basic range of behaviors. Sometimes, linear springs are replaced with biphasic springs to account for *crimp* in real fabrics. However, springs lack the ability to model bending in a satisfactory fashion and they cannot capture the anisotropic behavior of cloth.

Given these limitations, we instead assume that cloth can be modeled as a thin shell using an appropriate constitutive model from continuum mechanics and discretized using a finite element based approach (or some approximation thereof). This is essentially what was originally proposed by Terzopoulos et al. [1987].

While the approach conceptually is simple, the derivation of the partial differential equations (PDEs) for thin shells undergoing large displacements with large deformations is quite involved. Historically, the first attempt at modeling thin plates and shells was made by Leonhard Euler around 1760 while trying to calculate the sound made by drums (plates), [Euler, 1766b], and church bells (shells), [Euler, 1766a]. Interestingly, in trying to come up with the equations for the vibrations of the membrane, Euler made the analogy between a membrane and a piece of cloth. In fact, he wrote in [Euler, 1766b, §2] :

Quo igitur huiusmodi superficiei motum, ac potissimum tensionem, in calculum facilius inducere possimus, ad eius generationem, textura filorum secundum longitudinem et latitudinem expansorum ortam respici conueniet, cum hac ratione lintea reuera efficiantur, in membranis autem filorum numerum quasi infinitum concipere liceat.

A translation of this is provided by Truesdell in [Euler, 1960, p. 331] :

It is appropriate to regard a stretched cloth or membrane as composed of threads along its length and breadth, as is really the case for a cloth, while in a membrane the number of threads is to be regarded as infinite.

It turns out that the analogy to threads (strings) was flawed and didn't quite work out, but it remains notable that Euler literally started by thinking about cloth ! Other people came after him to try to fix the PDE (most notably Jakob II Bernoulli who also got it wrong while using a grid of beams, [Bernoulli, 1789]), but it took another 50 years before Sophie Germain finally got the thin plate equations right with some help from Joseph-Louis Lagrange. The work by Bernoulli and Germain was largely inspired by the (now classical) experiments presented by Chladni [1787] on resonance patterns of thin plates, and in another twist of history, these experiments illustrate exactly what the near kernel components are for the thin plate equation. As we shall see later, these components are of utmost importance in multigrid theory.

A somewhat gentle and modern introduction to the many difficulties of thin plates and shells is presented by Audoly and Pomeau [2010], while a more classic book is the one by Timoshenko and Woinowsky-Krieger [1959]. Much of the exposition by Audoly and Pomeau [2010] is based on the Föppl-von

Kármán equations, which were originally proposed in 1910. However, some of the implicit assumptions for these equations are questionable, [Ciarlet, 1980].

One complication in much of shell theory is the justification for the transition from 3D elasticity to a simplified 2D theory. Another complication is the fact that shells are inherently best described in curvilinear coordinates, and there is a delicate interplay between differential geometry and physics that is required for a correct formulation of the equations. This interplay has been carefully presented by Ciarlet [2000].

In order to arrive at tractable equations, many assumptions and approximations have to be made. Among the resulting models, Koiter's thin shell model, [Koiter, 1966], has proven to be surprisingly versatile despite being relatively simple. Unlike the Föppl-von Kármán equations, it has also been confirmed to be a proper 2D limit model of 3D nonlinear elasticity, [Steigmann, 2013].

Koiter's model is based on the Kirchhoff-Love assumptions, which do not allow for shear stresses across the thickness of the shell nor for changes in the thickness of the shell. A less restrictive set of assumptions is known as the Reissner-Mindlin plate theory, but for our purposes the Kirchhoff-Love assumptions will suffice. A more problematic assumption in Koiter's original model is that all strains are assumed to be small. For cloth simulation, this is generally not the case; neither for bending nor for shearing within the surface of the shell. This deficiency has recently been addressed by Steigmann [2012].

In Koiter's model, the energy can be written as a the sum of two terms : a membrane energy and a bend energy. The membrane energy is a function of the change of strain (first fundamental form) while the bend energy is a function of the change of curvature of the surface (second fundamental form). This basic setup has motivated much work in the graphics community, where the two contributions are typically modeled separately without too much concern about consistency between the two.

As an example, Grinspun et al. [2003] and Bridson et al. [2003] introduced bending models to improve upon mass-spring systems. Separately Delingette [2008] showed how a constant strain triangle discretization of an isotropic (continuum) membrane model can be thought of as a combination of tensile and angular springs such that the overall system still appears to be a mass-spring system.

In order to capture the anisotropic behavior of fabric, Etzmuß et al. [2003] was the first in graphics to consider an orthotropic constitutive model for the

membrane part of the cloth model. However, this was limited to co-rotated linear elasticity. Orthotropy for fabric was simultaneously considered in the textile literature by Wu et al. [2003] but only in the context of explicit integration. Volino et al. [2009] later considered a simplified orthotropic model that captures the independent stiffnesses in the warp and weft directions of a fabric, but assumes that Poisson's ratio is zero. Unlike previous papers, however, they did consider both implicit integration and the geometrically nonlinear Green-Lagrange strain.

In mechanical engineering, the models used for deformation of fabric tend to be more sophisticated. The interest in that community focuses on carbon-fiber reinforcement for lightweight structures and body armor, but much of the work is directly applicable for cloth simulation, [King, 2006; Williams, 2010]. Most engineering models are still based on orthotropic thin shells, but some have observed that the interaction between the threads in the cloth under shearing causes an additional stress field that is not accounted for by the standard orthotropic model, (see Zhang and Fu [2000, 2001]). Finally, there is much experimental data to suggest that the material response for fabric is highly nonlinear. An orthotropic model that can fit such data has been proposed by Itskov [2001]; Itskov and Aksel [2004] but it does not appear to have been applied specifically to fabric. (It was developed for biological tissue, specifically arteries.)

Given the continuous equations, there are many different ways to form a corresponding set of discrete equations. This particular problem is not the primary focus of this work. However, it should be noted that one especially challenging problem related to the discretization of thin shells is membrane locking. This can cause the shell to exhibit excessive stiffness, which leads to unrealistic simulations. For *linearized* elasticity (i.e., small deformations) the problem can be addressed with high-order mixed finite element methods. However, for nonlinear elasticity, no conclusive results exist. In the finite element community, there is a strong belief that quadrilateral elements are superior to triangular elements when it comes to locking and general convergence properties, but quadrilateral elements introduce complications with respect to collision detection, so most methods in graphics focus on triangular elements. English and Bridson [2008] proposed to deal with the problem by using a non-conforming discretization.

2.2 Time integration

Regardless of the constitutive model and the spatial discretization, the resulting ODEs must be integrated in time. The choice of implicit vs. ex-

plicit integration methods has been the topic of much research. An implicit method needs to solve a global problem, which is typically in the form of a large set of nonlinear equations. An explicit method does not require such a solve and is therefore much faster in terms of computing the update for a single timestep. However, due to the CFL condition, the timesteps in explicit methods usually have to be very small in order to maintain stability, so, in the end, implicit methods can be more desirable. The implicit method has the added advantage that it can resolve all collisions simultaneously such that the response to one collision can factor into responses made to other collisions. In this work, we focus on implicit solvers along the lines of what was introduced by Baraff and Witkin [1998], and refer to Harmon et al. [2009] and the follow-up work in Ainsley et al. [2012] and Ni et al. [2015] for recent ideas about how to leverage explicit integration methods to provide simulations with correctness and termination guarantees.

For implicit integration, one has to solve a set of nonlinear equations for each time step. In the work by Baraff and Witkin [1998], this is crudely approximated by simply solving the corresponding linearized set of equations. In the graphics literature, a lot of the focus has been on using the conjugate gradient (CG) method for this and, as a consequence, various researchers have investigated methods for coming up with good preconditioners for CG-based cloth solvers (e.g., [Boxerman and Ascher, 2004] and [Hauth et al., 2003]). However, the results have been mixed and, in practice, CG does not scale well with the size of the problem. As a consequence, some authors have considered using a direct solver like PARDISO, [Goldenthal et al., 2007; English and Bridson, 2008].

2.3 Multigrid methods

An alternative for preconditioning or even for solving the entire system of equations for time integration is to use multigrid methods. These methods are known to have $O(n)$ complexity for the kind of problems for which they are applicable, which is substantially better than a standard CG algorithm or even CG with a simple preconditioner.

Some of the earliest work on multigrid for thin shells appears in [Fish et al., 1996] where they investigated an unstructured multigrid method for thin shells. This effectively amounted to an extension of geometric multigrid that coarsened with knowledge of the problem. However, they acknowledge that their method has limitations in that it does not address higher-order interpolation and it was never tested on large-scale problems.

More recently, [Gee, 2004; Gee et al., 2005] addressed more complicated shell models for problems in statics using a finite element discretization. They used an aggregation-based approach that has many similarities to an smoothed aggregation (SA) hierarchy. However, for their method, they treat the shell as a (thin) 3D solid, unlike the typical 2D manifold approach used for cloth. To avoid severe ill-conditioning and to obtain convergence rates independent of conditioning, they apply "scaled director preconditioning". This also allows them to model varying thickness across the shell. In their follow-up work, [Gee and Tuminaro, 2006], the focus is on using a nonlinear solver, and adaptive SA is used to precondition the linearizations. However, adaptive SA is currently too expensive for typical cloth problems, although this may change in the future with improvements in the adaptive algorithm and demands for much more computationally intense simulations.

Related research within the graphics community has been focused primarily on applying various types of geometric multigrid (GMG) to cloth simulation. Oh et al. [2008] implemented a GMG method that uses CG for the smoother on each level, with a focus on preserving energy and mass at all levels of the hierarchy. Linear interpolation is used between levels, and the level hierarchy is attained through regular subdivision, while constraints are only treated on the fine grid. This approach produced a significant speedup for their simulations but failed to show linear scaling in the size of the problem, and the performance deteriorated in the presence of constraints. Lee et al. [2010] used a multi-resolution approach that focused on using adaptive meshing to only place subdivisions where needed to provide acceleration, as opposed to a GMG method that uses regular subdivision. It is not clear how to use this approach in a multilevel fashion, nor how it addresses the difficulties arising from the PDE and collisions. Jeon et al. [2013] extended the work in [Oh et al., 2008] to handle "hard" constraints, as presented in [Baraff and Witkin, 1998], by converting them to soft constraints to avoid the challenges of coarsening them. They use GMG as a direct solver, with preconditioned conjugate gradients (PCG) as their smoother for restriction and GS as their smoother for prolongation, but require an expensive V(5,10) cycle.

One of the exceptions to the use of geometric multigrid in the graphics literature is an AMG-like algorithm developed by Krishnan et al. [2013]. Their focus is on discrete Poisson equations written in terms of *M-matrices*, which (among other properties) have only nonpositive off-diagonal entries. Basically, their approach is to modify the original matrix by first selecting unknowns that represent the coarse grid and then eliminating interconnections between the remaining fine-grid-only unknowns. This elimination is accomplished in a *sparsification/compensation* process that is analogous to AMG's so-called *stencil collapsing* procedure. The difference is that, while AMG

eliminates these connections to determine interpolation, their goal is to produce a modified matrix (to be used as a preconditioner) that naturally preserves M-matrix character on coarser levels. Their method is shown to be superior to two algebraic multigrid techniques that were modified to similarly preserve M-matrix character, so it is no doubt important in some applications. However, their results show a loss of efficiency compared to more standard algebraic multigrid methods. Our aim here is to develop an AMG approach that is not restricted to M-matrix discretizations.

2.4 Contact handling

Adding contact and friction to the dynamics equations complicate the computations significantly. Here and in the following, the term "contact" is used to refer to non-penetration constraints and is relatively speaking easier than handling friction. Many existing implicit cloth solvers are based on the modified conjugate gradient (MCG) algorithm described by Baraff and Witkin [1998]. This method can incorporate *equality* constraints, but only for contact between cloth vertices and faces of collision objects. Unfortunately, this is somewhat limiting since it allows small collision-faces to completely penetrate through a coarse cloth mesh. Such a scenario occurs frequently when, for example, fingers interact with clothing.

The MCG algorithm by itself falls in the category of generalized conjugate direction methods. A number of authors prior to Baraff and Witkin [1998] considered how to use these methods to solve linear systems subject to general equality constraints (see Dennis and Turner [1987]; Shariff [1995]; Gould et al. [2001] among others). While this prior work should be suitable for including edge-edge and face-vertex constraints this has not actually been done for cloth simulations. Also, it should be emphasized that this would still be limited to equality constraints.

It should be noted that one of the reasons the MCG method is so popular is that it can accommodate the ever changing structure of the equations as constraints from collisions are added and removed. Direct solvers by comparison typically have to do a symbolic factorization, which depends on the sparsity pattern of the system, and since this factorization is expensive to perform, these methods can be less advantageous when the sparsity pattern constantly changes.

An alternative to incorporating the constraints directly into the solver was presented by Bridson et al. [2002]. In that method, an update due to internal dynamics is first computed, and contacts are then resolved by applying

impulses in Gauss-Seidel or Jacobi fashion until all constraints are satisfied. However, there is no guarantee that all constraints will be satisfied, so, to ensure termination, the paper also introduces a *failsafe*. Furthermore, the decoupling between the dynamics and collision response can lead to artifacts unless very small timesteps are used.

Another approach that has been proposed is to use the volume of overlap between thickened elements to create penalty forces (e.g. [Heidelberger et al., 2003; Faure et al., 2008]). However, this is somewhat problematic for implicit integration methods because the second derivative of the the overlap volume with respect to positions is zero. To see this, note that the overlap volume is linear in the separation distance, which means that the first derivative wrt. positions is constant, so the second derivative is zero. The force Jacobian needed for an implicit integration scheme is based on this second derivative.

In a more general setting, the contact problem can be stated as a complementarity problem. The complementarity ensures that there is either is a gap between two objects or there is a contact force. This formulation leads to inequality constraints, which are more challenging to handle, but avoids "sticking artifacts" associated with equality constraints. In practice, the complementarity problem is often linearized to obtain an LCP (linear complementarity problem). However, as shown by Smith et al. [2012], even linear LCPs in general suffer from sticking artifacts for anything but inelastic collisions. In practice, this may not be a big limitation for cloth, since most cloth collisions are fairly inelastic.

In the context of cloth simulation, Otaduy et al. [2009] used an LCP to solve for cloth dynamics and contact simultaneously. By combining the LCP formulation with linear compliance, the collision response produced by this method is aware of the underlying dynamics, which was shown to reduce "jitter" artifacts. However, as we will show, the method still has problems handling high speed impacts due to the linearization.

Adding friction makes all of the above more complicated. Partly because our understanding of friction phenomena (tribology) is still far from complete, but partly also because even a simple friction model interacts with the non-penetration constraints in subtle but critical ways. An overview of some of the problems with contact and friction can be found in Stewart [2000]. Recently, it was shown by Anitescu and Tasora [2010] that the combined contact and friction problem for rigid bodies can be approximated as a "cone complementarity problem" (CCP). This has successfully been applied to rigid body simulation by Mazhar et al. [2015], but not to thin elastic objects.

CHAPTER 3

Continuum mechanics

In this chapter, we introduce some of the basics of continuum mechanics needed in subsequent chapters. In particular, we focus on the theory of invariants and develop the foundation for constitutive models for 2D orthotropic membranes. Much of the existing literature is written for 3D solids, so the contribution here is to extract out the key concepts from the general theory and apply them in the 2D setting.

In the next chapter, we apply this theory to derive an orthotrophic St. Venant-Kirchhoff constitutive model for membranes. The St. Venant-Kirchhoff constitutive model is an extension of the linear elastic model to handle large deformations through the use of Green's strain tensor. However, as in the linear elastic model, the elasticity tensor is assumed to be constant.

3.1 Basics of continuum mechanics

Elastic materials are generally those that return to their original shape after being deformed. For any material point within an elastic material, the stress is only a function of the local state of deformation. This is unlike fluids, which also depend on the velocity field.

The deformation is given through the deformation function, ϕ, which maps a point, $\bar{x}$, in the undeformed material space, Ω, to the corresponding point, x, in the deformed space. For a thin plate (or membrane), the material space

is inherently two dimensional since the rest configuration is flat, but the plate is typically embedded in three-dimensional space, so we usually have $\phi : \Omega = \mathbb{R}^2 \to \mathbb{R}^3$. It should be noted that the material space can also be embedded in a higher-dimensional space, but all the information about the actual deformation in that case is contained in a two-dimensional subspace.

Since we assume that the medium can be treated as a continuum, the local state of deformation can be characterized by the deformation gradient :

$$\boldsymbol{F} = \nabla \phi$$

In our setting, we end up with $\boldsymbol{F} \in \mathbb{R}^{3\times 2}$.

In the following we focus on the subset of all elastic materials known as hyperelastic, for which the existence of an energy-density, ψ, is assumed or postulated. The existence of such an energy-density implies that these materials are conservative, unlike the more general Cauchy-elastic materials, for which the only restriction is that the stress is a function of $\boldsymbol{F}$.

Assuming that the material is conservative rules out any dissipation, which means that damping and hysteresis cannot be taken into account. These phenomena are important for cloth simulation, but can be added back in later.

A fundamental principle in elasticity is the notion of "material frame indifference", [Truesdell and Noll, 1965]. This is sometimes also referred to as "objectivity"[1] and basically states that the behavior of the system should not depend on the choice of the reference (coordinate) frame. It can be shown that a consequence of material frame indifference is that ψ must be a function of the (right) Cauchy-Green deformation tensor, $\boldsymbol{C} = \boldsymbol{F}^T\boldsymbol{F}$. This is important because it implies that if a proposed energy density cannot be written as a function of $\boldsymbol{C}$, then the model depends on the choice of coordinate system. Another consequence of this principle is that the energy must be invariant under rotations and translations, which we use later.

The equivalent of Newton's second law for a continuum is Cauchy's first law of motion :

$$\nabla \cdot \boldsymbol{\sigma} + \rho \boldsymbol{b} = \rho \dot{\boldsymbol{v}}$$

where $\boldsymbol{\sigma}$ is the Cauchy stress, ρ is the mass density, and $\boldsymbol{b}$ is a vector field of body forces (typically gravity). For solid elasticity, these equations amount

[1] There has been much debate about the equivalence between "material frame indifference" and "objectivity" or lack thereof, see e.g. [Liu, 2005; Noll, 2005]. However, in this context, it appears safe to treat them as equivalent.

to three coupled second order PDEs. The thin shells equations derive from these equations, but are more complicated.

In Newtonian mechanics, it is well known that the the rate of change of the angular momentum equals torque. In continuum mechanics, the equivalent statement is known as the balance of moment of momentum. Unlike in Newtonian mechanics, it is an independent hypothesis in continuum mechanics. Combined with Cauchy's first law of motion, it leads to the conclusion that the Cauchy stress and (consequently) the second Piola-Kirchhoff stress must be symmetric [Marsden and Hughes, 1983, Theorem 2.10]. This statement is also referred to as Cauchy's second law of motion.

We rely on the symmetry of the stress tensor in the following and also on the symmetry of the Green-Lagrange strain. The Green-Lagrange strain is given by

$$\boldsymbol{E} = \frac{1}{2}(\boldsymbol{C} - \boldsymbol{I})$$

which is clearly symmetric. By definition, the strain tensor is a mapping of two vectors from the tangent space of the domain for the deformation function into the reals. The Green-Lagrange strain specifically maps two vectors in material space to their dot product in the deformed space. Other strain tensors can be defined based on the principal stretches given by the polar decomposition of $\boldsymbol{F}$, see e.g. [Ogden, 1984, Sec. 2.2.7].

In classical mechanics, energy equals force times displacement, from which it follows that force is the derivative of energy wrt. displacement. The force and the displacements are also said to be *energy-conjugate*. This notion of energy-conjugate variables can be generalized and, in continuum mechanics stress and strain is an example of such a pair of energy conjugate variables. For a hyperelastic material, this leads to[2]

$$\boldsymbol{P} = \frac{\partial \psi}{\partial \boldsymbol{F}}, \quad \boldsymbol{S} = \frac{\partial \psi}{\partial \boldsymbol{E}}$$

where $\boldsymbol{P}$ and $\boldsymbol{S}$ are the first and second Piola-Kirchhoff stress tensors. The second Piola-Kirchhoff stress and the Cauchy stress are related by

$$\boldsymbol{\sigma} = \frac{1}{J}\boldsymbol{F}\boldsymbol{S}\boldsymbol{F}^T$$

where $J = \det \boldsymbol{F}$. Given the choice of strain measure and energy density, the stress is given by the above expressions. Alternatively, the constitutive model can be given by way of the stress-strain relationship, from which the

[2]The derivatives here require a fairly sophisticated mathematical machinery to be defined in a rigorous fashion. We refer to Marsden and Hughes [1983] for details.

energy density is then derived using the tensor (double contraction) product :

$$\psi = \boldsymbol{P} : \boldsymbol{F} = \boldsymbol{S} : \boldsymbol{E}$$

The double contraction between two second-order tensors $\boldsymbol{A}$ and $\boldsymbol{B}$ is defined as $\boldsymbol{A} : \boldsymbol{B} = \mathrm{tr}(\boldsymbol{A}^T\boldsymbol{B}) = \mathrm{tr}(\boldsymbol{A}\boldsymbol{B}^T) = \sum \boldsymbol{A}_{ij}\boldsymbol{B}_{ij}$. The two methods for specifying the constitutive model are equivalent. However, it should be noted that the stress and strain measures must be energy-conjugate.

3.2 The elasticity tensor

Doing a Taylor expansion of the strain energy density, $\psi(\boldsymbol{E}) = \boldsymbol{S} : \boldsymbol{E}$, around the undeformed configuration gives

$$\psi(\boldsymbol{E}) = \psi_0 + \boldsymbol{S}_0^{ij}\boldsymbol{E}_{ij} + \mathbb{C}^{ijkl}\boldsymbol{E}_{ij}\boldsymbol{E}_{kl}$$

Here, $\boldsymbol{S}_0^{ij}$ is the (second Piola-Kirchhoff) stress in the undeformed configuration, which is usually assumed to be zero, and if we set the energy of the undeformed configuration to zero, it follows that $\psi_0 = 0$. This leads to :

$$\psi(\boldsymbol{E}) = \mathbb{C}^{ijkl}\boldsymbol{E}_{ij}\boldsymbol{E}_{kl}$$

where

$$\mathbb{C}^{ijkl} = \frac{\partial \boldsymbol{S}}{\partial \boldsymbol{E}} = \frac{\partial^2 \psi(\boldsymbol{E})}{\partial \boldsymbol{E}_{ij}\partial \boldsymbol{E}_{kl}} = 4\frac{\partial^2 \psi(\boldsymbol{C})}{\partial \boldsymbol{C}_{ij}\partial \boldsymbol{C}_{kl}}$$

is referred to as the elasticity tensor. In 3 dimensions, $\mathbb{C}^{ijkl}$ has $3^4 = 81$ components, while it has $2^4 = 16$ components in 2 dimensions. However, due to the symmetry of the stress tensor, $\mathbb{C}^{ijkl}$ is symmetric in its first pair of indices :

$$\mathbb{C}^{ijkl} = \mathbb{C}^{jikl}$$

Similarly, the symmetry of the strain tensors results in symmetry of the second pair of indices :

$$\mathbb{C}^{ijkl} = \mathbb{C}^{ijlk}$$

These are called minor symmetries. Since the order of differentiation with respect to $\boldsymbol{E}_{ij}$ and $\boldsymbol{E}_{kl}$ doesn't matter, it also follows that

$$\mathbb{C}^{ijkl} = \mathbb{C}^{klij}$$

which is called the major symmetry. This symmetry is a consequence of the existence of a (hyperelastic) energy density. Given these symmetries, it is relatively easy to show using the definition of the double contraction that

$\mathbb{C}\boldsymbol{A} : \boldsymbol{B} = \boldsymbol{A} : \mathbb{C}\boldsymbol{B}$ for all symmetric matrices $\boldsymbol{A}$ and $\boldsymbol{B}$. In He and Zheng [1996], a tensor, $\mathbb{C}$, with this property is referred to as a *hyperelastic tensor*. By contrast, an *elastic tensor* represents a "Cauchy-elastic" material and need not possess major symmetry. In the following, we only consider hyperelastic materials, and will refer to $\mathbb{C}$ simply as the elasticity tensor.

To illustrate the symmetry, we write the two dimensional elasticity tensor explicitly as a matrix of matrices. The small matrices are indexed by (i, j), while the big matrix is indexed by (k, l). Letters indicate unique values and the full index is given under each element :

$$\mathbb{C}^{ijkl} = \begin{pmatrix} \mathbb{C}^{ij00} & \mathbb{C}^{ij01} \\ \mathbb{C}^{ij10} & \mathbb{C}^{ij11} \end{pmatrix} = \left(\begin{pmatrix} \underset{0000}{a} & \underset{0100}{c} \\ \underset{1000}{c} & \underset{1100}{b} \end{pmatrix} \begin{pmatrix} \underset{0001}{c} & \underset{0101}{f} \\ \underset{1001}{f} & \underset{1101}{e} \end{pmatrix} \right) \left(\begin{pmatrix} \underset{0010}{c} & \underset{0110}{f} \\ \underset{1010}{f} & \underset{1110}{e} \end{pmatrix} \begin{pmatrix} \underset{0011}{b} & \underset{0111}{e} \\ \underset{1011}{e} & \underset{1111}{d} \end{pmatrix} \right)$$

As can be seen, there are only 6 unique entries. In general, a symmetric second-order tensor has $\frac{1}{2}n(n+1)$ unique elements, so since $\mathbb{C}$ is a linear map between two symmetric second order tensors, it follows that it can have at most $\frac{1}{4}n^2(n+1)^2$ unique elements. This accounts for the minor symmetries. Adding the major symmetry reduces the number further, and in 3d it ends up being 21. This number as well as the one in 2d is reduced as various material symmetries are introduced and, for a completely isotropic material, only two parameters are necessary to fully specify $\mathbb{C}$.

As stated above, we can also write $\mathbb{C} = \partial \boldsymbol{S} / \partial \boldsymbol{E}$, which explains why $\mathbb{C}$ is sometimes also referred to as the tangent modulus. It is the tangent slope of the stress-strain "curve". For linear elasticity, $\mathbb{C}$ is constant, while in general it depends on $\boldsymbol{E}$ for nonlinear materials.

The tangent modulus is conceptually very similar to the force gradient, since they are both second derivatives of the energy. However, it should be noted that the force gradient is a derivative with respect to positions, while the tangent modulus is a derivative with respect to the strain. Obviously, the two are related through the chain rule, but not in a particularly simple way.

3.3 Strong ellipticity

The only restriction stated on ψ in the previous sections is that it must be a function of $\boldsymbol{C}$. However, to model physics in a plausible fashion, the system

as a whole should (ideally) satisfy a number of additional conditions. In particular, [Antman, 2005, Sec. 13.2] lists the following desiderata :

1. An increase in a component of strain should be accompanied by an increase in the corresponding component of stress.
2. Extreme strains (those for which an eigenvalue of $\boldsymbol{C}$ is 0 or ∞) should be maintained by infinite stresses. (In an extreme strain, $|\boldsymbol{C}| = \infty$ or $\det \boldsymbol{C} = 0$.)
3. The equations of motion should admit solutions with wave-like behavior.
4. Well-set initial-boundary-value problems for the equations of motion should have solutions.
5. For appropriate data, the equilibrium equations should admit multiple solutions, so that buckling can be described.
6. Solutions should have an appropriate level of regularity.

The first of these criteria is also known as Hill's stability criterion, [Hill, 1958], or occasionally as Drucker's stability postulate, Drucker [1957][3].

Unfortunately, it is not obvious how to achieve all of the above criteria. One restriction that would address the first desideratum would be to require that ψ is strictly convex. However, as pointed out in Hill [1957], this would preclude the possibility of buckling due to the non-uniqueness of such solutions. Thus, we must conclude that it is not physically plausible to require strict convexity of ψ.

A less restrictive class of functions are characterized by *quasiconvexity*, and it was shown by Morrey [1952], that subject to certain restrictions, quasiconvexity can guarantee existence for some boundary value problems related to nonlinear elasticity.

If a function is both quasiconvex and twice continuously differentiable, then it is said to be "strongly elliptic" and satisfy the *Strong Legendre-Hadamard Condition of the Calculus of Variations*. Throughout the literature, strong ellipticity and the Legendre-Hadamard condition are often used almost interchangeably, but strong ellipticity can also apply to non-differentiable functions. We refer to [Antman, 2005, Sec. 13.3] for the details.

Conceptually, the important aspect is that this condition ensures that "the $\boldsymbol{ab}$-component of the first Piola-Kirchhoff stress tensor is an increasing function of the corresponding component of $\boldsymbol{F}$". Here, $\boldsymbol{a}$ and $\boldsymbol{b}$ are arbitrary

[3]The work by Drucker might precede that of Hill, but it was classified by the US Navy until 1977.

vectors, and the $\boldsymbol{ab}$-component of $\boldsymbol{P}$ is $\boldsymbol{a}^T\boldsymbol{P}\boldsymbol{b}$. Note that $\boldsymbol{P}$ maps a direction (a normal of a surface element) in the undeformed configuration to a force in the deformed configuration (normalized by surface area in the undeformed configuration to give a stress). Thus, $\boldsymbol{f} = \boldsymbol{P}\boldsymbol{b}$ gives the force associated with the direction $\boldsymbol{b}$ in the undeformed configuration. Consequently, $\boldsymbol{a}^T\boldsymbol{P}\boldsymbol{b}$ is the component of $\boldsymbol{f}$ in the $\boldsymbol{a}$ direction in the deformed space. Similarly, $\boldsymbol{a}^T\boldsymbol{F}\boldsymbol{b}$ computes the deformation gradient associated with direction $\boldsymbol{b}$ in material space and the component along the direction $\boldsymbol{a}$ in the deformed space. In other words, if the deformation gradient for a given pair of directions is increasing, then the associated stress (force) should also be increasing.

An alternative interpretation of strong ellipticity is given by Rivlin [1992] (also available in [Rivlin, 1997]), who shows that strong ellipticity is a necessary and sufficient condition for *material stability*. Here, an elastic material is said to be "stable at some specified value of the deformation gradient $\boldsymbol{F}(\neq \boldsymbol{I})$ if and only if a body of the material can be held stably in this state of homogeneous deformation with no applied body forces and the boundary held fixed."

Strong ellipticity on its own in not quite sufficient to guarantee existence of solutions to nonlinear elasticity as pointed out in the seminal paper by Ball [1976]. To that end, one must also add *coercivity* conditions as outlined in [Antman, 2005, Sec. 13.4]. Ball [1976] showed that one can achieve all of the necessary conditions by requiring that ψ be *polyconvex*, and this has led to a significant amount of work related to constitutive modeling using polyconvex functions.

We do not delve into this any further. The key point to make note of here is that ψ cannot in general be expected to be convex and, as a consequence, the associated energy Hessian cannot in general be expected to be positive definite. However, we *should* expect ψ to be strongly elliptic, and there is a nice physical interpretation of what that means. Strong ellipticity also relates to the ellipticity of the PDE we ultimately would like to solve, which facilitates the application of multigrid later in the thesis. The connection between strong ellipticity and PDEs is covered in Yan [1999].

3.4 Invariants

Another desideratum for the hyperelastic energy is given by Neumann's principle, [Neumann, 1885]. This states that *"the symmetry group of a considered material must be included in the symmetry group of any tensor function of the constitutive laws of the material"*. Hence, if the material is described by a

symmetry group, $\mathcal{G}$, then it must be true that

$$\psi(\boldsymbol{F}) = \psi(\boldsymbol{Q}\boldsymbol{F}\boldsymbol{Q}^T), \quad \forall \boldsymbol{Q} \in \mathcal{G} \tag{3.1}$$

Note, however, that the implication only goes one way. The constitutive law may possess additional symmetries that are not directly associated with the material. Another way of stating this is that two materials with different symmetries may be indistinguishable from a mechanical point of view. An example of this is a piece of plywood with 6 layers oriented with a 6-fold rotation symmetry. Mechanically, this is indistinguishable from an isotropic material, [He and Zheng, 1996, Sec. 4.2].

If we *assume* that the energy density can be written as a polynomial of the components of the tensors upon which it depends, then Neumann's principle implies that the energy density can be written in terms of an *integrity basis*. The notion of an integrity basis is rooted in "The first main theorem of invariant theory" by Hilbert, but can be stated as follows : *"an integrity basis is a set of polynomials, each invariant under the group of transformations, such that any polynomial function invariant under the group is expressible as a polynomial in elements of the integrity basis"* (quote from Betten [1982]).

The number of elements in the integrity basis depends on the symmetry (i.e., the group of invariant transformations) as well as the type of function in question (scalar, vector, or tensor valued). It is in general desirable to find an *irreducible* basis to allow the constitutive parameters to be uniquely identified.

Establishing an irreducible integrity basis for a given transformation group is non-trivial, but has fortunately already been done for all the transformation groups we care about here. In particular, Zheng [1993b] lists irreducible sets of bases for all the functions of two-dimensional tensor needed for shells. As with any other basis, it should be noted that, although a basis may be irreducible, it is usually not unique. We shall see an example of that later in Section 4.4.

The assumption about the energy density being a polynomial function of the tensor components can be lifted by considering arbitrary functions of the integrity basis (see Wineman and Pipkin [1964]). In this case, however, a minimal integrity basis is not guaranteed to lead to a minimal functional basis, but it will remain a basis, which makes it convenient to focus on irreducible integrity bases for constitutive modeling.

For isotropic scalar functions of 3-dimensional second-order tensors, there are two different bases that are commonly used. The *main* invariants are denoted by J and the *principal* invariants by I. These are given by

Main invariants in 3D	*Principal invariants in 3D*
$J_1 = \mathrm{tr}(\boldsymbol{C}) = I_1$	$I_1 = \mathrm{tr}(\boldsymbol{C}) = J_1$
$J_2 = \mathrm{tr}(\boldsymbol{C}^2) = I_1^2 - 2I_2$	$I_2 = \mathrm{tr}(\mathrm{Cof}(\boldsymbol{C})) = \frac{1}{2}(J_1^2 - J_2)$
$J_3 = \mathrm{tr}(\boldsymbol{C}^3) = I_1^3 - 3I_1I_2 + 3I_3$	$I_3 = \det \boldsymbol{C} = \frac{1}{3}\left(J_3 - \frac{3}{2}J_1J_2 + \frac{1}{2}J_1^3\right)$

Here, $\mathrm{Cof}(\boldsymbol{C})$ is used to denote the cofactor matrix for $\boldsymbol{C}$. The derivation of the relationship between the main invariants and the principal invariants is outlined in Schröder [2010]. For two-dimensional problems, the invariants are given by the first two lines in the table above, but I_2 may conveniently also be written in terms of the determinant :

Main invariants in 2D	*Principal invariants in 2D*
$J_1 = \mathrm{tr}(\boldsymbol{C}) = I_1$	$I_1 = \mathrm{tr}(\boldsymbol{C}) = J_1$
$J_2 = \mathrm{tr}(\boldsymbol{C}^2) = I_1^2 - 2I_2$	$I_2 = \det(\boldsymbol{C}) = \frac{1}{2}(J_1^2 - J_2)$

In the table above, the invariants are expressed in terms of $\boldsymbol{C}$ since ψ, as explained above, has to be a function of $\boldsymbol{C}$ to ensure objectivity. However, since $\boldsymbol{E}$ is a (non-singular) linear function of $\boldsymbol{C}$, it follows that ψ can be equally well be written in terms of $\boldsymbol{E}$, and the integrity basis is valid for any symmetric second order tensor. This means that we can trivially construct additional sets of invariants by replacing $\boldsymbol{C}$ by $\boldsymbol{E}$ in the above expressions.

In particular, this means that $\mathrm{tr}(\boldsymbol{E})$ and $\mathrm{tr}(\boldsymbol{E}^2)$ form another basis in two dimensions. In the following, we state the main invariants in terms of $\boldsymbol{E}$ rather than $\boldsymbol{C}$.

3.5 Symmetry

To develop anisotropic constitutive models, we briefly recap a few relevant facts about symmetry groups in this section.

It is well known from crystallography that there are 230 symmetry groups in 3D (space groups) and 17 such groups in 2D (plane symmetry groups). The latter are also known as “wallpaper groups” because they represent all the patterns you can put on wallpaper and have it repeat infinitely in two directions. Both space groups and wallpaper groups are generated by an underlying *point group* along with a *translation lattice*. There are 10 different

point groups in 2D and five different lattice types. Together, they combine to generate the 17 unique wallpaper groups.

The point group essentially specifies the symmetry of an elementary cell (fundamental domain) of the pattern, while the translation lattice specifies all translations that leave the overall pattern invariant. Another way of stating this is that if $\mathcal{G}$ is a wallpaper group and $\mathcal{T}$ is the translation lattice, then the associated point group is the quotient group $\mathcal{G}/\mathcal{T}$. An excellent and more in-depth discussion of the group structure of wallpaper patterns can be found in Morandi [2007].

For continuous and homogenous media, there is no underlying translation lattice, so the only thing that really matters is the point group. More specifically, Zheng [1994] states the "Principle of symmetry of continuum" : *Compact point groups describe and classify all kind of real or ideal material symmetry and physical symmetry, while the description of the symmetry of a continuous media or its any physical property by a non-compact point group is an unreality.*

While there are 10 finite point symmetry groups in 2D, there are also two infinite symmetry groups corresponding to continuous rotations, $SO(n)$ and $O(n)$. However, for a continuous and homogeneous medium in 2D, some of these are indistinguishable. In particular, every material symmetry group includes the inversion, [Zheng and Boehler, 1994]. This immediately reduces the number of symmetries to 6, [Zheng, 1994, Eq. 3.22], but He and Zheng [1996] show that, for linear elasticity, there are actually only 4 independent hyperelastic symmetry groups in 2D. These correspond to fully anisotropic, orthotropic, square-symmetric, and isotropic.

The point groups are often labeled using Schoenflies notation, but there are other notations like Hermann-Mauguin's notation, which can also be used. Since the Hermann-Mauguin notation can incorporate translational symmetry elements, it is more general than Schoenflies notation and is usually preferred in crystallography. More recently, the so-called "orbifold" notation has also been introduced, and there are others. However, in the following, we use Schoenflies notation since that seems to be most common in elasticity.

All point groups are subgroups of the orthogonal group. In 2D, they can be further subdivided into two sets of groups : The cyclic groups, C_n, consisting of rotations through $360^\circ/n$, and the dihedral groups, D_n. In 3D, D_n is made up of an n-fold rotation axis (like C_n) and an additional two-fold rotation axis in the plane perpendicular to the n-fold rotation axis. In 2D, this additional two-fold rotation axis effectively amounts to "flipping the object over", which is the same as a reflection as long as you don't keep track of the front-side vs. the back-side of the object. Thus, in 2D, the dihedral groups

contain rotations and reflections. However, in 3D, there is a distinction between reflecting and flipping over. In both 2D and 3D, C_n is a subgroup of D_n.

In 3D, the notation C_{nv} is used to denote C_n with an additional n mirror planes containing the rotation axis. The letter "v" in this notation is short for "vertical", which stems from the fact that the additional mirror planes contain the rotation axis and the rotation axis is assumed to be vertical (by convention). In 2D, C_{nv} is the same as D_n, and the two notations are sometimes used interchangeably. However, it should be noted that, in 3D, C_{nv} and D_n are different groups.

3.6 Anisotropy

Anisotropic materials are often characterized by their symmetry and, as stated earlier by Neumann's principle, any material symmetry must be reflected in the corresponding energy density function, ψ. More specifically this can be expressed through the "Principle of isotropy of space" as originally introduced by Noll [1954] and paraphrased by Zheng [1994] as : *The tensorfunctions in the physical and constitutive laws of an anisotropic material are expressible as isotropic tensor functions with the structural tensors as additional agencies.*

In this context, $\mathcal{G}$ is said to be the symmetry group of a tensor function, ψ, if $\psi(\boldsymbol{C}) = \psi(\boldsymbol{Q}\boldsymbol{C}\boldsymbol{Q}^T)$ for all $\boldsymbol{Q} \in \mathcal{G}$. If the function has multiple arguments, the transformation is applied to all arguments. Given this definition, a tensor function is said to be isotropic if its symmetry group, $\mathcal{G}$, is the full orthogonal group.

A *structural tensor* is characterized by being invariant under all operations in a specified group. As such, it is easily seen that an energy density written as a function of structural tensors satisfies Neumann's principle. The goal of the following section is to identify the proper material symmetry groups for orthotropic material, the associated structural tensor, and the integrity bases for functions of this structural tensor.

3.7 Orthotropy

In the context of cloth simulation, we primarily focus on orthotropy, which is characterized in the rest configuration (in 2D) by two orthogonal directions

that do not have a notion of orientation. I.e., the material has two orthogonal reflection lines and a 2-fold rotation symmetry around the origin. The point symmetry group is therefore D_2 or, equivalently, C_{2v}. In 3D, there are multiple point symmetry groups that are all considered to correspond to orthotropic materials (D_{2h}, C_{2v}, and D_2). See [Zheng, 1994, Section 6].

Given a symmetry group, the question remains of how to find the structural tensors for that group. We omit the details here, but simply refer to Zheng [1993a], which provides a derivation and lists the relevant results for 2D. In particular, he shows that, for a scalar valued function, ψ, of one symmetric second-order tensor, $\boldsymbol{E}$, any orthotropic function can be written as an isotropic function of the following three arguments :

$$J_1 = \operatorname{tr}(\boldsymbol{E}), \quad J_2 = \operatorname{tr}(\boldsymbol{E}^2), \quad J_3 = \operatorname{tr}(\boldsymbol{A}\boldsymbol{E})$$

where $\boldsymbol{A} = \hat{\boldsymbol{a}} \otimes \hat{\boldsymbol{a}}$ is the structural tensor and $\hat{\boldsymbol{a}}$ is a normalized vector in one of the directions of orthotropy. Given the definition of $\boldsymbol{A}$, it is obvious that it is idempotent, so since the trace operator is invariant under cyclic permutation of the matrices, it follows that

$$J_3 = \operatorname{tr}(\boldsymbol{A}\boldsymbol{E}) = \operatorname{tr}(\boldsymbol{A}^2\boldsymbol{E}) = \operatorname{tr}(\boldsymbol{A}\boldsymbol{E}\boldsymbol{A})$$

This formulation later turns out to be more convenient since it preserves the symmetry of the argument to the trace operator. The tensor $\boldsymbol{A}\boldsymbol{E}\boldsymbol{A}$ represents the strain in the direction of $\hat{\boldsymbol{a}}$ since $\boldsymbol{A}$ takes any vector and projects it onto $\hat{\boldsymbol{a}}$.

3.8 Derivatives of invariants

When computing the stress and elasticity tensor associated with a given energy, we need the derivatives of the invariants wrt. $\boldsymbol{E}$. Some care has to be taken here since $\boldsymbol{E}$ is a *symmetric* second-order tensor. However, using Einstein notation, it can be shown that we get

$$\begin{aligned}
\frac{\partial J_1}{\partial \boldsymbol{E}} &= \frac{\partial}{\partial \boldsymbol{E}} \operatorname{tr}(\boldsymbol{E}) = \boldsymbol{I} \\
\frac{\partial J_2}{\partial \boldsymbol{E}} &= \frac{\partial}{\partial \boldsymbol{E}} \operatorname{tr}(\boldsymbol{E}^2) = 2\boldsymbol{E} \\
\frac{\partial J_3}{\partial \boldsymbol{E}} &= \frac{\partial}{\partial \boldsymbol{E}} \operatorname{tr}(\boldsymbol{A}\boldsymbol{E}\boldsymbol{A}) = \boldsymbol{A}^2 = \boldsymbol{A} = \hat{\boldsymbol{a}} \otimes \hat{\boldsymbol{a}}
\end{aligned}$$

3.9 Summary

In summary this chapter established some of the basic requirements for hyperelastic energy functions. As a consequence, we have seen that the hyperelastic energy function for *any* orthotropic 2D material can be written in terms of the following three invariants :

$$J_1 = \mathrm{tr}(\boldsymbol{E}), \quad J_2 = \mathrm{tr}(\boldsymbol{E}^2), \quad J_3 = \mathrm{tr}(\boldsymbol{A}\boldsymbol{E}\boldsymbol{A})$$

where $\boldsymbol{E}$ is the Green-Lagrange strain tensor and $\boldsymbol{A}$ is a tensor which projects any vector in material space onto one of the two orthogonal directions. Importantly, these results are valid for arbitrarily large deformations. In the following, we use this to establish the expressions for an orthotropic St. Venant-Kirchhoff material, but more sophisticated constitutive models also have to be functions of these same three invariants (or an equivalent set).

CHAPTER 4

St. Venant-Kirchhoff membranes

In this chapter, we start by considering the St. Venant-Kirchhoff constitutive model for isotropic and orthotropic materials in the continuum setting. This model is characterized by a constant elasticity tensor similar to linear elasticity, but is based on the nonlinear Green-Lagrange strain, which makes it suitable for handling large displacements.

It should be noted that *no* material that is modeled with a constant elasticity tensor can satisfy the strong ellipticity conditions globally, [Böhlke and Bertram, 2002]; even if one considers other stress and strain measures. Hence, this is a known limitation of the chosen model.

After establishing the model, we derive the expressions necessary for efficient implementation of these models for a constant strain triangle (CST) discretization. Occasionally, constant strain triangles are also referred to as "Turner" triangles because they were introduced by M. J. Turner, Turner et al. [1956]. Constant strain triangles are covered extensively in many finite element texts, but usually only in 2D. In the following, we consider the more general situation where the triangle is embedded in 3D. In particular, we derive the details necessary for force computations and the computation of the force Jacobians.

Our approach follows the ideas in Gingold et al. [2004] for discretizing the strain tensor and ends with results similar to those presented in Delingette [2008]. This approach computes the strain tensor directly from the edges of the triangle, and computes the forces and force Jacobians by direct differentiation of the energy with respect to the vertex positions.

The method presented in this chapter extends the work in Gingold et al. [2004] and Delingette [2008] to handle orthotropy and, unlike the work in Volino et al. [2009], we do not rely on any approximations.

4.1 Bases for symmetric matrices

We start with some basic linear algebra development related to symmetric matrices. This is convenient for representing tensors in matrix form using Voigt notation[1], and for later expressing the strain tensor directly in terms of the edge vectors in a triangle mesh.

Let $\mathbb{S}^n$ be the set of symmetric matrices in $\mathbb{R}^{n\times n}$. Any $\boldsymbol{A} \in \mathbb{S}^n$ can be written in terms of the canonical basis :

$$\begin{aligned}\boldsymbol{A} &= \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{12} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{1n} & a_{2n} & \dots & a_{nn} \end{bmatrix} \\ &= a_{11}\begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 \end{bmatrix} + a_{12}\begin{bmatrix} 0 & 1 & \dots & 0 \\ 1 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 \end{bmatrix} + \dots \\ &\equiv a_{11}\boldsymbol{A}_{11} + a_{12}\boldsymbol{A}_{12} + \ldots + a_{nn}\boldsymbol{A}_{nn}\end{aligned}$$

where the matrices $\boldsymbol{A}_{ij}$ are the basis matrices for $1 \le i \le j \le n$. However, there are many other natural choices. Here, we focus on two such choices. First, let us consider the basis that is the foundation for the Voigt notation, [Moakher, 2008, Sec. 2.4].

Lemma 1. Let $\{\boldsymbol{e}_i\}_{1\le i\le n}$ be an orthonormal basis for $\mathbb{R}^n$ and let δ_{ij} be Kronecker's delta. Then the symmetric matrices

$$\left\{2^{-\frac{1}{2}(1+\delta_{ij})}\left(\boldsymbol{e}_i \otimes \boldsymbol{e}_j + \boldsymbol{e}_j \otimes \boldsymbol{e}_i\right)\right\}_{1\le i\le j\le n}$$

form an orthonormal basis of $\mathbb{S}^n$ with respect to the standard inner product for matrices, $\langle \boldsymbol{A}, \boldsymbol{B}\rangle = \operatorname{tr}(\boldsymbol{A}^T\boldsymbol{B})$.

As an example, for $n = 2$, this basis becomes

$$\{\tilde{\boldsymbol{m}}_1, \tilde{\boldsymbol{m}}_2, \tilde{\boldsymbol{m}}_3\} = \left\{\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}, \frac{1}{\sqrt{2}}\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}\right\}$$

[1] So named after Woldemar Voigt.

For the next proof we need the following lemma :

Lemma 2. Let $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^n \setminus \{0\}$ be two linearly independent vectors. Then $\alpha \boldsymbol{a} \otimes \boldsymbol{a} + \beta \boldsymbol{b} \otimes \boldsymbol{b} = 0$ if and only if $\alpha = \beta = 0$.

Proof. The lemma is clearly true if $\alpha = \beta = 0$. To show the "only if" part, suppose that $\alpha \boldsymbol{a}\boldsymbol{a}^T + \beta \boldsymbol{b}\boldsymbol{b}^T = 0$. Then right multiply by $\boldsymbol{a}$ to obtain

$$\alpha \boldsymbol{a}\boldsymbol{a}^T\boldsymbol{a} + \beta \boldsymbol{b}\boldsymbol{b}^T\boldsymbol{a} = 0$$

Since $\boldsymbol{a}$ is not a null vector, it follows that $\tilde{\alpha} = \alpha \boldsymbol{a}^T\boldsymbol{a} \neq 0$, while $\tilde{\beta} = \boldsymbol{b}^T\boldsymbol{a}$ may or may not be zero. However, due to the linear independence of $\boldsymbol{a}$ and $\boldsymbol{b}$, it follows that $\tilde{\alpha} = \tilde{\beta} = 0$ and, therefore, that $\alpha = 0$. By right multiplying $\alpha \boldsymbol{a}\boldsymbol{a}^T + \beta \boldsymbol{b}\boldsymbol{b}^T = 0$ with $\boldsymbol{b}$, we can similarly conclude that $\beta = 0$. □

Based on this result we now show that another basis for symmetric matrices is given by the following :

Lemma 3. Let $\boldsymbol{x}_i \in \mathbb{R}^n, 0 \leq i \leq n$, be a set of $n+1$ points that form a non-degenerate n-simplex in $\mathbb{R}^n$. Furthermore, let $\boldsymbol{e}_{ij} = \boldsymbol{x}_i - \boldsymbol{x}_j, \quad 0 \leq i < j \leq n$, be the edges of this simplex. Any symmetric matrix, $\boldsymbol{A} \in \mathbb{S}^n$ can then be written as

$$\boldsymbol{A} = \sum_{i<j} \omega_{ij} (\boldsymbol{e}_{ij} \otimes \boldsymbol{e}_{ij})$$

Proof. It is obvious that $\boldsymbol{e}_{ij} \otimes \boldsymbol{e}_{ij} \in \mathbb{S}^n$, so we clearly have $\sum_{i<j} \omega_{ij} (\boldsymbol{e}_{ij} \otimes \boldsymbol{e}_{ij}) \in \mathbb{S}^n$. Next, we show that the set of matrices $\boldsymbol{e}_{ij} \otimes \boldsymbol{e}_{ij}, 0 \leq i < j \leq n$, are linearly independent by showing that the only solution of

$$\sum_{i<j} \omega_{ij} (\boldsymbol{e}_{ij} \otimes \boldsymbol{e}_{ij}) = \boldsymbol{0} \tag{4.1}$$

is $\omega_{ij} = 0$. To this end, pick a vector $\boldsymbol{n}_i$ that is perpendicular to the face opposite $\boldsymbol{x}_i$. If we right-multiply both sides of Eq. (4.1) by $\boldsymbol{n}_i$, then we get zero for all the terms involving the edges incident to the face. The remaining n edges are linearly independent (since the simplex is non-degenerate), so, by the above lemma, it follows that the corresponding ω_{ij} coefficients must all be zero. However, by repeating this argument for all i, we conclude that all ω_{ij} must be zero. Hence, the matrices must be linearly independent. □

Next, consider a transformation of the edge vectors by a skew-symmetric matrix. For skew-symmetric matrices, we have $\boldsymbol{B} = -\boldsymbol{B}^T$, from which we deduce that

$$\boldsymbol{x}^T\boldsymbol{B}\boldsymbol{x} = (\boldsymbol{x}^T\boldsymbol{B}\boldsymbol{x})^T = \boldsymbol{x}^T\boldsymbol{B}^T\boldsymbol{x} = -\boldsymbol{x}^T\boldsymbol{B}\boldsymbol{x} = 0, \quad \forall \boldsymbol{x} \in \mathbb{R}^n$$

If we define $\boldsymbol{e}'_{ij} = \boldsymbol{B}\boldsymbol{e}_{ij}$, it therefore follows that $\boldsymbol{e}_{ij}$ and $\boldsymbol{e}'_{ij}$ are orthogonal, which turns out to be convenient in subsequent derivations (because many terms drop out). If $\boldsymbol{B}$ is non-singular then it also follows that they must form a simplex, since $\boldsymbol{B}$ simply represents a basis change. Thus, we can apply lemma 3 to the transformed set of points. This only works when n is even, because, for n odd, it can be shown that *all* skew-symmetric matrices are singular. However, for $n = 2$ where the simplex is simply a triangle, we can choose

$$\boldsymbol{B} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$$

This matrix is non-singular and represents a 90 degree inplane rotation. Thus, $\boldsymbol{m}_i = \boldsymbol{B}\boldsymbol{e}_i$ represents the inplane edge normals for the triangle. From this result and the above lemma, it follows that any symmetric matrix $\boldsymbol{A} \in \mathbb{S}^2$ can be written as

$$\boldsymbol{A} = \sum_i \omega_i (\boldsymbol{m}_i \otimes \boldsymbol{m}_i).$$

4.2 Isotropic St. Venant-Kirchhoff material

The simplest nonlinear hyperelastic material model is the St. Venant-Kirchhoff model, which states that the strain energy density is given by

$$\psi_m = \frac{\lambda}{2} \operatorname{tr}^2(\boldsymbol{E}) + \mu \operatorname{tr}(\boldsymbol{E}^2),$$

where λ and μ are the Lamé coefficients. Note that Delingette [2008] uses an unconventional variation of the above expression, where μ is divided by 2 (equation (6) in that paper). This is the same energy function as for linear elasticity except that linear elasticity uses the engineering strain rather than the Green-Lagrange strain. In terms of invariants the St. Venant-Kirchhoff model can be written as

$$\psi_m = \frac{\lambda}{2} J_1^2 + \mu J_2.$$

The second Piola-Kirchhoff stress tensor follows from

$$\boldsymbol{S} = \frac{\partial \psi_m}{\partial \boldsymbol{E}} = \lambda \operatorname{tr}(\boldsymbol{E}) \boldsymbol{I} + 2\mu \boldsymbol{E}.$$

For infinitesimally small deformations, we expect the model to be consistent with linear elasticity. In the following, we use this to identify the parameters in terms of the engineering elastic constants (Young's modulus and the Poisson ratio). The elasticity tensor can be computed by differentiation :

$$\mathbb{C} = \frac{\partial \boldsymbol{S}}{\partial \boldsymbol{E}} = \lambda \boldsymbol{I} \otimes \boldsymbol{I} + 2\mu \mathbb{I}^{(s)},$$

where $\mathbb{I}^{(s)}$ is the fourth order symmetric identity tensor. We note that this expression does not depend on $\boldsymbol{E}$ but is in fact constant. The only nonlinearity in the St. Venant-Kirchhoff model is therefore the one introduced by the strain measure.

The (non-symmetric) identity tensor maps its argument (in this case a second order tensor) onto itself. Given an orthonormal basis, this can be written as $\mathbb{I} = \delta_{ik}\delta_{jl}\boldsymbol{e}_i \otimes \boldsymbol{e}_j \otimes \boldsymbol{e}_k \otimes \boldsymbol{e}_l$, or

$$\mathbb{I}^{ijkl} = \left(\begin{pmatrix} \begin{pmatrix} 1 & 0 \\ 0000 & 0100 \end{pmatrix} & \begin{pmatrix} 0 & 0 \\ 1000 & 1100 \end{pmatrix} \\ \begin{pmatrix} 0 & 0 \\ 0010 & 0110 \end{pmatrix} & \begin{pmatrix} 1 & 0 \\ 1010 & 1110 \end{pmatrix} \end{pmatrix} \quad \begin{pmatrix} \begin{pmatrix} 0 & 1 \\ 0001 & 0101 \end{pmatrix} & \begin{pmatrix} 0 & 0 \\ 1001 & 1101 \end{pmatrix} \\ \begin{pmatrix} 0 & 0 \\ 0011 & 0111 \end{pmatrix} & \begin{pmatrix} 0 & 1 \\ 1011 & 1111 \end{pmatrix} \end{pmatrix} \right).$$

The transpose of the identity tensor maps its argument to its transpose, which gives $\mathbb{I}^T = \delta_{jk}\delta_{il}\boldsymbol{e}_i \otimes \boldsymbol{e}_j \otimes \boldsymbol{e}_k \otimes \boldsymbol{e}_l$. By combining the two, we get the symmetric identity tensor, which maps its (potentially non-symmetric) argument to its symmetric part, i.e., $\mathbb{I}^{(s)} = \frac{1}{2}(\mathbb{I} + \mathbb{I}^T)$.

In component form, this gives

$$\mathbb{C}^{ijkl} = \left(\begin{pmatrix} \begin{pmatrix} 2\mu+\lambda & 0 \\ 0000 & 0100 \end{pmatrix} & \begin{pmatrix} 0 & \mu \\ 1000 & 1100 \end{pmatrix} \\ \begin{pmatrix} 0 & \mu \\ 0010 & 0110 \end{pmatrix} & \begin{pmatrix} \mu & 0 \\ 1010 & 1110 \end{pmatrix} \end{pmatrix} \quad \begin{pmatrix} \begin{pmatrix} 0 & \mu \\ 0001 & 0101 \end{pmatrix} & \begin{pmatrix} \mu & 0 \\ 1001 & 1101 \end{pmatrix} \\ \begin{pmatrix} \lambda & 0 \\ 0011 & 0111 \end{pmatrix} & \begin{pmatrix} 0 & 2\mu+\lambda \\ 1011 & 1111 \end{pmatrix} \end{pmatrix} \right)$$

In Voigt-notation, this becomes

$$\tilde{C} = \begin{pmatrix} 2\mu+\lambda & \lambda & 0 \\ \lambda & 2\mu+\lambda & 0 \\ 0 & 0 & 2\mu \end{pmatrix}.$$

Alternatively, we could have observed that the second Piola-Kirchhoff stress and the Cauchy stress are related by

$$\boldsymbol{\sigma} = \frac{1}{J}\boldsymbol{F}\boldsymbol{S}\boldsymbol{F}^T,$$

where $J = \det \boldsymbol{F}$. For small deformations, $\boldsymbol{F} \approx \boldsymbol{I}$ and $J \approx 1$, so $\boldsymbol{\sigma} \approx \boldsymbol{S}$. Furthermore, $\boldsymbol{\epsilon} \approx \boldsymbol{E}$, which then gives

$$\boldsymbol{\sigma} = \lambda \operatorname{tr}(\boldsymbol{\epsilon})\boldsymbol{I} + 2\mu\boldsymbol{\epsilon}.$$

In agreement with the result above, this immediately leads to

$$\begin{bmatrix} \boldsymbol{\sigma}_{xx} \\ \boldsymbol{\sigma}_{yy} \\ \boldsymbol{\sigma}_{xy} \end{bmatrix} = \begin{bmatrix} 2\mu+\lambda & \lambda & 0 \\ \lambda & 2\mu+\lambda & 0 \\ 0 & 0 & 2\mu \end{bmatrix} \begin{bmatrix} \boldsymbol{\epsilon}_{xx} \\ \boldsymbol{\epsilon}_{yy} \\ \boldsymbol{\epsilon}_{xy} \end{bmatrix}$$

In the case of plane stress and small deformations, we can also write the stress-strain relationship in terms of Young's modulus and the Poisson ratio (using the compliance matrix), which gives

$$\begin{bmatrix} \boldsymbol{\epsilon}_{xx} \\ \boldsymbol{\epsilon}_{yy} \\ \boldsymbol{\epsilon}_{xy} \end{bmatrix} = \frac{1}{Y} \begin{bmatrix} 1 & -\nu & 0 \\ -\nu & 1 & 0 \\ 0 & 0 & 1+\nu \end{bmatrix} \begin{bmatrix} \boldsymbol{\sigma}_{xx} \\ \boldsymbol{\sigma}_{yy} \\ \boldsymbol{\sigma}_{xy} \end{bmatrix}$$

Simple inversion then gives

$$\begin{bmatrix} \boldsymbol{\sigma}_{xx} \\ \boldsymbol{\sigma}_{yy} \\ \boldsymbol{\sigma}_{xy} \end{bmatrix} = \frac{Y}{1-\nu^2} \begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & 1-\nu \end{bmatrix} \begin{bmatrix} \boldsymbol{\epsilon}_{xx} \\ \boldsymbol{\epsilon}_{yy} \\ \boldsymbol{\epsilon}_{xy} \end{bmatrix}$$

At this point, we see that

$$\lambda = \frac{Y\nu}{1-\nu^2}, \quad 2\mu = \frac{Y(1-\nu)}{1-\nu^2} = \frac{Y}{1+\nu} \tag{4.2}$$

It is important to note that these are *not* the relationships that are usually tabulated for the conversion between (λ, μ) and (Y, ν), because this is for 2D plane stress as opposed to 3D.

4.3 Orthotropic St. Venant-Kirchhoff material

In order to extend the St. Venant-Kirchhoff constitutive model to an orthotropic material, we follow Bonet and Burton [1998]. We know that the orthotropic energy can be written as a function of the two main invariants plus the mixed invariant :

$$\psi_m = \psi_m(J_1, J_2, J_3)$$

Without loss of generality we can split ψ_m into isotropic orthotropic components :

$$\psi_m = \psi_{\text{iso}}(J_1, J_2) + \psi_{\text{ortho}}(J_1, J_2, J_3) = \frac{\lambda}{2} J_1^2 + \mu J_2 + \psi_{\text{ortho}}(J_1, J_2, J_3)$$

The St. Venant-Kirchhoff model is characterized by being quadratic in the strain, but based on a nonlinear strain function. To extend it to the orthotropic case, we require that the energy remains quadratic in the strain and stress free in the undeformed configuration.

The only combinations of the invariants that yield a quadratic energy are J_1^2, J_2, J_3^2, and $J_1 J_3$. The first two of these are already accounted for in the isotropic part of the energy, so it follows that we must have

$$\psi_{\text{ortho}}(J_1, J_2, J_3) = (\alpha J_1 + \beta J_3)\, J_3.$$

The associated contribution to the stress tensor follows by differentiation :

$$\boldsymbol{S}_{\text{ortho}} = \alpha(J_1 \boldsymbol{A} + J_3 \boldsymbol{I}) + 2\beta J_3 \boldsymbol{A}$$

In the undeformed configuration, we have $J_1 = J_2 = J_3 = \boldsymbol{E} = 0$, so we easily see that the stress is zero in this case. From this we conclude that the energy cannot contain any elements that are linear in the strain. Although this would still keep the energy quadratic, it would cause the stress in the undeformed configuration to be non-zero.

Another round of differentiation yields the elasticity tensor for the orthotropic component :

$$\mathbb{C}_{\text{ortho}} = \frac{\partial \boldsymbol{S}_{\text{ortho}}}{\partial \boldsymbol{E}} = \alpha(\boldsymbol{I} \otimes \boldsymbol{A} + \boldsymbol{A} \otimes \boldsymbol{I}) + 2\beta \boldsymbol{A} \otimes \boldsymbol{A}$$

To arrive at this, we use that :

$$\frac{\partial}{\partial \boldsymbol{E}} J_1 \boldsymbol{A} = \frac{\partial}{\partial \boldsymbol{E}} J_3 \boldsymbol{I} = \tfrac{1}{2}\alpha(\boldsymbol{I} \otimes \boldsymbol{A} + \boldsymbol{A} \otimes \boldsymbol{I}),$$

which is based on the (major) symmetry of $\mathbb{C}$.

In order to identify the parameters $(\lambda, \mu, \alpha, \beta)$ in the above model with the more common engineering moduli we will consider the special case where the preferred direction is aligned with the y-axis. In this case we have $\hat{\boldsymbol{a}} = (0, 1)$. It therefore follows that

$$\boldsymbol{A} = \hat{\boldsymbol{a}} \otimes \hat{\boldsymbol{a}} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \begin{pmatrix} 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}$$

and

$$\boldsymbol{I} \otimes \boldsymbol{A} = \begin{pmatrix} \boldsymbol{A} & 0 \\ 0 & \boldsymbol{A} \end{pmatrix}, \quad \boldsymbol{A} \otimes \boldsymbol{I} = \begin{pmatrix} 0 & 0 \\ 0 & \boldsymbol{I} \end{pmatrix}, \quad \boldsymbol{A} \otimes \boldsymbol{A} = \begin{pmatrix} 0 & 0 \\ 0 & \boldsymbol{A} \end{pmatrix}$$

From this we get :

$$\begin{aligned}\mathbb{C}_{\text{ortho}} &= \alpha\left(\begin{pmatrix}\boldsymbol{A} & 0\\ 0 & \boldsymbol{A}\end{pmatrix}+\begin{pmatrix}0 & 0\\ 0 & \boldsymbol{I}\end{pmatrix}\right)+2\beta\begin{pmatrix}0 & 0\\ 0 & \boldsymbol{A}\end{pmatrix}\\ &= \begin{pmatrix}\alpha\boldsymbol{A} & 0\\ 0 & (\alpha+2\beta)\boldsymbol{A}+\alpha\boldsymbol{I}\end{pmatrix}\end{aligned}$$

Based on this we can write the full elasticity tensor as follows :

$$\mathbb{C}^{ijkl} = \begin{pmatrix} \begin{pmatrix}2\mu+\lambda & 0\\ {\scriptstyle 0000} & {\scriptstyle 0100}\\ 0 & \lambda+\alpha\\ {\scriptstyle 1000} & {\scriptstyle 1100}\end{pmatrix} & \begin{pmatrix}0 & \mu\\ {\scriptstyle 0001} & {\scriptstyle 0101}\\ \mu & 0\\ {\scriptstyle 1001} & {\scriptstyle 1101}\end{pmatrix}\\ \begin{pmatrix}0 & \mu\\ {\scriptstyle 0010} & {\scriptstyle 0110}\\ \mu & 0\\ {\scriptstyle 1010} & {\scriptstyle 1110}\end{pmatrix} & \begin{pmatrix}\lambda+\alpha & 0\\ {\scriptstyle 0011} & {\scriptstyle 0111}\\ 0 & 2\mu+\lambda+2\alpha+2\beta\\ {\scriptstyle 1011} & {\scriptstyle 1111}\end{pmatrix}\end{pmatrix}$$

In Voigt notation this becomes :

$$\tilde{C} = \begin{pmatrix}2\mu+\lambda & \lambda+\alpha & 0\\ \lambda+\alpha & 2\mu+\lambda+2\alpha+2\beta & 0\\ 0 & 0 & 2\mu\end{pmatrix}$$

The compliance matrix written in terms of the engineering moduli has the following form :

$$\tilde{D} = \begin{pmatrix}1/Y_x & -\nu_{yx}/Y_y & 0\\ -\nu_{xy}/Y_x & 1/Y_y & 0\\ 0 & 0 & 1/G\end{pmatrix}$$

Here, Y_x and Y_y are Young's modulus along the two orthogonal directions and G is the shear modulus. ν_{xy} and ν_{yx} are the Poisson ratios (ν_{xy} is the Poisson's ratio that corresponds to a contraction in direction y when an extension is applied in direction x). Due to symmetry, the Poisson ratios are constrained such that $\nu_{xy}/Y_x = \nu_{yx}/Y_y$.

If we let $Y = Y_x$, $\nu = \nu_{xy}$ and $\eta = Y_y/Y_x$ then it follows that we have :

$$\eta = \frac{Y_y}{Y_x} = \frac{\nu_{yx}}{\nu_{xy}} = \frac{Y_y}{Y} = \frac{\nu_{yx}}{\nu}$$

and we can then write

$$\tilde{D} = \begin{pmatrix}1/Y & -\nu/Y & 0\\ -\nu/Y & 1/(\eta Y) & 0\\ 0 & 0 & 1/G\end{pmatrix}$$

If we invert this matrix we get :

$$\tilde{C} = \begin{pmatrix} \frac{Y}{1-\eta\nu^2} & \frac{\eta\nu Y}{1-\eta\nu^2} & 0 \\ \frac{\eta\nu Y}{1-\eta\nu^2} & \frac{\eta Y}{1-\eta\nu^2} & 0 \\ 0 & 0 & G \end{pmatrix} \tag{4.3}$$

Equating each of the non-zero elements in $\tilde{C}$ and solving for λ, μ, α and β gives :

$$\begin{aligned} \lambda &= \frac{Y}{1-\eta\nu^2} - G \\ \mu &= \tfrac{1}{2}G \\ \alpha &= \frac{\eta\nu - 1}{1-\eta\nu^2}Y + G \\ \beta &= \frac{1+\eta-2\eta\nu}{2(1-\eta\nu^2)}Y - G \end{aligned} \tag{4.4}$$

For an isotropic material we have $\eta = 1$ and $G = 2\mu = \frac{Y}{1+\nu}$ and if we insert this above we do in fact recover the expressions for λ and μ from Eq. (4.2) as well as $\alpha = \beta = 0$.

Using the result from Wu et al. [2003] we can write the effective stiffness for an arbitrary direction relative to the preferred directions. Let θ denote the angle relative to the x-axis. The stiffness is then given by :

$$\frac{1}{Y_\theta} = \frac{1}{Y_x}\cos^4\theta + \left(\frac{1}{G} - \frac{2\nu}{Y_x}\right)\cos^2\theta\sin^2\theta + \frac{1}{Y_y}\sin^4\theta$$

It should be noted however that unlike in Wu et al. [2003] the orthotropic model derived here only has four parameters (plus the implied choice of preferred direction which is common in both). Based on the careful analysis in He and Zheng [1996] of symmetry groups and the associated classes of hyperelastic tensors we know that there can only be four *independent* parameters (plus a direction) for a 2D orthotropic material. Any additional parameters must necessarily be conflated with existing parameters.

4.4 Natural orthotropic invariants

Başar et al. [2000] also develops an orthotropic St. Venant-Kirchhoff model, but they write the energy in terms of two *oriented strain tensors*. Their formulation is for 3D materials, but is easily simplified to 2D materials. Ultimately this formulation is equivalent to the one presented above, but provides a

more natural interpretation of the material parameters. It also leads to a nice generalization as shown in Itskov [2001]. In this section we will show that the models are in fact identical, which is not immediately obvious.

As a foundation they start out with *two* structural tensors (in 2D) :

$$\boldsymbol{L}_1 = \hat{\boldsymbol{a}}_1 \otimes \hat{\boldsymbol{a}}_1, \quad \boldsymbol{L}_2 = \hat{\boldsymbol{a}}_2 \otimes \hat{\boldsymbol{a}}_2$$

where $\hat{\boldsymbol{a}}_1$ and $\hat{\boldsymbol{a}}_2$ are normalized vectors in the two orthogonal directions of orthotropy. Let now the directional strain tensors be defined as

$$\boldsymbol{E}_1 \equiv \boldsymbol{E}\boldsymbol{L}_1, \quad \boldsymbol{E}_2 \equiv \boldsymbol{E}\boldsymbol{L}_2$$

Based on this they consider the following invariants

$$L_1 = \mathrm{tr}(\boldsymbol{E}_1), \quad L_2 = \mathrm{tr}(\boldsymbol{E}_2), \quad L_3 = \mathrm{tr}(\boldsymbol{E}_1\boldsymbol{E}_2)$$

Using these invariants, Başar et al. [2000] and Itskov [2001] write the energy density for a St. Venant-Kirchhoff material in a nicely symmetric way :

$$\psi_m = \tfrac{1}{2} \sum_{1 \le i,j \le 2} a_{ij} L_i L_j + G_{12} L_3 \tag{4.5}$$

where a_{11}, a_{22}, $a_{12} = a_{21}$, and G_{12} are the material parameters. By comparison in Section 4.3 we wrote the energy density as :

$$\psi_m = \frac{\lambda}{2} J_1^2 + \mu J_2 + \alpha J_1 J_3 + \beta J_3^2$$

To show that the two formulations are equivalent, we will show that J_1^2, J_2, $J_1 J_3$, and J_3^2 can all be expressed in terms of L_1, L_2 and L_3. To this end let us define the following orthogonal matrix :

$$\boldsymbol{R} \equiv \begin{bmatrix} \hat{\boldsymbol{a}}_1 & \hat{\boldsymbol{a}}_2 \end{bmatrix} \in \mathbb{R}^{2\times 2}$$

It's then easy to show that

$$\boldsymbol{L}_1 = \boldsymbol{R} \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \boldsymbol{R}^T, \quad \boldsymbol{L}_2 = \boldsymbol{R} \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \boldsymbol{R}^T$$

If we define the rotated strain tensor as $\tilde{\boldsymbol{E}} \equiv \boldsymbol{R}^T \boldsymbol{E} \boldsymbol{R}$ then it follows that $\boldsymbol{E} = \boldsymbol{R}\tilde{\boldsymbol{E}}\boldsymbol{R}^T$. Since the two are related by a similarity transformation, then it also follows that their eigenvalues are identical and therefore they have the same trace. Inserting these definitions we get :

$$L_1 = \mathrm{tr}(\boldsymbol{E}_1) = \mathrm{tr}(\boldsymbol{E}\boldsymbol{L}_1) = \mathrm{tr}\left(\boldsymbol{R}\tilde{\boldsymbol{E}}\boldsymbol{R}^T \boldsymbol{R} \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \boldsymbol{R}^T \right) = \tilde{e}_{11}$$

where $\tilde{e}_{ij}$ represents the (i,j) component of $\tilde{\boldsymbol{E}}$. The identity $L_2 = \tilde{e}_{22}$ follows in an identical fashion. To evaluate L_3 we note that :

$$\hat{\boldsymbol{a}}_1^T \boldsymbol{E} \hat{\boldsymbol{a}}_2 = \hat{\boldsymbol{a}}_1^T \boldsymbol{R} \tilde{\boldsymbol{E}} \boldsymbol{R}^T \hat{\boldsymbol{a}}_2 = \hat{\boldsymbol{a}}_1^T \begin{bmatrix} \hat{\boldsymbol{a}}_1 & \hat{\boldsymbol{a}}_2 \end{bmatrix} \tilde{\boldsymbol{E}} \begin{bmatrix} \hat{\boldsymbol{a}}_1^T \\ \hat{\boldsymbol{a}}_2^T \end{bmatrix} \hat{\boldsymbol{a}}_2 = \begin{bmatrix} 1 & 0 \end{bmatrix} \tilde{\boldsymbol{E}} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \tilde{e}_{12}$$

From this it follows that :

$$\begin{aligned} L_3 &= \operatorname{tr}(\boldsymbol{E}_1 \boldsymbol{E}_2) \\ &= \operatorname{tr}(\boldsymbol{E} \hat{\boldsymbol{a}}_1 \hat{\boldsymbol{a}}_1^T \boldsymbol{E} \hat{\boldsymbol{a}}_2 \hat{\boldsymbol{a}}_2^T) \\ &= \tilde{e}_{12} \operatorname{tr}(\boldsymbol{E} \hat{\boldsymbol{a}}_1 \hat{\boldsymbol{a}}_2^T) \\ &= \tilde{e}_{12} \operatorname{tr}(\hat{\boldsymbol{a}}_2^T \boldsymbol{E} \hat{\boldsymbol{a}}_1) \\ &= \tilde{e}_{12}^2 \end{aligned}$$

Using the definitions from Section 3.7 we now see that

$$\begin{aligned} J_1^2 &= \operatorname{tr}^2(\boldsymbol{E}) = \operatorname{tr}^2(\tilde{\boldsymbol{E}}) = (L_1 + L_2)^2 \\ J_2 &= \operatorname{tr}(\boldsymbol{E}^2) = \operatorname{tr}(\tilde{\boldsymbol{E}}^2) = L_1^2 + L_2^2 + 2L_3 \\ J_3^2 &= \operatorname{tr}^2(\boldsymbol{E}\boldsymbol{L}_1) = L_1^2 \\ J_1 J_3 &= \operatorname{tr}(\boldsymbol{E}) \operatorname{tr}(\boldsymbol{E}\boldsymbol{L}_1) = (L_1 + L_2) L_1 \end{aligned}$$

As intermediate variables let :

$$\tilde{L}_1 \equiv L_1^2, \quad \tilde{L}_2 \equiv L_2^2, \quad \tilde{L}_3 \equiv L_3, \quad \tilde{L}_4 \equiv L_1 L_2,$$

We can then write the above results as :

$$\begin{aligned} J_1^2 &= \tilde{L}_1 + \tilde{L}_2 + 2\tilde{L}_4 \\ J_2 &= \tilde{L}_1 + \tilde{L}_2 + 2\tilde{L}_3 \\ J_3^2 &= \tilde{L}_1 \\ J_1 J_3 &= \tilde{L}_1 + \tilde{L}_4 \end{aligned}$$

Since

$$\begin{aligned} \psi_m &= \frac{\lambda}{2} J_1^2 + \mu J_2 + \alpha J_1 J_3 + \beta J_3^2 \\ &= \frac{\lambda}{2} (\tilde{L}_1 + \tilde{L}_2 + 2\tilde{L}_4) + \mu (\tilde{L}_1 + \tilde{L}_2 + 2\tilde{L}_3) + \alpha (\tilde{L}_1 + \tilde{L}_4) + \beta \tilde{L}_1 \\ &= (\tfrac{1}{2}\lambda + \mu + \alpha + \beta) \tilde{L}_1 + (\tfrac{1}{2}\lambda + \mu) \tilde{L}_2 + 2\mu \tilde{L}_3 + (\lambda + \alpha) \tilde{L}_4 \\ &= \tfrac{1}{2} a_{11} \tilde{L}_1 + \tfrac{1}{2} a_{22} \tilde{L}_2 + a_{12} \tilde{L}_4 + G_{12} \tilde{L}_3 \end{aligned}$$

it follows that the model established in the previous section and the model in Eq. (4.5) are identical if :

$$\begin{aligned} \tfrac{1}{2}\lambda + \mu + \alpha + \beta &= \tfrac{1}{2} a_{11} \\ \tfrac{1}{2}\lambda + \mu &= \tfrac{1}{2} a_{22} \\ \lambda + \alpha &= a_{12} \\ 2\mu &= G_{12} \end{aligned}$$

By inserting the values from Eq. (4.4) and simplifying we get

$$\begin{aligned} a_{11} &= \frac{\eta Y}{1-\eta\nu^2} \\ a_{22} &= \frac{Y}{1-\eta\nu^2} \\ a_{12} &= \frac{\eta\nu Y}{1-\eta\nu^2} \\ G_{12} &= G \end{aligned}$$

Notably these are exactly the values in Eq. (4.3). If we define $\boldsymbol{\epsilon} \equiv (L_2, L_1, \sqrt{L_3})^T$, then we can therefore write Eq. (4.5) as :

$$\psi_m = \tfrac{1}{2}\boldsymbol{\epsilon}^T \begin{pmatrix} a_{22} & a_{12} & 0 \\ a_{12} & a_{11} & 0 \\ 0 & 0 & G_{12} \end{pmatrix} \boldsymbol{\epsilon} = \tfrac{1}{2}\boldsymbol{\epsilon}^T \tilde{C} \boldsymbol{\epsilon}$$

This is exactly the same expression as for linear elasticity except that $\boldsymbol{\epsilon}$ has been replaced by $\boldsymbol{\epsilon}$. Thus $\boldsymbol{\epsilon}$ compactly encodes all the information about the geometric nonlinearity in the system. It can be argued that this way of writing the orthotropic St. Venant-Kirchhoff model in some sense provides a more natural parameterization of the model because it literally uses the same coefficients as linear elasticity.

The fundamental difference in the derivation of this model is the choice of invariants which is different than the choice used in Section 4.3. However, this just goes to show that even a minimal integrity basis is not unique. In the following we will do our computations based on J_1, J_2, and J_3 since this is computationally convenient, but at this point it should be clear that this is equivalent to using the model by Başar et al. [2000].

4.5 Constant strain discretization

Given the constitutive model for an orthotropic St. Venant-Kirchhoff material this section will focus on how to compute the corresponding forces and force Jacobians using a constant strain triangle discretization.

4.5.1 Representation of the strain tensor

As stated earlier the Green-Lagrange strain tensor is given by $\boldsymbol{E} = \frac{1}{2}(\boldsymbol{F}^T\boldsymbol{F} - \boldsymbol{I})$ where $\boldsymbol{F}$ is the deformation gradient. Since the material space is inherently two dimensional it follows that $\boldsymbol{E}$ can be represented by a symmetric

2×2 tensor. I.e. it has three degrees of freedom. These may be embedded inside a much larger tensor, but the additional entries in the tensor represent a null space. In the following we shall take advantage of that because we would like to use the edges of a triangle in 3D to compute the strain within the plane of the triangle.

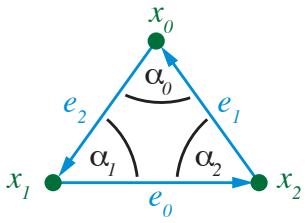


Figure 4.1: *The labeling scheme used for vertices, edges, and interior angles of a single triangle.*

In the following we will use the numbering convention shown in Fig. 4.1. Let l_i be the deformed length of edge i, $\bar{l}_i$ the undeformed length, and $s_i = \frac{1}{2}(l_i^2 - \bar{l}_i^2)$. By definition we have

$$s_i = \bar{\boldsymbol{e}}_i^T \boldsymbol{E} \bar{\boldsymbol{e}}_i \tag{4.6}$$

where $\bar{\boldsymbol{e}}_i$ is the undeformed edge in the material domain. Since $\boldsymbol{E}$ is symmetric we can write it as $\boldsymbol{E} = \sum_i \omega_i \boldsymbol{m}_i \otimes \boldsymbol{m}_i$, where $\boldsymbol{m}_i$ is the inplane normal for edge $\bar{\boldsymbol{e}}_i$ (see Section 4.1). Note that $\boldsymbol{m}_i$ refers to the undeformed configuration and it only serves to define a basis for the strain tensor, while all the information about the actual deformation is encoded in ω_i. Note also that the construction holds even if the undeformed triangle is embedded in a larger space like $\mathbb{R}^3$. In this case, the outer product will project away any component that is not in the tangent space of the deformation function (i.e. in the plane of the undeformed triangle).

Using the convention that (i, j, k) is a cyclic permutations of $(0, 1, 2)$ it follows that

$$s_i = \omega_j (\bar{\boldsymbol{e}}_i \cdot \boldsymbol{m}_j)^2 + \omega_k (\bar{\boldsymbol{e}}_i \cdot \boldsymbol{m}_k)^2$$

and since $\boldsymbol{m}_i$ is an 90 degree inplane rotation of $\bar{\boldsymbol{e}}_i$ we have that $\bar{\boldsymbol{e}}_i \cdot \boldsymbol{m}_j = \|\bar{\boldsymbol{e}}_i \times \bar{\boldsymbol{e}}_j\| = 2A$ where A is the area of the triangle. As we gather everything up, we therefore get :

$$\begin{bmatrix} s_0 \\ s_1 \\ s_2 \end{bmatrix} = 4A^2 \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} \omega_0 \\ \omega_1 \\ \omega_2 \end{bmatrix}$$

Solving these equations wrt. ω_i gives :

$$\omega_i = \frac{1}{8A^2}(s_j + s_k - s_i)$$

and therefore

$$\boldsymbol{E} = \frac{1}{8A^2}\sum_i (s_j + s_k - s_i)(\boldsymbol{m}_i \otimes \boldsymbol{m}_i)$$

Using that $\boldsymbol{m}_i + \boldsymbol{m}_j + \boldsymbol{m}_k = 0$ we can rewrite this as

$$\begin{aligned}
\boldsymbol{E} &= \frac{1}{8A^2}\sum_i (s_j + s_k - s_i)(\boldsymbol{m}_i \otimes (-\boldsymbol{m}_j - \boldsymbol{m}_k)) \\
&= \frac{1}{8A^2}\sum_i (-s_j - s_k + s_i)(\boldsymbol{m}_i \otimes \boldsymbol{m}_j) + (-s_j - s_k + s_i)(\boldsymbol{m}_i \otimes \boldsymbol{m}_k) \\
&= \frac{1}{8A^2}\sum_i (-s_k - s_i + s_j)(\boldsymbol{m}_j \otimes \boldsymbol{m}_k) + (-s_i - s_j + s_k)(\boldsymbol{m}_k \otimes \boldsymbol{m}_j) \\
&= -\frac{1}{8A^2}\sum_i s_i(\boldsymbol{m}_j \otimes \boldsymbol{m}_k + \boldsymbol{m}_k \otimes \boldsymbol{m}_j) - (s_j - s_k)(\boldsymbol{m}_j \otimes \boldsymbol{m}_k - \boldsymbol{m}_k \otimes \boldsymbol{m}_j)
\end{aligned}$$

The third equality is based on the fact that we are summing over all i, so we can permute the indices cyclically. To simplify the last expression further, let $d_{ij} = \boldsymbol{m}_i \otimes \boldsymbol{m}_j - \boldsymbol{m}_j \otimes \boldsymbol{m}_i$. It then follows that

$$\begin{aligned}
d_{ij} - d_{ki} &= \boldsymbol{m}_i \otimes \boldsymbol{m}_j - \boldsymbol{m}_j \otimes \boldsymbol{m}_i - \boldsymbol{m}_k \otimes \boldsymbol{m}_i + \boldsymbol{m}_i \otimes \boldsymbol{m}_k \\
&= \boldsymbol{m}_i \otimes (\boldsymbol{m}_j + \boldsymbol{m}_k) - (\boldsymbol{m}_j + \boldsymbol{m}_k) \otimes \boldsymbol{m}_i \\
&= -\boldsymbol{m}_i \otimes \boldsymbol{m}_i + \boldsymbol{m}_i \otimes \boldsymbol{m}_i \\
&= 0
\end{aligned}$$

As a consequence we have

$$\begin{aligned}
\sum_i (s_j - s_k) d_{jk} &= (s_1 - s_2)d_{12} + (s_2 - s_0)d_{20} + (s_0 - s_1)d_{01} \\
&= s_0(d_{01} - d_{20}) + s_1(d_{12} - d_{01}) + s_2(d_{20} - d_{12}) \\
&= 0
\end{aligned}$$

and therefore

$$\boldsymbol{E} = -\frac{1}{8A^2}\sum_i s_i(\boldsymbol{m}_j \otimes \boldsymbol{m}_k + \boldsymbol{m}_k \otimes \boldsymbol{m}_j)$$

4.5.2 Evaluation of St. Venant-Kirchhoff energy

To compute an explicit representation for the membrane energy we need to compute $\mathrm{tr}^2(\boldsymbol{E})$, $\mathrm{tr}(\boldsymbol{E}^2)$ and $\mathrm{tr}(\boldsymbol{AEA})$. For these computations it will be convenient to introduce (l, m, n) as another cyclic permutation of $(1, 2, 3)$. With this we get :

$$\mathrm{tr}(\boldsymbol{E}) = -\frac{1}{4A^2}\sum_i s_i(\boldsymbol{m}_j \cdot \boldsymbol{m}_k)$$

and therefore

$$\mathrm{tr}^2(\boldsymbol{E}) = \frac{1}{16A^4}\sum_{i,l} s_i s_l (\boldsymbol{m}_j \cdot \boldsymbol{m}_k)(\boldsymbol{m}_m \cdot \boldsymbol{m}_n)$$

Using the notation $S(\boldsymbol{A}) = \boldsymbol{A} + \boldsymbol{A}^T$ for the symmetric part of a matrix, we furthermore get :

$$\boldsymbol{E}^2 = \frac{1}{64A^4}\sum_{i,l} s_i s_l S(\boldsymbol{m}_j \otimes \boldsymbol{m}_k) S(\boldsymbol{m}_m \otimes \boldsymbol{m}_n)$$

Since

$$(\boldsymbol{m}_j \otimes \boldsymbol{m}_k)(\boldsymbol{m}_m \otimes \boldsymbol{m}_n) = \boldsymbol{m}_j \boldsymbol{m}_k^T \boldsymbol{m}_m \boldsymbol{m}_n^T = (\boldsymbol{m}_k \cdot \boldsymbol{m}_m)(\boldsymbol{m}_j \otimes \boldsymbol{m}_n)$$

it follows that

$$\begin{aligned} S(\boldsymbol{m}_j \otimes \boldsymbol{m}_k) S(\boldsymbol{m}_m \otimes \boldsymbol{m}_n) =& (\boldsymbol{m}_k \cdot \boldsymbol{m}_m)(\boldsymbol{m}_j \otimes \boldsymbol{m}_n) + (\boldsymbol{m}_j \cdot \boldsymbol{m}_m)(\boldsymbol{m}_k \otimes \boldsymbol{m}_n) \\ &+ (\boldsymbol{m}_k \cdot \boldsymbol{m}_n)(\boldsymbol{m}_j \otimes \boldsymbol{m}_m) + (\boldsymbol{m}_j \cdot \boldsymbol{m}_n)(\boldsymbol{m}_k \otimes \boldsymbol{m}_m) \end{aligned}$$

and therefore

$$\begin{aligned} \mathrm{tr}\left(S(\boldsymbol{m}_j \otimes \boldsymbol{m}_k) S(\boldsymbol{m}_m \otimes \boldsymbol{m}_n)\right) =& (\boldsymbol{m}_k \cdot \boldsymbol{m}_m)(\boldsymbol{m}_j \cdot \boldsymbol{m}_n) + (\boldsymbol{m}_j \cdot \boldsymbol{m}_m)(\boldsymbol{m}_k \cdot \boldsymbol{m}_n) \\ &+ (\boldsymbol{m}_k \cdot \boldsymbol{m}_n)(\boldsymbol{m}_j \cdot \boldsymbol{m}_m) + (\boldsymbol{m}_j \cdot \boldsymbol{m}_n)(\boldsymbol{m}_k \cdot \boldsymbol{m}_m) \\ =& 2(\boldsymbol{m}_k \cdot \boldsymbol{m}_m)(\boldsymbol{m}_j \cdot \boldsymbol{m}_n) + 2(\boldsymbol{m}_k \cdot \boldsymbol{m}_n)(\boldsymbol{m}_j \cdot \boldsymbol{m}_m) \end{aligned}$$

Consequently we have

$$\mathrm{tr}(\boldsymbol{E}^2) = \frac{1}{32A^4}\sum_{i,l} s_i s_l \left((\boldsymbol{m}_k \cdot \boldsymbol{m}_m)(\boldsymbol{m}_j \cdot \boldsymbol{m}_n) + (\boldsymbol{m}_k \cdot \boldsymbol{m}_n)(\boldsymbol{m}_j \cdot \boldsymbol{m}_m)\right)$$

For the orthotropic component we have :

$$\begin{aligned} \boldsymbol{AEA} &= -\frac{1}{8A^2}\sum_i s_i \hat{\boldsymbol{a}}\hat{\boldsymbol{a}}^T S(\boldsymbol{m}_j \otimes \boldsymbol{m}_k)\hat{\boldsymbol{a}}\hat{\boldsymbol{a}}^T \\ &= -\frac{1}{8A^2}\sum_i s_i \hat{\boldsymbol{a}}\hat{\boldsymbol{a}}^T (\boldsymbol{m}_j \boldsymbol{m}_k^T + \boldsymbol{m}_k \boldsymbol{m}_j^T)\hat{\boldsymbol{a}}\hat{\boldsymbol{a}}^T \\ &= -\frac{2}{8A^2}\sum_i s_i \hat{\boldsymbol{a}}(\hat{\boldsymbol{a}} \cdot \boldsymbol{m}_j)(\boldsymbol{m}_k \cdot \hat{\boldsymbol{a}})\hat{\boldsymbol{a}}^T \\ &= -\frac{2}{8A^2}\sum_i s_i (\hat{\boldsymbol{a}} \cdot \boldsymbol{m}_j)(\hat{\boldsymbol{a}} \cdot \boldsymbol{m}_k)\hat{\boldsymbol{a}} \otimes \hat{\boldsymbol{a}} \end{aligned}$$

and since $\mathrm{tr}(\boldsymbol{A}) = \hat{\boldsymbol{a}} \cdot \hat{\boldsymbol{a}} = 1$ it follows that :

$$\mathrm{tr}(\boldsymbol{AEA}) = -\frac{1}{4A^2}\sum_i s_i (\hat{\boldsymbol{a}} \cdot \boldsymbol{m}_j)(\hat{\boldsymbol{a}} \cdot \boldsymbol{m}_k)$$

It should be noted here that $\hat{\boldsymbol{a}}$ is a vector in one of the two preferred directions *within* the plane of the triangle. As such it is now a 3 dimensional vector which means that the dot products make sense. Using the above result we get :

$$\mathrm{tr}^2(\boldsymbol{AEA}) = \frac{1}{16A^4}\sum_{i,l} s_i s_l (\hat{\boldsymbol{a}}\cdot\boldsymbol{m}_j)(\hat{\boldsymbol{a}}\cdot\boldsymbol{m}_k)(\hat{\boldsymbol{a}}\cdot\boldsymbol{m}_m)(\hat{\boldsymbol{a}}\cdot\boldsymbol{m}_n)$$

In order to evaluate $J_1 J_3$ we also need :

$$\mathrm{tr}(\boldsymbol{E})\,\mathrm{tr}(\boldsymbol{AEA}) = \frac{1}{16A^4}\sum_{i,l} s_i s_l (\boldsymbol{m}_j\cdot\boldsymbol{m}_k)(\hat{\boldsymbol{a}}\cdot\boldsymbol{m}_m)(\hat{\boldsymbol{a}}\cdot\boldsymbol{m}_n)$$

Since $J_1 J_3$ is a scalar we have $J_1 J_3 = (J_1 J_3)^T$, so we can symmetrize this expression :

$$\begin{aligned}\mathrm{tr}(\boldsymbol{E})\,\mathrm{tr}(\boldsymbol{AEA}) = \frac{1}{32A^4}\sum_{i,l} s_i s_l \big((\boldsymbol{m}_j\cdot\boldsymbol{m}_k)(\hat{\boldsymbol{a}}\cdot\boldsymbol{m}_m)(\hat{\boldsymbol{a}}\cdot\boldsymbol{m}_n)\\ + (\boldsymbol{m}_m\cdot\boldsymbol{m}_n)(\hat{\boldsymbol{a}}\cdot\boldsymbol{m}_j)(\hat{\boldsymbol{a}}\cdot\boldsymbol{m}_k)\big)\end{aligned}$$

Let us now define a vector $\boldsymbol{s} = [s_0, s_1, s_2]^T$ as well as four symmetric matrices $T^{(q)}$:

$$\begin{aligned}T^{(1)}_{il} &= (\boldsymbol{m}_j\cdot\boldsymbol{m}_k)(\boldsymbol{m}_m\cdot\boldsymbol{m}_n)\\ T^{(2)}_{il} &= (\boldsymbol{m}_k\cdot\boldsymbol{m}_m)(\boldsymbol{m}_j\cdot\boldsymbol{m}_n) + (\boldsymbol{m}_k\cdot\boldsymbol{m}_n)(\boldsymbol{m}_j\cdot\boldsymbol{m}_m)\\ T^{(3)}_{il} &= (\hat{\boldsymbol{a}}\cdot\boldsymbol{m}_j)(\hat{\boldsymbol{a}}\cdot\boldsymbol{m}_k)(\hat{\boldsymbol{a}}\cdot\boldsymbol{m}_m)(\hat{\boldsymbol{a}}\cdot\boldsymbol{m}_n)\\ T^{(4)}_{il} &= (\boldsymbol{m}_j\cdot\boldsymbol{m}_k)(\hat{\boldsymbol{a}}\cdot\boldsymbol{m}_m)(\hat{\boldsymbol{a}}\cdot\boldsymbol{m}_n) + (\boldsymbol{m}_m\cdot\boldsymbol{m}_n)(\hat{\boldsymbol{a}}\cdot\boldsymbol{m}_j)(\hat{\boldsymbol{a}}\cdot\boldsymbol{m}_k)\end{aligned} \tag{4.7}$$

The results can then be written succintly as

$$\begin{aligned}J_1^2 = \mathrm{tr}^2(\boldsymbol{E}) &= \frac{1}{16A^4}\sum_{i,l} s_i s_l T^{(1)}_{il} = \frac{1}{16A^4}\boldsymbol{s}^T T^{(1)}\boldsymbol{s}\\ J_2 = \mathrm{tr}(\boldsymbol{E}^2) &= \frac{1}{32A^4}\sum_{i,l} s_i s_l T^{(2)}_{il} = \frac{1}{32A^4}\boldsymbol{s}^T T^{(2)}\boldsymbol{s}\\ J_3^2 = \mathrm{tr}^2(\boldsymbol{AEA}) &= \frac{1}{16A^4}\sum_{i,l} s_i s_l T^{(3)}_{il} = \frac{1}{16A^4}\boldsymbol{s}^T T^{(3)}\boldsymbol{s}\\ J_1 J_3 = \mathrm{tr}(\boldsymbol{E})\,\mathrm{tr}(\boldsymbol{AEA}) &= \frac{1}{32A^4}\sum_{i,l} s_i s_l T^{(4)}_{il} = \frac{1}{32A^4}\boldsymbol{s}^T T^{(4)}\boldsymbol{s}\end{aligned}$$

Since $\boldsymbol{m}_i$ is just an inplane rotation of $\boldsymbol{e}_i$ it is easy to see that $\boldsymbol{m}_i\cdot\boldsymbol{m}_l = \boldsymbol{e}_i\cdot\boldsymbol{e}_l$, so none of the edge normals are actually required for the isotropic case. For

the orthotropic case we have $\boldsymbol{m}_i \cdot \hat{\boldsymbol{a}} = \boldsymbol{e}_i \cdot \hat{\boldsymbol{a}}_\perp$ where $\hat{\boldsymbol{a}}_\perp = \boldsymbol{n} \times \hat{\boldsymbol{a}}$ is a 90 degree inplane rotation of $\hat{\boldsymbol{a}}$. Whether we use $\boldsymbol{n} \times \hat{\boldsymbol{a}}$ or $\hat{\boldsymbol{a}} \times \boldsymbol{n} = -\boldsymbol{n} \times \hat{\boldsymbol{a}}$ is immaterial since we always have a product of an even number of terms with $\hat{\boldsymbol{a}}$.

It should be noted that $\boldsymbol{m}_i$ and $\hat{\boldsymbol{a}}$ refer to the undeformed configuration (material space), so all the T matrices can be precomputed. A also refers to the area of the triangle in the rest configuration. However, $T^{(q)}$ are only matrices and not tensors since they do not transform as tensors.

By making all of the above transformations we now have a simple quadratic expression for the energy density as a function of the edge elongations. To get the total energy per element we integrate the density across the element. Since the strain is assumed to be constant across the element, the energy density will also be constant, so the integration amounts to multiplication by the volume, which is Ah, where h is the thickness of the membrane :

$$W_{mem} = Ah\psi_m = \frac{h}{32A^3}\boldsymbol{s}^T\left(\lambda T^{(1)} + \mu T^{(2)} + 2\beta T^{(3)} + \alpha T^{(4)}\right)\boldsymbol{s} \tag{4.8}$$

For the isotropic case where $\alpha = \beta = 0$ this simplifies to :

$$W_{mem} = \frac{1}{32A^3}\frac{Yh}{(1-\nu^2)}\boldsymbol{s}^T\left(\nu T^{(1)} + \tfrac{1}{2}(1-\nu)T^{(2)}\right)\boldsymbol{s}$$

For convenience in subsequent computations we gather all the constant terms in a single matrix and write this as :

$$W_{mem} = \boldsymbol{s}^T T \boldsymbol{s} = \boldsymbol{s}^T \begin{bmatrix} p_0 & q_2 & q_1 \\ q_2 & p_1 & q_0 \\ q_1 & q_0 & p_2 \end{bmatrix} \boldsymbol{s} = \sum_i p_i s_i^2 + 2\sum_i q_i s_j s_k$$

4.5.3 Evaluation of other orthotropic energy functions

It should be noted that *all* isotropic and orthotropic hyperelastic energies must be functions of the invariants derived in Section 3.7. For a constant strain triangle discretization all such energies can therefore be evaluated using the results in the previous section. Not all of them will lead to simple quadratic forms, but they can all be computed as functions of the edge lengths.

4.5.4 Membrane force evaluation

The membrane force are given by $-\nabla W_{mem}$ where the gradient is with respect to the vertex positions. To compute it directly for the St. Venant-

Kirchhoff membrane directly we introduce :

$$[\varepsilon_{ij}] = \begin{bmatrix} 0 & -1 & 1 \\ 1 & 0 & -1 \\ -1 & 1 & 0 \end{bmatrix}$$

Note that for $k \neq i \neq j$ we have that ε_{ij} is 1 if (i, j, k) is an odd permutation of $(0, 1, 2)$ while it is -1 if (i, j, k) is an even permutation. This is opposite the usual convention for the (Levi-Civita) permutation tensor, so we could flip the sign to be consistent with that convention. However, this will introduce a lot of extra minus signs to keep track of, so we will use the definition stated above. With this convention we can write :

$$\boldsymbol{e}_i = \sum_l \varepsilon_{il} \boldsymbol{x}_l$$

and thus

$$\frac{\partial \boldsymbol{e}_i}{\partial \boldsymbol{x}_m} = \varepsilon_{im} \boldsymbol{I}$$

where $\boldsymbol{e}_i$ is a deformed edge vector. Since $s_i = \frac{1}{2}(l_i^2 - L_i^2)$ it follows that :

$$\begin{aligned} \frac{\partial s_i}{\partial \boldsymbol{x}_m} &= l_i \frac{\partial l_i}{\partial \boldsymbol{x}_m} = l_i \frac{\partial \sqrt{\boldsymbol{e}_i \cdot \boldsymbol{e}_i}}{\partial \boldsymbol{x}_m} \\ &= l_i \frac{1}{2\sqrt{\boldsymbol{e}_i \cdot \boldsymbol{e}_i}} (\boldsymbol{e}_i^T \nabla_{\boldsymbol{x}_m} \boldsymbol{e}_i + \boldsymbol{e}_i^T \nabla_{\boldsymbol{x}_m} \boldsymbol{e}_i) \\ &= l_i \frac{1}{2l_i} 2\boldsymbol{e}_i^T \varepsilon_{im} \boldsymbol{I} \\ &= \varepsilon_{im} \boldsymbol{e}_i^T \end{aligned}$$

and therefore

$$\frac{\partial s_i^2}{\partial \boldsymbol{x}_m} = 2\varepsilon_{im} s_i \boldsymbol{e}_i^T$$

From this it furthermore follows that

$$\frac{\partial}{\partial \boldsymbol{x}_m} \sum_i p_i s_i^2 = 2 \sum_i \varepsilon_{im} p_i s_i \boldsymbol{e}_i^T$$

which gives us :

$$\frac{\partial}{\partial \boldsymbol{x}_m} \sum_i p_i s_i^2 = 2 p_n s_n \boldsymbol{e}_n^T - 2 p_l s_l \boldsymbol{e}_l^T$$

For the second term of the energy we get

$$\frac{\partial}{\partial \boldsymbol{x}_m} (q_i s_j s_k) = q_i \left(s_j \frac{\partial s_k}{\partial \boldsymbol{x}_m} + s_k \frac{\partial s_j}{\partial \boldsymbol{x}_m} \right) = q_i \left(\varepsilon_{km} s_j \boldsymbol{e}_k^T + \varepsilon_{jm} s_k \boldsymbol{e}_j^T \right)$$

and thus

$$\begin{aligned}\frac{\partial}{\partial \boldsymbol{x}_m}\left(\sum_i q_i s_j s_k\right) &= \sum_i q_i \left(\varepsilon_{km} s_j \boldsymbol{e}_k^T + \varepsilon_{jm} s_k \boldsymbol{e}_j^T\right) \\ &= \sum_i (q_j s_k + q_k s_j)\varepsilon_{im} \boldsymbol{e}_i^T \\ &= -(q_m s_n + q_n s_m)\boldsymbol{e}_l^T + (q_l s_m + q_m s_l)\boldsymbol{e}_n^T\end{aligned}$$

where the second equality follows by cyclic rotation of the indices. Combining all the results we get

$$\begin{aligned}\frac{\partial W_{mem}}{\partial \boldsymbol{x}_m} &= 2p_n s_n \boldsymbol{e}_n^T - 2p_l s_l \boldsymbol{e}_l^T + 2((q_l s_m + q_m s_l)\boldsymbol{e}_n^T - (q_m s_n + q_n s_m)\boldsymbol{e}_l^T) \\ &= 2(p_n s_n + q_l s_m + q_m s_l)\boldsymbol{e}_n^T - 2(p_l s_l + q_m s_n + q_n s_m)\boldsymbol{e}_l^T\end{aligned}$$

To write this more succintly, let :

$$r_l = 2(p_l s_l + q_m s_n + q_n s_m)$$

We can then write the stretch forces as :

$$\begin{aligned}f_m &= -\left(\frac{\partial W_{mem}}{\partial \boldsymbol{x}_m}\right)^T \\ &= r_l \boldsymbol{e}_l - r_n \boldsymbol{e}_n \\ &= r_l(\boldsymbol{x}_n - \boldsymbol{x}_m) - r_n(\boldsymbol{x}_m - \boldsymbol{x}_l) \\ &= r_l \boldsymbol{x}_n + r_n \boldsymbol{x}_l - (r_l + r_n)\boldsymbol{x}_m\end{aligned}$$

4.5.5 Membrane force Jacobians

For the second order derivatives and any i, m, and n, we get :

$$\begin{aligned}\frac{\partial^2 s_i^2}{\partial \boldsymbol{x}_n \boldsymbol{x}_m} &= \frac{\partial}{\partial \boldsymbol{x}_n}\left(\frac{\partial s_i^2}{\partial \boldsymbol{x}_m}\right)^T \\ &= \frac{\partial}{\partial \boldsymbol{x}_n}\left(2\varepsilon_{im} s_i \boldsymbol{e}_i^T\right)^T \\ &= 2\varepsilon_{im}\left(s_i \frac{\partial \boldsymbol{e}_i}{\partial \boldsymbol{x}_n} + \boldsymbol{e}_i \frac{\partial s_i}{\partial \boldsymbol{x}_n}\right) \\ &= 2\varepsilon_{im}\left(s_i \varepsilon_{in} \boldsymbol{I} + \boldsymbol{e}_i \varepsilon_{in} \boldsymbol{e}_i^T\right) \\ &= 2\varepsilon_{im}\varepsilon_{in}\left(s_i \boldsymbol{I} + \boldsymbol{e}_i \otimes \boldsymbol{e}_i\right)\end{aligned}$$

For $m = n$ we have $\varepsilon_{im}\varepsilon_{in} = 1 - \delta_{im}$ and for $m \neq n$ we see that $\varepsilon_{im}\varepsilon_{in}$ is only non-zero (and in fact -1) for $i \neq m \neq n$ so this leads to :

$$\frac{\partial^2}{\partial \boldsymbol{x}_n \boldsymbol{x}_m}\left(\sum_i p_i s_i^2\right) = \begin{cases} -2p_l\left(s_l \boldsymbol{I} + \boldsymbol{e}_l \otimes \boldsymbol{e}_l\right), & l \neq m \neq n \\ 2\sum_{i\neq m,n} p_i\left(s_i \boldsymbol{I} + \boldsymbol{e}_i \otimes \boldsymbol{e}_i\right), & m = n \end{cases}$$

For the other term we get for any (i, j, k), m, and n :

$$\begin{aligned} \frac{\partial^2 q_i s_j s_k}{\partial \boldsymbol{x}_n \boldsymbol{x}_m} &= q_i \frac{\partial}{\partial \boldsymbol{x}_n}\left(\frac{\partial s_j s_k}{\partial \boldsymbol{x}_m}\right)^T \\ &= q_i \frac{\partial}{\partial \boldsymbol{x}_n}\left(\varepsilon_{km} s_j \boldsymbol{e}_k^T + \varepsilon_{jm} s_k \boldsymbol{e}_j^T\right)^T \\ &= q_i \varepsilon_{km}\left(s_j \frac{\partial \boldsymbol{e}_k}{\partial \boldsymbol{x}_n} + \boldsymbol{e}_k \varepsilon_{jn} \boldsymbol{e}_j^T\right) + q_i \varepsilon_{jm}\left(s_k \frac{\partial \boldsymbol{e}_j}{\partial \boldsymbol{x}_n} + \boldsymbol{e}_j \varepsilon_{kn} \boldsymbol{e}_k^T\right) \\ &= q_i \varepsilon_{km}\left(s_j \varepsilon_{kn} \boldsymbol{I} + \boldsymbol{e}_k \varepsilon_{jn} \boldsymbol{e}_j^T\right) + q_i \varepsilon_{jm}\left(s_k \varepsilon_{jn} \boldsymbol{I} + \boldsymbol{e}_j \varepsilon_{kn} \boldsymbol{e}_k^T\right) \\ &= q_i(\varepsilon_{km}\varepsilon_{kn} s_j + \varepsilon_{jm}\varepsilon_{jn} s_k)\boldsymbol{I} + q_i\left(\varepsilon_{km}\varepsilon_{jn} \boldsymbol{e}_k \otimes \boldsymbol{e}_j + \varepsilon_{jm}\varepsilon_{kn} \boldsymbol{e}_j \otimes \boldsymbol{e}_k\right) \end{aligned}$$

and thus

$$\begin{aligned} \frac{\partial^2}{\partial \boldsymbol{x}_n \boldsymbol{x}_m}\left(\sum_i q_i s_j s_k\right) = &\sum_i q_i(\varepsilon_{km}\varepsilon_{kn} s_j + \varepsilon_{jm}\varepsilon_{jn} s_k)\boldsymbol{I} + \\ &\sum_i q_i\left(\varepsilon_{km}\varepsilon_{jn} \boldsymbol{e}_k \otimes \boldsymbol{e}_j + \varepsilon_{jm}\varepsilon_{kn} \boldsymbol{e}_j \otimes \boldsymbol{e}_k\right) \end{aligned} \tag{4.9}$$

To reduce this we can rotate the indices cyclically to obtain :

$$\sum_i q_i(\varepsilon_{km}\varepsilon_{kn} s_j + \varepsilon_{jm}\varepsilon_{jn} s_k) = \sum_i (q_i s_k + q_k s_i)\varepsilon_{jm}\varepsilon_{jn}$$

Using the same arguments as above, the right hand side of this equality can further simplify to :

$$\sum_i (q_i s_k + q_k s_i)\varepsilon_{jm}\varepsilon_{jn} = \begin{cases} -q_n s_m - q_m s_n, & m \neq n \\ \sum_{i\neq m} q_i s_k + q_k s_i, & m = n \end{cases}$$

Simplifying the second term in Eq. (4.9) is more complicated. However, for $l \neq m \neq n$ we can consider all the possible permutations of 1, 2, and 3 for l, m, and n (not necessarily cyclic). Expanding the sum for each permutation and evaluating all the ε terms then shows that regardless of the permutation we get :

$$\begin{aligned} &\sum_i q_i\left(\varepsilon_{km}\varepsilon_{jn} \boldsymbol{e}_k \otimes \boldsymbol{e}_j + \varepsilon_{jm}\varepsilon_{kn} \boldsymbol{e}_j \otimes \boldsymbol{e}_k\right) = \\ &q_n(\boldsymbol{e}_l \otimes \boldsymbol{e}_m) + q_m(\boldsymbol{e}_n \otimes \boldsymbol{e}_l) - q_l(\boldsymbol{e}_n \otimes \boldsymbol{e}_m), \quad l \neq m \neq n \end{aligned}$$

When we combine all these results we get for $l \neq m \neq n$:

$$\begin{aligned}\frac{\partial^2 W_{mem}}{\partial \boldsymbol{x}_n \boldsymbol{x}_m} =& \frac{\partial^2}{\partial \boldsymbol{x}_n \boldsymbol{x}_m}\left(\sum_i p_i s_i^2 + 2\sum_i q_i s_j s_k\right) \\ =& \; -2p_l\left(s_l \boldsymbol{I} + \boldsymbol{e}_l \otimes \boldsymbol{e}_l\right) - 2(q_n s_m + q_m s_n)\boldsymbol{I} + \\ & 2(q_n(\boldsymbol{e}_l \otimes \boldsymbol{e}_m) + q_m(\boldsymbol{e}_n \otimes \boldsymbol{e}_l) - q_l(\boldsymbol{e}_n \otimes \boldsymbol{e}_m)) \\ =& \; 2(q_n(\boldsymbol{e}_l \otimes \boldsymbol{e}_m) + q_m(\boldsymbol{e}_n \otimes \boldsymbol{e}_l) - q_l(\boldsymbol{e}_n \otimes \boldsymbol{e}_m) - p_l(\boldsymbol{e}_l \otimes \boldsymbol{e}_l)) \\ & -2(p_l s_l + q_n s_m + q_m s_n)\boldsymbol{I}\end{aligned}$$

or

$$-\frac{\partial^2 W_{mem}}{\partial \boldsymbol{x}_n \boldsymbol{x}_m} = \; r_l \boldsymbol{I} - 2(q_n(\boldsymbol{e}_l \otimes \boldsymbol{e}_m) + q_m(\boldsymbol{e}_n \otimes \boldsymbol{e}_l) - q_l(\boldsymbol{e}_n \otimes \boldsymbol{e}_m) - p_l(\boldsymbol{e}_l \otimes \boldsymbol{e}_l))$$

This result is in agreement with equation (16) in Delingette [2008] modulo a little bit of translation. As in Delingette [2008] the easiest way to compute the second derivative for $m = n$ is to use the fact (due to translation invariance) that :

$$\sum_i \frac{\partial^2 W_{mem}}{\partial \boldsymbol{x}_i \boldsymbol{x}_m} = 0$$

For implementation purposes we define the following shorthand notations :

$$\begin{aligned} H_{mn} &= \nabla_{\boldsymbol{x}_n}(\nabla_{\boldsymbol{x}_m} W_{mem})^T \\ Q_i &= 2q_i(\boldsymbol{e}_j \otimes \boldsymbol{e}_k) \\ P_i &= 2p_i(\boldsymbol{e}_i \otimes \boldsymbol{e}_i) \end{aligned}$$

The membrane force Jacobians can then be assembled from the following blocks :

$$\begin{aligned} H_{01} &= r_2 \boldsymbol{I} - Q_1 - Q_0 + Q_2^T + P_2 \\ H_{02} &= r_1 \boldsymbol{I} - Q_2^T - Q_0^T + Q_1 + P_1 \\ H_{23} &= r_0 \boldsymbol{I} - Q_2 - Q_1 + Q_0^T + P_0 \\ H_{00} &= -H_{01} - H_{02} \\ H_{11} &= -H_{01}^T - H_{12} \\ H_{22} &= -H_{02}^T - H_{12}^T \end{aligned}$$

4.6 Animated rest state

In the world of animation, physical correctness is not always the goal. In fact it can sometimes be useful to locally grow or shrink the cloth during

a simulation. As an example this can be used to avoid wrinkles in regions where wrinkles are not desired (for artistic reasons) by changing the rest state to follow a simple skinned surface underneath the cloth. As the surface stretches the amount of cloth grows so it doesn't induce shear buckling from the stretch, and as the surface shrinks the amount of cloth is reduced so it doesn't buckle due to compression. An example of this is shown in Fig. 4.2 where the inflatable vinyl is simulated as a thin shell.

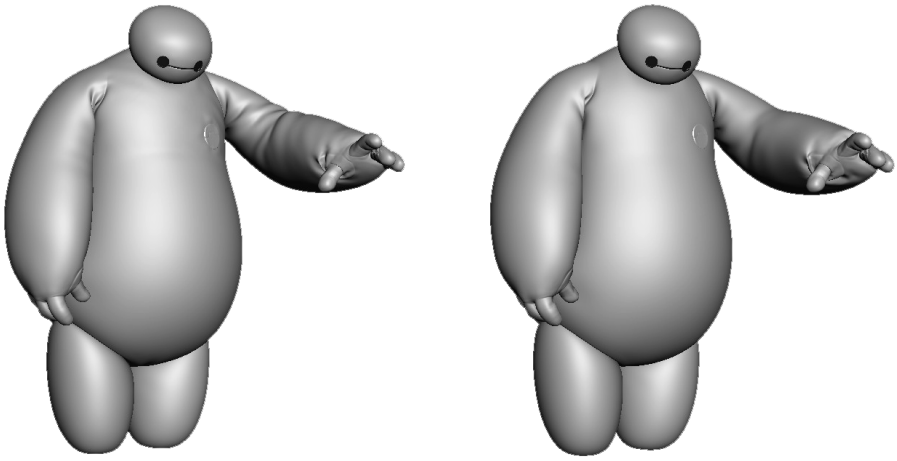


Figure 4.2: *The original simulation of Baymax from Disney's Big Hero 6 on the left shows more wrinkles than was artistically desirable. On the right the same simulation has been run with an animated rest state. © Disney*

Simply changing the rest state to be exactly what's given by the kinematic object, however, will simply replicate the kinematic object, which defeats the purpose of doing any simulation. Thus, it is desirable to be able to blend between the behavior given by the kinematic object and the behavior given by the original rest state.

Furthermore, it should ideally be possible to vary the relative weighting spatially (in one region you might want the kinematic behavior while you want the cloth behavior in another region). However, you cannot arbitrarily grow and shrink a piece of geometry in various places and still expect to have a single contiguous piece of geometry. You therefore cannot expect to simply be able to model or sculpt a new rest state for the geometry to achieve the desired effect.

To overcome these obstacles we represent the rest shape(s) by the corresponding set of edge lengths. The edge lengths represent an *intrinsic* measure of the surface because they can be measured without knowledge of the

embedding of the surface. This is contrary to the vertex positions whose position must be given with respect to the embedding space.

Given the intrinsic representation, we can interpolate between the edge lengths of the two rest shapes and use the result to define a new rest shape. Since the kinematically driven rest shape is typically animated, this is also referred to as "animated rest length" or "animated rest state".

This operation is different than simple interpolation between the vertices for two reasons. Most obvious is the fact that each edge length is a nonlinear function of the vertex positions. However, more importantly the "rest shape" given by the kinematic deformation is typically far from the true rest configuration, so linear interpolation between this shape and the true rest shape of the cloth is unlikely to produce a reasonable looking shape. By encoding everything intrinsically the actual configuration of the shape no longer matters.

In order for this idea to work it must be possible to express the constitutive model in terms of edge lengths rather than vertex positions. Looking back at Eq. (4.7) and Eq. (4.8), this requires that all the dot products in Eq. (4.7) can be computed and that the area of the triangle in Eq. (4.8) can also be computed.

The area is easy to compute since Heron's formula states that :

$$A = \tfrac{1}{4}\sqrt{(\|\boldsymbol{e}_0\|^2 + \|\boldsymbol{e}_1\|^2 + \|\boldsymbol{e}_2\|^2)^2 - 2(\|\boldsymbol{e}_0\|^4 + \|\boldsymbol{e}_1\|^4 + \|\boldsymbol{e}_2\|^4)}.$$

To compute the dot products we refer to Fig. 4.1 and note from the standard relation between the dot product and the angle between two vectors that :

$$\boldsymbol{e}_j \cdot \boldsymbol{e}_k = -\|\boldsymbol{e}_j\|\|\boldsymbol{e}_k\| \cos\alpha_i$$

From the law of cosines we have :

$$\|\boldsymbol{e}_i\|^2 = \|\boldsymbol{e}_j\|^2 + \|\boldsymbol{e}_k\|^2 - 2\|\boldsymbol{e}_j\|\|\boldsymbol{e}_k\| \cos\alpha_i$$

Combining the two we get :

$$\|\boldsymbol{e}_i\|^2 = \|\boldsymbol{e}_j\|^2 + \|\boldsymbol{e}_k\|^2 + 2\boldsymbol{e}_j \cdot \boldsymbol{e}_k$$

or

$$\boldsymbol{e}_j \cdot \boldsymbol{e}_k = \tfrac{1}{2}(\|\boldsymbol{e}_i\|^2 - \|\boldsymbol{e}_j\|^2 - \|\boldsymbol{e}_k\|^2) \tag{4.10}$$

For the orthotropic component we assume that the orthotropic direction, $\hat{\boldsymbol{a}}$, is given with respect to the true rest configuration of the cloth. Given this

orientation we solve for two scalar coefficients such that $\hat{\boldsymbol{a}}_\perp = \omega_0 \boldsymbol{e}_0 + \omega_1 \boldsymbol{e}_1$. At this point we then have

$$\begin{aligned}
\hat{\boldsymbol{a}}_\perp \cdot \boldsymbol{e}_0 &= \omega_0 \|\boldsymbol{e}_0\|^2 + \omega_1 (\boldsymbol{e}_1 \cdot \boldsymbol{e}_0) \\
\hat{\boldsymbol{a}}_\perp \cdot \boldsymbol{e}_1 &= \omega_0 (\boldsymbol{e}_0 \cdot \boldsymbol{e}_1) + \omega_1 \|\boldsymbol{e}_1\|^2 \\
\hat{\boldsymbol{a}}_\perp \cdot \boldsymbol{e}_2 &= \omega_0 (\boldsymbol{e}_0 \cdot \boldsymbol{e}_2) + \omega_1 (\boldsymbol{e}_1 \cdot \boldsymbol{e}_2)
\end{aligned}$$

This can be evaluated given edge lengths by using Eq. (4.10).

CHAPTER 5

Discrete shells

Having covered the membrane model in the two preceding chapters, the focus in this chapter is on efficient evaluation of the Hessians of the bending energy for the discrete shells bending model. This model was first presented in [Grinspun et al., 2003], while an equivalent formulation appeared in simultaneously in [Bridson et al., 2003]. The material model used earlier by Baraff and Witkin [1998] is essentially an approximation of the discrete shell model.

The discrete shell bending energy is based on the dihedral angle between two adjacent triangles (see Fig. 5.1). Unlike many of the commonly used models in shell analysis it is well suited for large deformations which are typical in cloth simulations. Models based on linear elasticity are insufficient for this, and we note that models based on co-rotational linear elasticity (see e.g. [Thomaszewski et al., 2006; Kaufmann et al., 2009]) are also insufficient since they assume that the strain is small everywhere (even if the displacements are large).

The dihedral angle is straightforward to compute given the vertex positions, and its gradient with respect to vertex positions (required for the forces) is easily found in the literature. However, the Hessian of the bend angle, which is required to compute the associated force Jacobians, has not been documented in the literature prior to our contribution in [Tamstorf and Grinspun, 2013].

Readily available computations of the force Jacobians, such as those produced by symbolic algebra systems, or by autodifferentiation codes, are ex-

pensive to compute, and therefore less useful. In this chapter, we present compact, easily reproducible, closed form expressions for the Hessian of the bend angle. Compared to automatic differentiation, we measure up to 7× speedup for the evaluation of the bending forces and their Jacobians.

In addition to the application for cloth and thin shell simulation, the hinge angle is being used for the

- deformation energy for example-driven deformations [Fröhlich and Botsch, 2011]
- Willmore energy used in mesh smoothing [Wardetzky et al., 2007], and
- dissipative potential of viscous liquid sheets [Batty et al., 2012].

These applications therefore also stand to benefit from the work presented in this chapter.

5.1 Background and overview

Part of the motivation for this work has been our experience that analytical derivatives are exceedingly tedious to derive by hand, with compact formulations sometimes consuming weeks of manual derivation. This process is error prone, often leading to analytic expressions that disagree with numerical validation. The process can be suboptimal, missing opportunities for gathering like terms, thus leading to longer source code and more expensive computation. These liabilities are detrimental to the adoption of efficient numerical methods for hinge-based energies, as evidenced in the literature:

- Bridson et al. [2003] avoided Hessians by treating bending forces explicitly; similarly, Fröhlich and Botsch [2011] avoided Hessians by using Gauss-Newton's method;
- Baraff and Witkin [1998] introduced approximating assumptions (e.g., inextensible cloth, undergoing only small deformations, with flat rest shape) treating normals and edge lengths as constants;
- Bergou et al. [2006] and Wardetzky et al. [2007] derived a simplified Hessian formula for the special case energy $\sin^2(\theta/2)$, using a technique that does not accommodate the general case; and
- Grinspun et al. [2003] computed the Hessian using automatic differentiation, which dominated the computational cost of the method.

Contributions. In light of these observations, this chapter seeks to facilitate adoption, code legibility, and computational efficiency of hinge-based bending energies.

- We present a compact and efficient formulation of the Hessian for the general case of a hinge-based bending energy.
- By taking advantage of several symmetries in the expressions (some less obvious than others), we observe that many terms can be reused when assembling the Hessian for an entire mesh, further reducing the cost of computation.
- We present the results of experiments documenting up to 7× speedup of the formulation compared to autodifferentiation and up to 4× speedup compared to an existing (but unpublished) symbolic derivation.

5.2 Orthotropy

The extension of the discrete shell bending model to orthotropic materials was presented by Garg et al. [2007]. The underlying assumption in that paper is that the elasticity tensor is constant as in the St. Venant-Kirchhoff model. Given that assumption, the orthotropic model amounts to a scaling of the stiffness associated with each edge. As such, the work presented here is immediately applicable in their framework too.

5.3 Bending energy

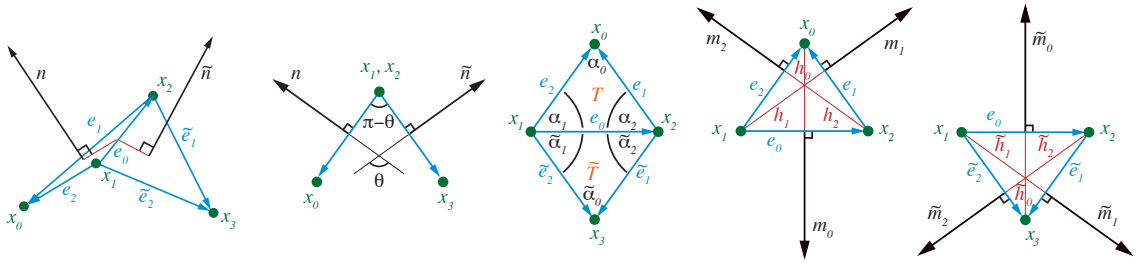


Figure 5.1: *Vertices, edges, normals, and angles around the edge shared by two triangles. The two rightmost schematics show the in-plane edge normals and the associated altitudes from one edge to the opposing vertex. All of these quantities are straightforward to compute given the edge vectors.*

Notation. Figure 5.1 presents the labels and indices for a single *hinge stencil*, consisting of four vertices $\boldsymbol{x}_i$, five edges $\boldsymbol{e}_i$ and $\tilde{\boldsymbol{e}}_i$, two normals $\boldsymbol{n}$ and $\tilde{\boldsymbol{n}}$, bend angle θ, interior angles α_i and $\tilde{\alpha}_i$, and heights h_i and $\tilde{h}_i$. Typically, the index i takes on values $0, 1$ and 2. Arithmetic on all indices is performed modulo 3. Observe that edges (and all related quantities) are generally labeled the same as the opposing vertices. The tilde decoration is used to distinguish corresponding quantities on the upper and lower triangles T and $\tilde{T}$, respectively. Triangle and edge normals are all assumed to be normalized.

Energy. For a given triangle mesh, consider an arbitrary energy given by a summation over all the interior edges (indexed by i), or "hinges," of a triangle mesh,

$$W_{bend}(\boldsymbol{x}) = \sum_i \psi_i(\theta_i) \,, \tag{5.1}$$

The "bend angle" θ here is the angle between the normals of the two triangles incident to the hinge, and $\psi : \mathbb{R} \to \mathbb{R}$ is an application-specific transformation of the bend angle. Drawing from the literature, examples for $\psi_i(\theta_i)$ include

$a_i(\theta_i - b_i)^2$	*Discrete shells [Grinspun et al., 2003]*
$a_i(\sin(\theta_i/2))^2$	*Discrete Willmore energy [Wardetzky et al., 2007]*
$a_i(\cos(\theta_i/2) - b_i\theta_i)$	*Simulation of clothing [Bridson et al., 2003]*

where a_i and b_i are application-specific scalar coefficients. These typically depend on the local geometry of the mesh and, in physical simulations, the material constitutive properties.

Bridson et al. [2003] presented a force, not an energy; above we have integrated the (conservative) force to obtain the corresponding energy. By focusing on the conservative setting, we can roughly halve the computation time, since the conservative force Jacobian is the negated energy Hessian, which is symmetric by definition.

In Chapter 3, we used ψ to refer to an continuous energy density. Here, ψ_i effectively refers to a discretized (bend) energy associated with one hinge.

Bending forces and Hessians. Let the set of all vertex positions be denoted by $\mathbf{q} = (\boldsymbol{x}_0, \ldots, \boldsymbol{x}_3)$. We then differentiate the energy, Eq. (5.1), with respect to $\mathbf{q}$ to obtain the bending forces and the energy Hessian :

$$\boldsymbol{f}(\boldsymbol{x}) = -\sum_i \nabla\psi_i \quad \text{and} \quad H(\boldsymbol{x}) = \sum_i \mathrm{Hess}(\psi_i) \,.$$

For one particular hinge i, dropping the implied subscript from ψ_i and θ_i, the chain rule gives

$$\nabla\psi = \psi'\nabla\theta\,, \tag{5.2}$$

$$\mathrm{Hess}(\psi) = \psi'\,\mathrm{Hess}(\theta) + \psi''\nabla\theta^T\nabla\theta\,, \tag{5.3}$$

using the prime to differentiate a univariate function with respect to its scalar argument, e.g., $\psi' = \mathrm{d}\psi/\,\mathrm{d}\theta$.

Observe that the Hessian of the energy is a weighted sum of $\mathrm{Hess}(\theta)$ and the outer product $\nabla\theta^T\nabla\theta$, with the same weighting function ψ' appearing in both $\nabla\psi$ and $\mathrm{Hess}(\psi)$.

5.4 Hinge-angle gradient and Hessian

The expression for $\nabla\theta$ has been previously documented in the literature in several forms equivalent to

$$\begin{aligned}
\nabla_{x_1}\theta &= \frac{\cos\alpha_2}{h_1}\boldsymbol{n}^T + \frac{\cos\tilde{\alpha}_2}{\tilde{h}_1}\tilde{\boldsymbol{n}}^T, & \nabla_{x_0}\theta &= -\frac{1}{h_0}\boldsymbol{n}^T,\\
\nabla_{x_2}\theta &= \frac{\cos\alpha_1}{h_2}\boldsymbol{n}^T + \frac{\cos\tilde{\alpha}_1}{\tilde{h}_2}\tilde{\boldsymbol{n}}^T, & \nabla_{x_3}\theta &= -\frac{1}{\tilde{h}_0}\tilde{\boldsymbol{n}}^T\,.
\end{aligned} \tag{5.4}$$

By contrast, the expressions for the hinge angle Hessian are not (to our knowledge) recorded in the literature. Like others, we found the derivation to be extended and error-prone, and have therefore archived a complete derivation in a technical report [Tamstorf, 2013]. The final expressions for $H^\theta \equiv \mathrm{Hess}(\theta)$ are conveniently expressed in terms of the building blocks :

$$\begin{aligned}
S(\cdot) &= (\cdot) + (\cdot)^T\,,\\
\omega_{ij} &= 1/(h_i h_j)\,, & \tilde{\omega}_{ij} &= 1/(\tilde{h}_i\tilde{h}_j)\,,\\
\boldsymbol{M}_i &= \boldsymbol{n}\boldsymbol{m}_i^T\,, & \tilde{\boldsymbol{M}}_i &= \tilde{\boldsymbol{n}}\tilde{\boldsymbol{m}}_i^T\,,\\
\boldsymbol{N}_i &= \boldsymbol{M}_i/\|\boldsymbol{e}_i\|^2\,, & \tilde{\boldsymbol{N}}_i &= \tilde{\boldsymbol{M}}_i/\|\tilde{\boldsymbol{e}}_i\|^2\,,\\
\boldsymbol{P}_{ij} &= \omega_{ij}\cos\alpha_i\boldsymbol{M}_j^T\,, & \tilde{\boldsymbol{P}}_{ij} &= \tilde{\omega}_{ij}\cos\tilde{\alpha}_i\tilde{\boldsymbol{M}}_j^T\,,\\
\boldsymbol{Q}_j &= \omega_{0j}\boldsymbol{M}_j\,, & \tilde{\boldsymbol{Q}}_j &= \omega_{0j}\tilde{\boldsymbol{M}}_j\,.
\end{aligned} \tag{5.5}$$

The 3×3 subblocks, H^θ_{ij}, of H^θ are then

$$
\begin{aligned}
H^\theta_{00} &= -S(\boldsymbol{Q}_0) \\
H^\theta_{33} &= \qquad\qquad -S(\tilde{\boldsymbol{Q}}_0) \\
\\
H^\theta_{11} &= S(\boldsymbol{P}_{11}) - \boldsymbol{N}_0 \qquad +S(\tilde{\boldsymbol{P}}_{11}) - \tilde{\boldsymbol{N}}_0 \\
H^\theta_{22} &= S(\boldsymbol{P}_{22}) - \boldsymbol{N}_0 \qquad +S(\tilde{\boldsymbol{P}}_{22}) - \tilde{\boldsymbol{N}}_0 \\
\\
H^\theta_{10} &= \boldsymbol{P}_{10} - \boldsymbol{Q}_1 \\
H^\theta_{20} &= \boldsymbol{P}_{20} - \boldsymbol{Q}_2 \\
H^\theta_{13} &= \qquad\qquad \tilde{\boldsymbol{P}}_{10} - \tilde{\boldsymbol{Q}}_1 \\
H^\theta_{23} &= \qquad\qquad \tilde{\boldsymbol{P}}_{20} - \tilde{\boldsymbol{Q}}_2 \\
\\
H^\theta_{12} &= \boldsymbol{P}_{12} + (\boldsymbol{P}_{21})^T + \boldsymbol{N}_0 \qquad +\tilde{\boldsymbol{P}}_{12} + (\tilde{\boldsymbol{P}}_{21})^T + \tilde{\boldsymbol{N}}_0 \\
H^\theta_{03} &= 0\,.
\end{aligned}
\tag{5.6}
$$

$\overbrace{\qquad\qquad}^{\text{contribution of upper triangle}}$ $\overbrace{\qquad\qquad}^{\text{contribution of lower triangle}}$

The remaining blocks are obtained by symmetry of the Hessian, $H^\theta_{ij} = (H^\theta_{ji})^T$.

Exploiting symmetry. We have taken special care in laying out the expressions above, and in assigning the labels in Fig. 5.1. Observe that every contributing term depends on quantities from the hinge's upper triangle T, or lower triangle $\tilde{T}$, but not both. We write terms depending on T on the left column, and terms depending on $\tilde{T}$ on the right column. Comparing the two columns, we observe that *the two triangles contribute to the Hessian symmetrically.*

We now exploit this symmetry to derive a novel refactorization of the Hessian expressions, yielding a simpler, and more efficient, implementation.

5.5 Refactoring the bending energy Hessian

Assembling the Hessian for an entire mesh. Recall from Eq. (5.3) that the bending energy Hessian, $H(\boldsymbol{x})$, is the weighted sum of the hinge-angle Hessian, $\psi' \operatorname{Hess}(\theta)$, and the outer product of the hinge angle gradient with itself, $\psi'' \nabla\theta^T \nabla\theta$. Therefore, it is natural to split the computation of the energy

Hessian into two parts, iterating over triangles to compute $\sum \psi_i' \operatorname{Hess}(\theta_i)$, and iterating over interior edges to compute $\sum \psi_i'' \nabla\theta_i^T \nabla\theta_i$; we examine these two parts in Section 5.5.1 and Section 5.5.2, respectively.

5.5.1 Exploiting two levels of symmetry in $\psi' \operatorname{Hess}(\theta)$

The half-hinge. As is evident from the two columns of Eq. (5.6), the two triangles of a hinge contribute to the hinge angle Hessian symmetrically. We exploit this symmetry by thinking of each hinge as a pair of *half-hinges* (see Fig. 5.2a).

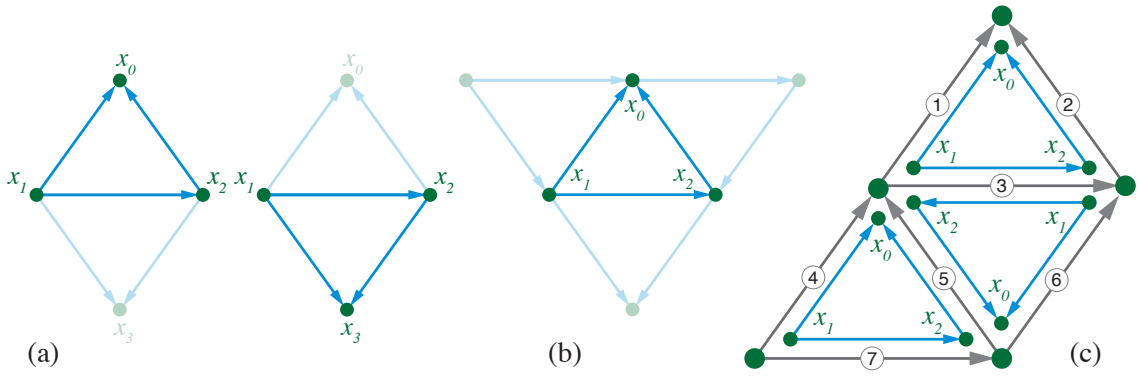


Figure 5.2: Refactoring the assembly: (a) *Every hinge is split into two* half-hinges. (b) *Each triangle associates with up to three half-hinges.* (c) *The Hessian is assembled by iterating a simple template over mesh triangles. In each iteration, the local indices of the template are mapped to corresponding global indices, and care is taken to account for mismatch in local/global edge orientation.*

The contributions of each half-hinge can each be computed using only the left column of Eq. (5.6), making the code compact, if care is taken to correct for the orientation of hinge edge $\boldsymbol{e}_0$.

To understand the needed correction, recall that the left and right columns of Eq. (5.6) are expressed with indices into the upper and lower triangles of Fig. 5.1, respectively. Hinge edge $\boldsymbol{e}_0$ flows *counter*clockwise versus clockwise along the upper and lower triangles, respectively. This disagreement in the assumed orientation of $\boldsymbol{e}_0$ is the only difference in the expressions derived for the left and right columns. In particular, if the lower half-hinge's contribution is computed using the expressions originally derived for the upper half-hinge, we must account for the reversal of $\boldsymbol{e}_0$.

To understand what is affected, we can rederive Eq. (5.6) with a revised Fig. 5.1 in which $\boldsymbol{e}_0$ is reversed. It turns out that the reversal affects only the computation of H_{12}, where $\boldsymbol{N}_0$ and $\tilde{\boldsymbol{N}}_0$ are now transposed. Since $\boldsymbol{N}_0 + \tilde{\boldsymbol{N}}_0$ is symmetric, transposing both does not alter the result, which is reassuring, since the Hessian should not depend on the (arbitrary choice of) orientation of $\boldsymbol{e}_0$ in the diagram. However, when we split the Hessian computation into a pair of half-hinges, both reusing the left column of Eq. (5.6), this amounts to reversing the orientation in Fig. 5.1 only for the lower half-hinge: the pair of half-hinges are now computed with inconsistent versions of Fig. 5.1, an error we must correct: *we must transpose $\boldsymbol{N}_0$ for exactly one of the two half-hinge applications of the left column of Eq. (5.6)*. This is reflected below in our final computation.

The three half-hinges of a triangle. A second level of symmetry is uncovered by observing that each triangle participates in up to three half-hinges (see Fig. 5.2b). Because these half-hinges involve the same triangle, their contributions to the bending energy Hessian all depend on the same set of local quantities. It therefore becomes natural to compute the bending energy Hessian by examining one triangle at a time.

Local triangle energy Hessian. The complete matrix $\sum \psi_i' \operatorname{Hess}(\theta_i)$ is assembled in the usual style of finite element stiffness matrix assembly, by visiting each triangle and computing a local Hessian $H^{\triangle}$.

Consider the contribution of one triangle. If the triangle lies in the mesh interior, it participates in three half-hinges, but if it is incident to a boundary, it may participate in fewer hinges. To account for the boundary cases without specialized formulae, we introduce the indicator function

$$\chi_i = \begin{cases} 1 & \text{edge } i \text{ lies in interior}, \\ 0 & \text{edge } i \text{ lies on boundary}. \end{cases}$$

We instantiate the left column of Eq. (5.6) three times, with labels permuted in correspondence to each of the three half-hinges. Per Eq. (5.3), we scale each half-hinge Hessian contribution by ψ_i', and sum the scaled contributions to arrive at the *local triangle Hessian*. The use of the indicator function, and the summation over three potentially participating half-hinges, exposes the second level of symmetry— a three-fold symmetry over the edges, vertices, and indeed all labels on the triangle. This allows for a surprisingly compact representation of the 3×3 subblocks of $H^{\triangle}$ as

$$H_{ij}^{\triangle} = \omega_{ij} \left(d_i \boldsymbol{M}_j^T + d_j \boldsymbol{M}_i \right) + \begin{cases} -\boldsymbol{R}_{i+1} - \boldsymbol{R}_{i+2} & i = j, \\ {\boldsymbol{R}_{i+2}}^{\dagger} & i \neq j, \end{cases}$$

where

$$\begin{aligned} c_i &= \chi_i \psi_i'(\theta_i) \,, \\ d_i &= c_{i-1} \cos \alpha_{i+1} + c_{i+1} \cos \alpha_{i-1} - c_i \,, \\ \boldsymbol{R}_i &= c_i \boldsymbol{N}_i \,. \end{aligned} \tag{5.7}$$

These expressions are valid for $i \in \{0,1,2\}$, $j \in \{i, i+1\}$, with the remaining subblocks being determined by $H_{ij}^{\triangle} = (H_{ji}^{\triangle})^T$.

The conditional transpose operator $(\dagger)$. The above expressions employ the conditional transpose operator denoted by a dagger: $\boldsymbol{R}_i^{\dagger}$ transposes $\boldsymbol{R}_i$ if and only if the orientation of mesh edge $\boldsymbol{e}_i$ is counterclockwise with respect to the triangle of interest. The choice of global orientation is immaterial so long as, it is held fixed throughout the assembly of the complete Hessian. Indeed, the precise definition of $\dagger$ is also immaterial, so long as for every interior edge, it transposes for exactly one of the two incident triangles.

Operation count. To assemble the local Hessian for one triangle, we first compute three cosine expressions needed for d_i, and three outer products $\boldsymbol{M}_i$ assembled in 12 linear combinations. This is more compact than a naïve computation of the Hessian for each hinge. The Hessian for a single full hinge requires six different outer products, 20 scaled versions of these, and 18 matrix additions. For a regular mesh, there are twice as many edges as faces, so the total (relative) cost for the entire mesh becomes 12 outer products, 40 scale operations, and 36 additions vs. 3, 12, and 15 operations respectively. Assuming that the computation is compute bound, we should therefore expect roughly a $3\times$ speedup. However, the locality of the above computation also improves cache-coherency, in practice leading to an additional speedup.

5.5.2 Computing $\psi'' \nabla\theta^T \nabla\theta$

All but two of the subblocks in the outer product contain mixed terms, i.e., terms involving data from both of incident triangles. Therefore, the outer product does not decompose in the way of the hinge angle Hessian; it is more naturally computed *per edge*.

However, while it is conceptually most naturally computed per edge, in practice it is still advantageous to include it in the same loop as the hinge-angle Hessian. We do this by assigning each edge to one of its two incident

triangles similar to how the conditional transpose operator only transposes for one of the two triangles. The advantage of using this approach is primarily that the calls to the assembly function for the global stiffness matrix can be consolidated. Assume that each call to the assembly function adds one subblock of the Hessian. If the two loops are kept separate, then a total of $6F + 10E$ calls to the assembly function are needed (where F denotes the number of faces and E denotes the number of edges). In a regular mesh, this is approximately equal to $13E$. By comparison, the combined loop only requires $6F + 4E \approx 7E$ calls to the assembly function (the naïve computation of the Hessian requires $10E$ calls to the assembly function).

5.6 Implementation of a thin shell code testbed

Our motivation to derive the hinge energy Hessian in this thesis stems from the need to implement an implicit time stepper for cloth simulation. To obtain a complete implementation for a cloth simulation, several additional observations are useful. We present these in this section, and then use this framework for the performance comparisons in the next section.

Tan-based energy. Discrete Shells [Grinspun et al., 2003] employ the hinge bending energy

$$\underbrace{\psi_i(\theta_i)}_{\textit{Discrete Shells}} = k\frac{3\|\bar{\boldsymbol{e}}_i\|^2}{\bar{A}_i}(\theta_i - \bar{\theta}_i)^2 \,,$$

where k is a bending stiffness and $\bar{A}_i$ is the sum of the areas of the two triangles incident to the hinge. A bar indicates that the quantity refers to the undeformed configuration.

More generally, one could consider some discrete approximation of the locally integrated mean curvature, $\varphi(\theta_i)$, and then express the hinge bending energy as

$$\underbrace{\psi_i(\theta_i)}_{\textit{Generalized}} = k\, a_i \left(\varphi(\theta_i) - \varphi(\bar{\theta}_i)\right)^2 \,, \tag{5.8}$$

where a_i is a scaling coefficient that accounts for the local discrete hinge geometry. This is similar to the formulation by Gingold et al. [2004].

We found $\varphi(\theta) = 2\tan(\frac{\theta}{2})$ to be useful in our application. This expression was used by Bobenko and Suris [1999], Hoffmann [2000], and Gingold et al.

[2004] to estimate the sum of the principal curvatures. The tan-based energy is convenient in practice because it leads to monotonically increasing forces as the bend angle increases. For this particular choice of $\varphi(\theta)$, the scaling coefficient remains $a_i = 3\|\bar{\boldsymbol{e}}_i\|^2 / \bar{A}_i$.

Physical materials do not interpenetrate, so we require

$$\theta \in]-\pi, \pi[\,, \tag{5.9}$$

$$\bar{\theta} \in]-\pi, \pi[\,. \tag{5.10}$$

The tan-based energy enforces Eq. (5.9), since the restoring force becomes unbounded as $\theta \to \pm\pi$; the condition Eq. (5.10) is enforced at initialization.

We also considered whether to employ $\varphi(\theta - \bar{\theta})$ in Eq. (5.8) in place of $\varphi(\theta) - \varphi(\bar{\theta})$. We choose the latter because the non-penetration condition Eq. (5.9) examines θ, not $\theta - \bar{\theta}$. Some earlier papers, perhaps less focused on enforcement of non-penetration in the elastic model, *do* employ $\varphi(\theta - \bar{\theta})$ [Garg et al., 2007; Wardetzky et al., 2008]. For small (infinitesimal) bend-angles, all of these methods are equivalent (the curvature approaches zero as the bend-angles approaches zero). However, for large (finite) bend-angles, the distinction becomes important.

Bending stiffness. For actual simulations, the bending stiffness, k, must be chosen. By considering the energy of a thin plate under small deflections, [Audoly and Pomeau, 2010, Sec. 6.6], we note that it can be identified with *half* the flexural rigidity, D. While we consider shells here rather than plates, the model must be consistent with the plate model for a flat rest state. Hence,

$$k = \frac{D}{2} = \frac{Yh^3}{24(1-\nu^2)}, \tag{5.11}$$

where Y denotes Young's modulus, ν is Poisson's ratio, and h is the thickness of the shell.

Trigonometric functions of signed hinge angle. To be able to determine which way a shell is bending, the bend-angle $\theta \in]-\pi, \pi[$ has to be treated as a signed quantity. We use the same convention as Bridson et al. [2003], where θ has the same sign as $(\boldsymbol{n}_1 \times \boldsymbol{n}_2) \cdot \boldsymbol{e}_0 = \det[\boldsymbol{n}_1, \boldsymbol{n}_2, \boldsymbol{e}_0]$, i.e., positive when the two normals point away from each other.

Referring to Fig. 5.3, the needed trigonometric functions for $\frac{\theta}{2}$ are

$$\sin\left(\tfrac{\theta}{2}\right) = \frac{\|\boldsymbol{n}_1 - \boldsymbol{n}_2\|}{2}\,, \qquad \cos\left(\tfrac{\theta}{2}\right) = \frac{\|\boldsymbol{n}_1 + \boldsymbol{n}_2\|}{2}\,,$$

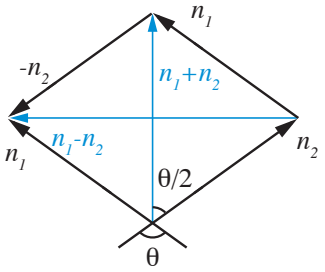


Figure 5.3: *Simple construction to compute the trigonometric functions for $\frac{\theta}{2}$ based on one of the right angled triangles. Note that each of the normal vectors have unit length.*

for $\theta \in [0, \pi]$. For values of θ outside the specified interval, the usual identities for trigonometric functions can be used. In particular, this gives

$$\tan\left(\frac{\theta}{2}\right) = \operatorname{sgn}(\det[\boldsymbol{n}_1, \boldsymbol{n}_2, \boldsymbol{e}_0]) \frac{\|\boldsymbol{n}_1 - \boldsymbol{n}_2\|}{\|\boldsymbol{n}_1 + \boldsymbol{n}_2\|}, \tag{5.12}$$

for $\theta \in]-\pi, \pi[$. For $\theta = 0$, the determinant becomes zero and the sign function is undefined. However, in this case, we have $\boldsymbol{n}_1 - \boldsymbol{n}_2 = 0$ so, as long as any finite value is chosen for the sign function, the result is correct. It should be noted that we never have to differentiate the right hand side of Eq. (5.12). Instead, we differentiate $\tan(\theta/2)$ using the chain rule. This avoids the trouble of the non-differentiability of the sign function at zero.

Efficient computation of viscous forces. The *discrete Rayleigh analogy* implementation of viscous damping introduces an *incremental dissipative potential*,

$$E_v(\boldsymbol{x}) = \frac{k_d}{\tau} \sum_i a_i \left(\varphi(\theta_i) - \varphi(\hat{\theta}_i)\right)^2,$$

where $\hat{\theta}_i$ is the value of θ_i at the end of the previous time step and τ is the size of the time step, [Bergou et al., 2010]. In this model, viscosity is modeled as an elastic-type energy that binds the end-of-step position to its start-of-step counterpart. The parameter k_d is the damping coefficient, which is conceptually similar to k in the elastic energy.

The *combined elastic and viscous* energy for a time step is then given by

$$\underbrace{\psi_i(\theta)}_{\textit{viscoelastic}} = k a_i \left(\varphi(\theta_i) - \varphi(\bar{\theta}_i)\right)^2 + \frac{k_d}{\tau} a_i \left(\varphi(\theta_i) - \varphi(\hat{\theta}_i)\right)^2,$$

and the expressions for the force and Hessian follow as before. In particular, per Eq. (5.3), the expensive hinge angle gradient, $\nabla\theta$, and Hessian, $\mathrm{Hess}(\theta)$, are computed just once (not twice), and the independent elastic and viscous contributions appear only in the scalars ψ' and ψ''. Compared to a separate evaluation of elastic and viscous forces, this provides a $2\times$ speedup since none of the vector or matrix computations are duplicated.

5.7 Method in brief

For convenience, we collect all the equations into pseudocode in Algorithm 1 and 2.

Algorithm 1: *ComputeForcesAndGradients*

Input: Deformed mesh

for all edges, $\boldsymbol{e}_i$ **do**
  Compute inverse edge lengths, $1/l_i = 1/\|\boldsymbol{e}_i\|$.
  Compute unit edge, $\hat{\boldsymbol{e}}_i = \boldsymbol{e}_i/l_i$.
  Compute $\varphi(\theta_i)$ (Eq. 5.12).
  Compute ψ' and ψ''.
end for
for all triangles, $\mathcal{T}$ **do**
  Compute area, A, of triangle.
  Compute cosines, $\cos\alpha_i = \hat{\boldsymbol{e}}_j \cdot \hat{\boldsymbol{e}}_k$.
  Compute inverse altitudes, $1/h_i = l_i/2A$.
  Compute edge normals, $\boldsymbol{m}_i = \hat{\boldsymbol{e}}_i \times \boldsymbol{n}$.
  Compute $d_i, \omega_{ij}, \boldsymbol{M}_i, \boldsymbol{N}_i, \boldsymbol{R}_i$ (Eqs. 5.5 and 5.7).
  Compute contributions to the Hessian, $H^{\triangle}_{ij}$.
  Assemble $\nabla \boldsymbol{f}_{ij} = H^{\triangle}_{ij}$.
  for all non-boundary CCW edges in $\mathcal{T}$ **do**
    Compute bend gradient, $\nabla\theta$ (Eq. 5.4).
    Compute bending forces (Eq. 5.2).
    Compute $\psi''\nabla\theta^T\nabla\theta$.
    Update $\nabla \boldsymbol{f} = \nabla \boldsymbol{f} + \psi''\nabla\theta^T\nabla\theta$.
  end for
end for

Algorithm 2: *Preprocessing*

Input: Undeformed mesh
1: **for all** edges, e_i **do**
2: Compute $\varphi(\bar{\theta}_i)$ using Eq. (5.12).
3: Compute $a_i = 3\|\bar{e}_i\|^2 / \bar{A}_i$.
4: Compute k using Eq. (5.11).
5: **end for**

The computation of k may need to be done in different places depending on whether the material parameters are spatially and/or temporally varying. Note that, on current microprocessors, it can be advantageous in Algorithm 1 to compute and store the inverse of the edge lengths and triangle altitudes. That way the subsequent number of divisions is reduced significantly.

5.8 Evaluation

To illustrate the practical benefits of the results presented in the preceding sections, we compare the performance and accuracy of our implementation to two existing alternatives. In particular, we compare against the following methods:

- *Automatic differentiation*. Originally, Grinspun et al. [2003] proposed to evaluate the force Jacobians using automatic differentiation. This method is characterized by providing accurate results without having to perform a lot of manual algebra.
- *Symbolic derivation*. The results presented in a number of papers, [Bergou et al., 2006; Wardetzky et al., 2007; Garg et al., 2007; Batty et al., 2012], are based on an unpublished symbolic derivation of the bending forces and their gradients, but this derivation does not leverage all the available symmetry.

For each of these methods, our comparison is based on the original source code provided by the authors, but ported into our testbed for a fair comparison. The comparison focuses on the computation of $\nabla\theta$ and $\mathrm{Hess}(\theta)$, so all other computations are kept the same among all the implementations.

5.8.1 Test cases

We consider a number of different cloth simulations for our evaluations. One set of these has a square piece of cloth falling onto and draping over a (static) sphere (see Figure 5.4). This simulation is run at 36 different cloth resolutions (ranging from 121 vertices up to 6561 vertices). The goal of this test is to evaluate the behavior of the methods in the presence of large bend angles. No explicit damping is included in this example.

Figure 5.4: *A piece of cloth draping over a sphere. This simulation is run at 36 different resolutions.*

Another set of simulations, shown in Figure 5.5, have a horizontal "beam" of varying thickness, where one end is fully constrained while the remainder of the beam is allowed to bend under gravity. The goal of these simulations is to evaluate the behavior for a range of different materials from soft cloth to relatively stiff thin shells. These simulations have been damped to reach an equilibrium relatively quickly. In practice, 5 different materials are considered and each one is run at three different resolutions (366, 787, and 1372 vertices).

All benchmarks were run on an Intel Core i7-2640 at 2.80 GHz. Since this work is not focused on parallelization, everything is run in a single thread for easier comparisons. However, both the assembly of the Hessian and the solution of the resulting linear systems can obviously be parallelized.

Figure 5.5: *Five different beams of varying thickness. The beams will be referred to by number with the one in front being number 1 and the one farthest away being number 5.*

5.8.2 Numerical accuracy

All of the methods considered for comparison should compute identical results in exact arithmetic. In floating point arithmetic slight differences are to be expected due to rounding operations of intermediate results. To confirm the correctness of the new method and to evaluate its accuracy compared to the existing methods, we computed the normwise relative error of the Hessian at each time step of all the cloth-sphere simulations.

Let H_{new} denote the Hessian of the bending energy using the method presented in this paper, and let H_{old} denote the Hessian based on one of the existing methods. The normwise relative error is then given by

$$\eta = \frac{\|H_{\text{new}} - H_{\text{old}}\|}{\|H_{\text{new}}\|} .$$

For our computations, we used the Frobenius norm in the above expression. The distribution of the resulting errors is shown in Figure 5.6. As can be seen from this figure, the difference between the various implementations is typically on the order of $1-2$ ulp (unit in last place). The maximum error observed was 4.9 ulps, so the results agree up to small differences which may affect the last $\lceil \log_2(4.9) \rceil = 3$ bits. These numbers are all based on double precision, where one ulp is 2^{-53}. This is all consistent with our expecta-

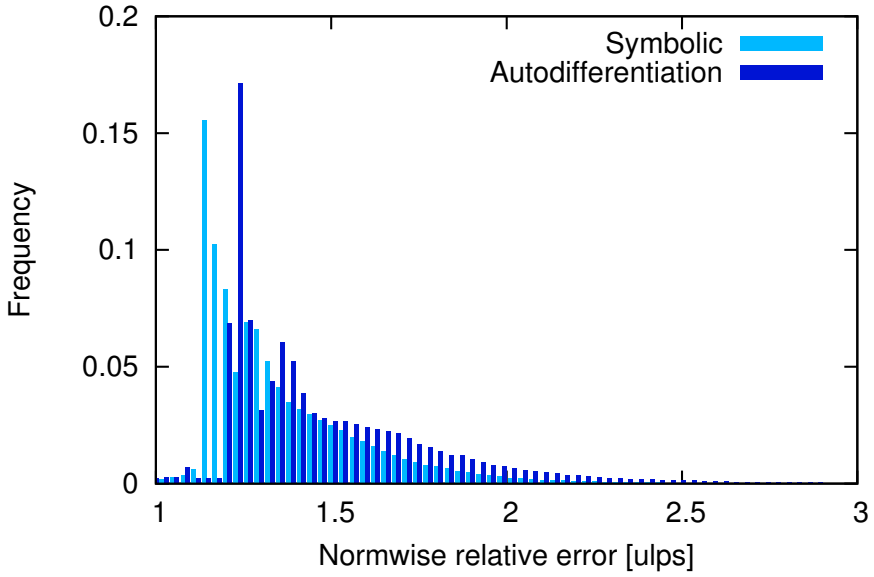


Figure 5.6: *The distribution of normwise relative errors when comparing the proposed method to an existing symbolic derivation and an implementation using automatic differentiation.*

tion that the results should be identical except for differences due to rounding. One thing worth noting is that the implementation based on automatic differentiation consistently has a higher error than the symbolic derivation. However, the difference is small enough that it is unlikely to be of practical significance.

5.8.3 Performance

Fundamentally, the flow of execution for the method proposed here is not dependent on the mesh configuration. We therefore expect that the performance gains are consistent across all the examples considered. Furthermore, we expect the cost to be proportional to the number of hinges. To verify this, we first consider the runtime for the cloth-sphere example as a function of the number of vertices. The results are shown in Figure 5.7. Each of the three implementations considered show near perfect linear scaling. However, other parts of the cloth simulation do not scale linearly with the number of vertices. Most notable is the linear solver, which in this case is PARDISO from Intel's MKL v. 11.0. With an estimated complexity of $O(n^{1.3})$, it is doing fairly well from a complexity point of view, but ultimately this is still bound to dominate any gain in the evaluation of the bending Hessian.

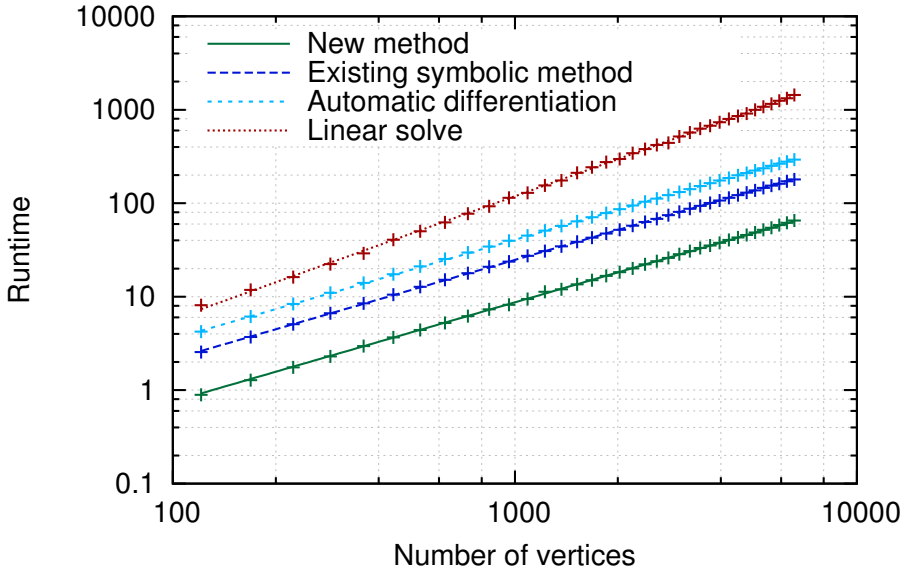


Figure 5.7: *The cost of evaluating bending forces and force gradients as well as assembling the associated stiffness matrix. Each of the three methods considered exhibit close to $O(n)$ complexity in the number of vertices. For comparison the cost of the linear solve is also shown. The estimated complexity for this phase is $O(n^{1.3})$.*

In addition to the linear solve, there are other costs, like collision detection and response, which depend on the particular simulation. Still, the fraction of the total time spent on bending force evaluation and stiffness matrix assembly can be significant as shown in Table 5.1.

Method	Cloth-sphere	Beams
Presented method	4–6%	9–12%
Existing symbolic	9–15%	17–23%
Autodifferentiation	14–22%	26–33%

Table 5.1: *The fraction of the total simulation time spent on bending force evaluation and the associated stiffness matrix assembly. These numbers are minimums and maximums over all the examples in each group.*

To get an accurate estimate of the speedup of the Hessian evaluation by itself, we model the runtime cost as a function of the number of vertices by $f(x) = ax^k$. We then estimate the two parameters, a and k, by linear regression of the associated function $g(\ln x) \equiv \ln f(x) = k \ln x + \ln a$. Based on an initial data analysis, we observed that, for each example the estimated exponent is the same for all the methods to within the statistical error of

the estimate. Hence, we have re-estimated $\ln a$ under the assumption that the value for k is the same for each method. Given the estimated values, the speedup is given as the ratio between the values of a for the different methods. The results are shown in Table 5.2.

Experiment	k	$\ln a_N$	$\ln a_S$	$\ln a_A$	Speedup over symbolic	Speedup over autodiff
Cloth-sphere	1.062	−5.175	−4.129	−3.632	2.85	4.68
Beam 1	1.079	−2.629	−1.829	−1.287	2.22	3.83
Beam 2	1.075	−3.305	−2.505	−1.963	2.23	3.83
Beam 3	1.079	−4.711	−3.909	−3.370	2.23	3.82
Beam 4	1.081	−6.467	−5.674	−5.134	2.21	3.79
Beam 5	1.079	−7.056	−6.261	−5.715	2.21	3.82

Table 5.2: *The results of the parameter estimation process for* $\ln f(x) = k \ln x + \ln a$*, where* x *is the number of vertices, and* $f(x)$ *is the runtime. The subscript "N" is used to denote the new method, "S" is used for the existing symbolic method, and "A" is used for the autodifferentiation method.*

It should be noted that, counter to expectation, there is a slight (but statistically significant) nonlinearity. This nonlinearity appears to be due to the sparse matrix data structure used for the assembly of the stiffness matrix.

What is measured and modeled above is the speedup for the combined evaluation of forces and force Jacobians plus the assembly of the stiffness matrix. The cost of the assembly operation is difficult to measure on its own in our codebase, but by using Intel's VTune, we estimated the assembly cost to be roughly 49% of the runtime for the new method, 25% for the existing symbolic method, and also 25% for the automatic differentiation method. Factoring this in, the speedup for the computation of the bending forces and their Jacobians on their own is 1.5× the numbers shown in Table 5.2. This gives a speedup between 3× and 4× relative to the existing symbolic method, which is remarkably close to the rough estimate provided at the end of Section 5.5.1. Compared to the automatic differentiation method the speedup can exceed 7×.

In addition to the comparisons above, we made a less extensive comparison to the approximate method employed by Baraff and Witkin [1998]. This comparison suggests that computing the exact derivatives using our method is no more expensive than computing the approximate derivates using their method.

5.9 Limitations

Hinge-based energies have a number of limitations that should be kept in mind in any implementation. These are not specific to the derivations in this paper, but rather inherent to the use of the hinge angle.

Degeneracies. The geometry depicted in Fig. 5.1 could degenerate (to first order) in two ways: *edge collapses*, where the hinge edge degenerates, and *altitude collapses*, where one of the vertices opposite the hinge edge becomes colinear with the hinge edge.

Either degeneracy causes at least one hinge altitude to vanish, leading to division by zero in the bend energy gradient and Hessian. In the case of an edge collapse, it is possible to show that all the relevant terms have well-defined limit values as the edge length decreases, so the division by zero can be handled by a numerically stable special case. By contrast, we do not know of a remedy for the altitude collapse. In practice, the membrane (stretch) energy of a shell should prevent any element from degenerating, but for constitutive models like Saint-Venant Kirchhoff which allow inversion, a collapse remains possible. However, neither type of degeneracy has occured in any of our experiments.

Indefiniteness of the hinge angle Hessian. In many situations, it is desirable for a matrix to be positive definite. In particular, this is a requirement for being able to use the conjugate gradient method or a Cholesky based direct method to solve the corresponding set of linear equations. However, for general mesh positions, the Hessian of the hinge angle is indefinite.

The fact that it is not positive definite is not at all surprising. As discussed in Chapter 3, strict convexity implies *uniqueness* of all solutions, which precludes bifurcation phenomena such as buckling [Hill, 1957; Ball, 1976].

CHAPTER 6

Unconstrained dynamics

In the previous chapters, we derived the hyperelastic energy functions to approximate the behavior of cloth and provided a spatial discretization. The focus of this chapter is the dynamic evolution in the absence of contacts. In the next chapter, we then augment this treatment by introducing contact and friction forces. The primary contribution here is to quantify the benefit of using an incremental potential for time integration and to provide some intuition as to why it is beneficial. We also point out some of the challenges when attempting to take large time steps.

6.1 Euler-Lagrange equations

To describe the dynamics of our system, we use Lagrangian mechanics as this facilitates the introduction of constraints later on. We therefore consider a system with a generalized coordinate representation given by $\mathbf{q} \in \mathbb{R}^n$, and a corresponding generalized velocity, $\dot{\mathbf{q}}$. The world-space position of a point $\boldsymbol{x} \in \mathbb{R}^3$ is assumed to be given by a function $\boldsymbol{x}(\mathbf{q})$, and the point velocity is obtained through the chain rule as $\dot{\boldsymbol{x}} = (\nabla_{\mathbf{q}} \boldsymbol{x})\dot{\mathbf{q}}$.

We refer to $\mathbf{q}$ as a *configuration*, which in our case typically consists of the positions of all the vertices stacked together. However, it extends relatively easily to rigid body dynamics by letting the generalized coordinates for a single rigid body consist of the location of the center of mass and its current rotation around the center of mass (the latter typically being represented by a quaternion). See Witkin and Baraff [1997] for details.

Associated with the generalized coordinates is a set of generalized (or conjugate) momenta, which we will denote by $\mathbf{p}$. For deformable objects such a cloth and hair, which are discretized by a finite element method we have $\mathbf{p} = \boldsymbol{M}\dot{\mathbf{q}}$, where $\boldsymbol{M}$ is the mass matrix associated with the discretization. For a rigid body, the generalized momentum includes the linear and angular momentum, and the (generalized) mass matrix consists of the mass of the object and its inertia tensor.

The configuration in general consists of both deformable and kinematic variables. At times, it will be convenient to distinguish between these two, in which case we partition $\mathbf{q} = \begin{bmatrix}(\mathbf{q}^f)^T & (\mathbf{q}^s)^T\end{bmatrix}^T$ into n^f free (deformable) and n^s scripted (kinematic) variables. Other quantities may be partitioned in a similar way and denoted with superscripts f and s.

Based on the above definitions, we can compute the Lagrangian, $\mathcal{L} = W_{kin} - W_{pot}$, where W_{kin} is the kinetic energy of the system and W_{pot} is the potential energy of the system. In practice, a dissipation potential is usually also needed to account for non-conservative forces in the system. Not all dissipative forces can be written as a potential, but viscous damping can relatively easily be included. However, for brevity, we omit further discussion of that here. In our case we therefore have :

$$\begin{aligned} W_{kin} &= \tfrac{1}{2}\dot{\mathbf{q}}^T\boldsymbol{M}\dot{\mathbf{q}}\,, \\ W_{pot} &= W_{elastic}(\mathbf{q}) + W_{gravity}(\mathbf{q})\,. \end{aligned}$$

Defining the *action* as the integral of $\mathcal{L}$ over time, Hamilton's principle of stationary action implicitly defines the evolution of the dynamic system. Applying the Euler-Lagrange equations from variational calculus to the action leads to the Lagrangian equations for unconstrained motion :

$$\frac{d}{dt}\left(\frac{\partial\mathcal{L}}{\partial\dot{\mathbf{q}}_i}\right) - \frac{\partial\mathcal{L}}{\partial\mathbf{q}_i} = 0, \quad 1 \le i \le n^f\,.$$

With the above definition of $\mathcal{L}$, this is equivalent to the following equation, which amounts to Newton's second law :

$$\boldsymbol{M}\ddot{\mathbf{q}} + \nabla W_{pot} = 0 \tag{EL}$$

Using the generalized momenta and assuming a constant mass matrix[1], we get the corresponding Hamiltonian form of the equations :

$$\begin{aligned} \boldsymbol{M}\dot{\mathbf{q}} &= \mathbf{p}\,, \\ \dot{\mathbf{p}} &= -\nabla W_{pot}\,. \end{aligned}$$

[1]This is generally not the case for rigid body dynamics, where the inertia terms change over time.

The above derivation gets more complicated for rigid bodies because the representation of the rotation as a quaternion has to be accompanied by a constraint to ensure unit length of the quaternion, [Betsch and Siebert, 2009]. Since the focus here is on deformable objects, we won't go into the details of this derivation, but just mention that rigid bodies fit within this framework.

6.2 Time discretization

Given the equations for the dynamics, we need to integrate them in time to obtain the temporal evolution of the system. There are a plethora of methods available for doing this with a variety of properties. For an extensive discussion of many of these, we refer to [Hairer et al., 2004; Hairer and Wanner, 2004; Hairer et al., 2006]. In this thesis, we use the backward Euler scheme for all our examples. This is the simplest example of the Runge-Kutta implicit schemes, which can also be interpreted as a member of the Adams–Moulton family of methods or alternatively as a backward differentiation scheme. In the latter case, it is known as the BDF-1 method. It is a popular scheme in graphics applications because it is unconditionally stable for linear problems and tends to be stable for nonlinear problems (although no formal guarantee for this exists). However, it suffers from significant amounts of numerical dissipation. While this is not ideal, damped behavior is frequently desired, so this is often acceptable.

One reason for using such a simple scheme is that we need to handle collisions. Higher-order schemes, which are typically much more accurate, require the functions being integrated to be smooth. However, with collisions in the system, we know that such smoothness won't exist. As we discuss in Section 7.9, the backward Euler scheme also has the benefit of effectively modeling all collisions as being inelastic, which is a reasonable approximation for cloth and hair.

Based on the discussion in the previous section, we introduce the *state vector*, $\mathbf{y} \equiv (\mathbf{q}^T, \mathbf{p}^T)^T$, and denote the corresponding *flow function* that describes the evolution of the dynamic system by $\varphi(t, \mathbf{y})$. We then have

$$\dot{\mathbf{y}} = \varphi(t, \mathbf{y}) = \begin{bmatrix} \boldsymbol{M}^{-1}\mathbf{p} \\ -\nabla W_{pot}(\mathbf{q}) \end{bmatrix}$$

The backward Euler scheme approximates the solution of this by

$$\mathbf{y}_{k+1} = \mathbf{y}_k + \tau\varphi(t_{k+1}, \mathbf{y}_{k+1}), \quad \forall k \geq 0\,, \tag{BE}$$

where $\mathbf{y}_0$ is a given initial state, subscripts indicate the discrete time step, and τ is the size of a time step.

Expanding $\mathbf{y}$ and φ gives

$$\begin{bmatrix}\mathbf{q}_{k+1}\\ \mathbf{p}_{k+1}\end{bmatrix} = \begin{bmatrix}\mathbf{q}_k\\ \mathbf{p}_k\end{bmatrix} + \tau \begin{bmatrix}\boldsymbol{M}^{-1}\mathbf{p}_{k+1}\\ -\nabla W_{pot}(\mathbf{q}_{k+1})\end{bmatrix}$$

Let the update to the configuration for one time step be defined as

$$\delta \equiv \mathbf{q}_{k+1} - \mathbf{q}_k = \tau \boldsymbol{M}^{-1}\mathbf{p}_{k+1}.$$

Substituting the first of these equations into the second then gives

$$\mathbf{p}_{k+1} = \mathbf{p}_k - \tau \nabla W_{pot}(\mathbf{q}_k + \delta)\ .$$

Premultiplication on both sides by τ transforms it into

$$\boldsymbol{M}\delta = \tau \mathbf{p}_k - \tau^2 \nabla W_{pot}(\mathbf{q}_k + \delta)$$

or, equivalently

$$\boldsymbol{M}\delta = \tau \boldsymbol{M}\dot{\mathbf{q}}_k - \tau^2 \nabla W_{pot}(\mathbf{q}_k + \delta) \qquad \text{(DEL)}$$

We refer to this equation as the *Discrete Euler-Lagrange equation* or DEL for short, and the solution to this equation is denoted by δ_u, where 'u' is for "unconstrained".

6.3 Root finding

The discrete equations obtained from the backward Euler scheme are in general nonlinear since W_{pot} is a nonlinear function of $\mathbf{q}$. There are a number of potential methods for solving such nonlinear equations. Many of them are based on variations of Newton's method using the norm of the residual as the merit function, [Nocedal and Wright, 2006, Chapter 11] and [Press et al., 2007, Chapter 9]. However, nonlinear root-finding for multivariate functions is notoriously a difficult problem. As an approximation, many papers in the graphics literature only consider a single iteration in Newton's method. This includes Baraff and Witkin [1998].

When writing the equations for Newton's method, one can start with Eq. (BE) and treat the entire state vector, $\mathbf{y}$, as the unknown. In this case, an initial guess has to be provided for *both* the position (configuration) and the velocity (momentum). This is the approach taken by Baraff and Witkin

[1998], where the initial guess amounts to no change in position *and* no change in velocity. One can argue that this is only a reasonable guess if the object is at rest, which is often not the case. However, in practice, it works surprisingly well.

An alternative approach is to apply Newton's method to the DEL equations and treat δ as the unknown. In this case, an initial guess is only required for δ, which indirectly establishes an initial guess for the velocity. More specifically, from Eq. (DEL), define the multivariate and nonlinear residual function, f, for which we want to find the roots :

$$f(\delta) \equiv \boldsymbol{M}\delta - \tau \boldsymbol{M}\dot{\mathbf{q}}_k + \tau^2 \nabla W_{pot}(\mathbf{q}_k + \delta)$$

Newton's method gives the i'th increment to the δ as

$$\nabla f\left(\delta^{(i)}\right) d\delta^{(i)} = -f\left(\delta^{(i)}\right) , \tag{6.1}$$

where superscripts in parentheses indicate iterates and $d\delta^{(i)} \equiv \delta^{(i+1)} - \delta^{(i)}$. Concretely, this means that

$$\begin{aligned}\left(\boldsymbol{M} + \tau^2 \boldsymbol{H}_{pot}(\mathbf{q}_k + \delta^{(i)})\right) d\delta^{(i)} = \\ -\boldsymbol{M}\delta^{(i)} + \tau \boldsymbol{M}\dot{\mathbf{q}}_k - \tau^2 \nabla W_{pot}\left(\mathbf{q}_k + \delta^{(i)}\right)\end{aligned} , \tag{6.2}$$

where $\boldsymbol{H}_{pot}$ is the Hessian of the potential energy. For a zero initial guess, $\delta^{(0)} = 0$, and only one iteration, this gives the *linearly implicit Euler* method[2] :

$$\left(\boldsymbol{M} + \tau^2 \boldsymbol{H}_{pot}(\mathbf{q}_k)\right) \delta = \tau \boldsymbol{M}\dot{\mathbf{q}}_k - \tau^2 \nabla W_{pot}\left(\mathbf{q}_k\right) \tag{LIE}$$

While this is akin to the linearization done by Baraff and Witkin [1998], it should be emphasized that it is *not* the same linearization due to the difference in the initial guess.

It should be noted that since W_{pot} in general is non-convex (as discussed in previous chapters), it follows that $\boldsymbol{H}_{pot}$ may be indefinite. As such, the standard Newton method described here has no convergence guarantee. One way to address this is to add either a line search, a trust region, or a similar technique. We discuss this further in Section 6.7.

[2]This is sometimes also referred to as the *semi-implicit Euler* method, but this term is also being used for the symplectic Euler method, so we avoid this terminology to avoid confusion.

6.4 Incremental potential

As an alternative to solving a set of nonlinear equations, we first restate Eq. (DEL) directly in terms of $\mathbf{q}_{k+1}$:

$$\boldsymbol{M}(\mathbf{q}_{k+1} - \mathbf{q}_k) = \tau \boldsymbol{M}\dot{\mathbf{q}}_k - \tau^2 \nabla W_{pot}(\mathbf{q}_{k+1})$$

Notice then that it can formally be integrated with respect to $\mathbf{q}_{k+1}$, leading to the *incremental potential* :

$$W_{inc}(\mathbf{q}) = \tfrac{1}{2}\mathbf{q}^T \boldsymbol{M}\mathbf{q} - \mathbf{q}^T \boldsymbol{M}(\mathbf{q}_k + \tau\dot{\mathbf{q}}_k) + \tau^2 W_{pot}(\mathbf{q})$$

Given this definition, the discrete Euler-Lagrange equations can simply be recovered as

$$\nabla W_{inc}(\mathbf{q}_{k+1}) = 0$$

Thus, the solution to DEL is given by the stationary points of the incremental potential, which means that the root finding problem can be turned into a minimization problem :

$$\mathbf{q}_{k+1} = \operatorname*{argmin}_{\mathbf{q}} W_{inc}(\mathbf{q})$$

The incremental potential was introduced by Radovitzky and Ortiz [1999] and later in graphics by Kharevych et al. [2006]. More recently, it is essentially also the idea being used in [Gast and Schroeder, 2014].

It is widely stated that minimization problems are easier than root finding problems, but the reason for this is often omitted. One exception is [Press et al., 2007, Section 9.6] where they start by stating "There are *no* good, general methods for solving systems of more than one nonlinear equation". Further, they explain that what makes optimization easier is that the components of a gradient vector are not independent functions, but related by integrability conditions that are highly restrictive. Thus, while you can always find a minimum by sliding downhill on a single surface, there is no analogous conceptual procedure for finding a multidimensional root.

Still, both problems are nonlinear, and both problems are typically solved by some variation of Newton's method. In practice, the implementations even end up looking very similar with some kind of line search or trust region strategy (one such example is the method proposed in [Press et al., 2007, Section 9.7]). They differ, however, by the choice of *merit function* used to evaluate progress, which is crucial.

For the minimization problem the merit function is simply $W_{inc}(\mathbf{q}_k + \delta)$, while for the root finding problem, it is typically the norm of the nonlinear residual, $\|f(\delta)\|^2$. The problem is that a reduction of one does not imply a reduction of the other, and vice versa.

It should be noted that Newton's method applied to the incremental potential leads to the same equation as in Eq. (6.1), so the *exact* solution for the Newton step in the minimization problem is the same as for the root finding approach. When the linear system is solved exactly, and a full step is taken, there is therefore *no* difference between the two approaches. The difference shows up when we are no longer considering the exact system and/or taking a full step. In particular, this comes up with *modified Newton methods* needed for non-convex problems, (i.e., when ∇f is indefinite). For these methods the resulting search directions have to be evaluated based on the merit function.

To illustrate why this can be problematic, consider a simple 1D example[3] where the elastic energy is given by $W(x) = x^4 - 6x^2 - 72x$ as shown in Fig. 6.1. The corresponding elastic force is $-\nabla W(x) = -4x^3 + 6x - 72$, which is zero for $x = 3$ corresponding to a minimum in $W(x)$. In this case the "residual" is simply the magnitude of the force function, so root finding will use $\|\nabla W(x)\|^2$ to judge the quality of a given step. If we start with an initial guess of $x = 0$ and a proposed search direction $\Delta x = 1$ then this will represent a descent direction for $W(x)$, but it is *not* a descent direction for $\|\nabla W(x)\|^2$. In fact, no step length in that direction will give a reduction in the residual. On the other hand, $\Delta x = -1$ will represent a descent direction for $\|\nabla W(x)\|^2$ but not for $W(x)$.

Even when ∇f is positive definite, minimization is often still advantageous; especially when Newton's method is combined with iterative linear solvers. Most Newton-Krylov solvers, for example, are based on *inexact Newton methods*, [Dembo et al., 1982], but these methods require an evaluation of whether the current (approximate) search direction is a good (enough) direction or not. In the context of inexact Newton methods, Nash and Sofer [1990] noted that evaluating the search direction based on the norm of the residual can be arbitrarily bad in terms of reducing the true objective function. By comparison, when performing a minimization, it is straight forward to evaluate if a given search direction reduces the objective function or not.

On top of this, it should be noted that since the differential operator, ∇, amplifies high frequencies, it follows that $\|\nabla W_{inc}\|$ in general will be much more oscillatory than W_{inc}. Thus, it will have more local minima compared

[3]This example is courtesy of Andy Milne, Walt Disney Animation Studios.

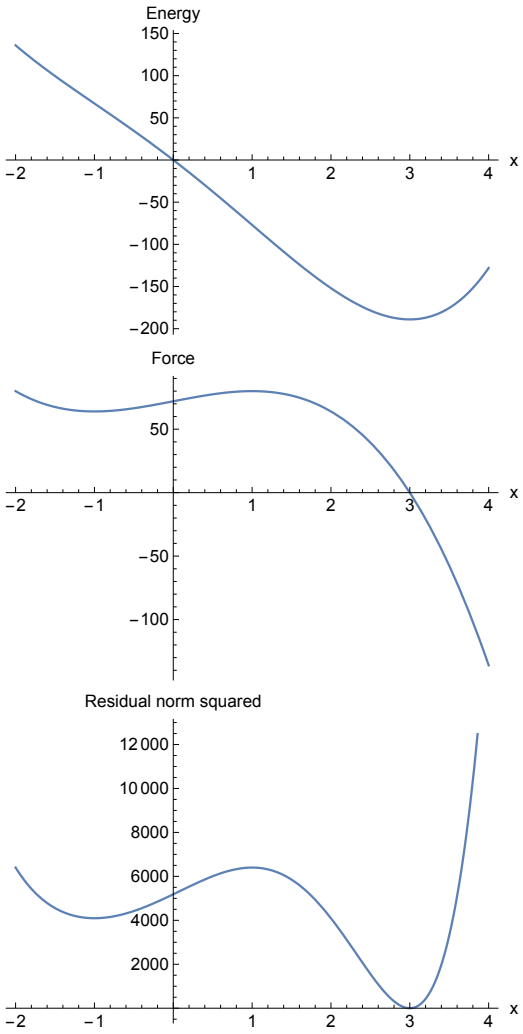


Figure 6.1: *A simple (nonconvex) energy function (top), the corresponding force function (middle), and the norm squared of the nonlinear residual function (bottom).*

to W_{inc}, which means that trust regions will have to be smaller and line searches will have to take smaller steps.

6.5 Integrating a 3D hinge

To illustrate the difference between using root-finding and the incremental potential for performing one unconstrained time step, consider the simple example of a single hinge based on an isotropic St. Venant-Kirchhoff membrane energy (see Chapter 4) and the discrete shell bending energy (see Chapter 5). The hinge consists of two adjacent triangles with a flat rest con-

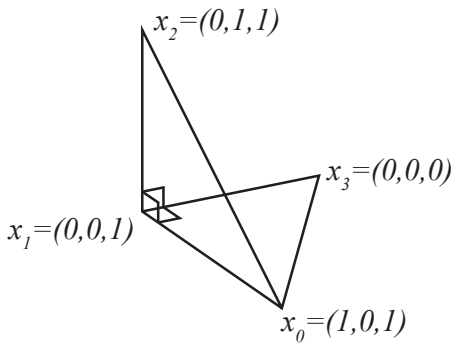


Figure 6.2: *The initial configuration of a simple hinge consisting of two triangles. The positions of the vertices are given in centimeters such that the combined area of the hinge is* 1 cm^2. *In the corresponding flat rest configuration the position of* x_2 *is* $(0, 0, 2)$.

figuration and an initially deformed configuration, where the hinge is bent 90 degrees (see Fig. 6.2). The material parameters for this experiment are :

- Thickness : 1 mm
- Young's modulus : 100 MPa
- Poisson's ratio : 0
- Mass density : 0.1 kg/m^2

These values are somewhat representative of typical cloth, although the exact values do vary quite a bit between different types of cloth.

Given the above setup, we proceed to compute the updated configuration after one time step based on both the root finding approach and the incremental potential. To ensure equal levels of sophistication in the two methods, both are implemented in Mathematica using the built-in functions for root finding (FindRoot) and function minimization (FindMinimum). FindRoot is being provided an analytically derived Jacobian of f, while FindMinimum is being provided an analytically derived gradient and the Hessian of W_{inc}. Since the gradient of W_{inc} is exactly f by construction, the two functions effectively have the same amount of information to work with.

For each setup, the dynamics is being solved for three different time steps : $\tau \in \{\frac{1s}{240}, \frac{5s}{240}, \frac{10s}{240}\}$. The largest of these corresponds to a full animation frame at current frame rates.

The convergence behavior for the two setups is shown in Fig. 6.3. Qualitatively, they are similar in that they have an early phase of fast convergence, followed by a phase of stagnation, before finally converging to numerical

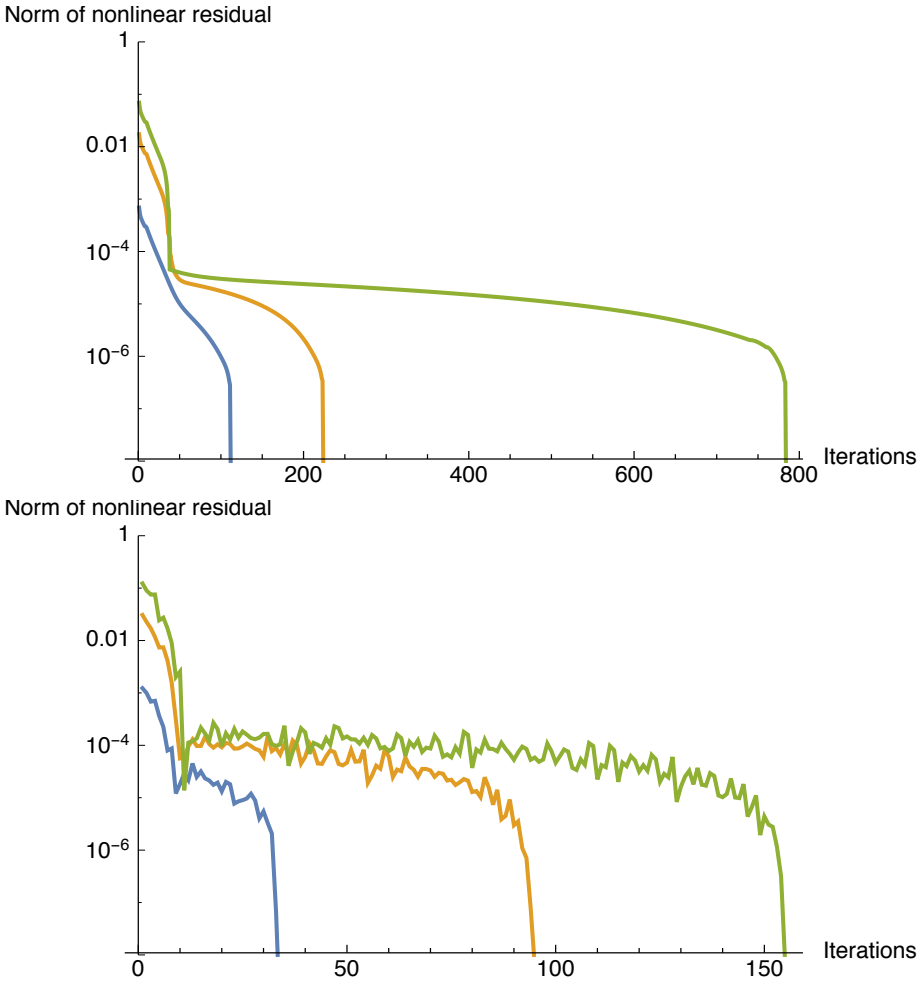


Figure 6.3: *The norm of the residual when using the FindRoot function for three different time steps (top) compared to the incremental potential deviation from the optimal value when using the FindMinimum function (bottom).*

precision. The graph for the root finding approach is smooth because it works directly with the norm of the nonlinear residual. The graph for the minimization approach, on the other hand, works with the incremental potential, so there is no guarantee that the nonlinear residual will behave in a monotonic way as is obvious here. Both graphs show the nonlinear residual to make comparisons easy.

Aside from smoothness, the most notable difference is that the optimization based approach requires $2-5\times$ fewer iterations than the approach based on root finding. This difference is largest for the largest time step, which suggests that large time steps are especially problematic for the root finding approach. Given that the cost of each iteration should be roughly equal in the two approaches, the differences translate into a significant saving when

using the incremental potential. A similar saving is obtained if the system is only solved to the first "knee" in the convergence curves, which is likely to be more typical of what would be done in an animation system.

What is not immediately obvious from the graphs is that the FindRoot function is numerically less stable. To produce the above graph, it was necessary to increase the internal working precision used by Mathematica for FindRoot above the default IEEE double precision. In practice, 50 digits were used, but this number quickly had to be increased for larger time steps or lower masses. However, for the time steps shown here, a similar increase in the working precision was not necessary for FindMinimum.

6.6 Ill-conditioning

The example introduced in the previous section is deceptively difficult because the matrix $\boldsymbol{H}_{pot}$ is singular by construction. In particular, since the elastic energy and forces are invariant under translations, it follows that $\boldsymbol{H}_{pot}$ has three eigenvalues that are zero.[4]

In practice, we have to solve problems with $\boldsymbol{M} + \tau^2 \boldsymbol{H}_{pot}(\mathbf{q}_k)$, so the mass matrix acts a regularizer, which helps. However, as the size of the time step increases, the relative contribution from the mass matrix decreases, and the resulting matrix becomes increasingly ill-conditioned until it is singular in the limit of solving a quasi-static problem. It is this near-null space that is responsible for the "knee" in the curves in Fig. 6.3. The change prior to the knee corresponds (roughly) to a deformation to get back to a flat rest state, while the change after the knee ensures that the inertial component of the motion is correct. Both of these are in fact important for a correct solution, so while it is tempting to stop when the solver stagnates, this will technically yield an incorrect solution.

In the presence of boundary conditions, the null-space of the system changes, which can help, but for cloth simulation it is quite common to have systems where the only boundary conditions are those provided indirectly through collision constraints. For hair simulations on the other hand, the root of the hair is typically constrained to the scalp, which eliminates the singularity of $\boldsymbol{H}_{pot}$.

It should be noted that the ill-conditioning is primarily a function of the time step size. In fact while changing the thickness of the cloth changes the

[4] Note that while the energy is also invariant under rotations, the forces are not since they rotate. Rotations are therefore in the near null-space rather than in the true null-space.

magnitude of the smallest non-zero eigenvalue of $\boldsymbol{H}_{pot}$, it does not change the fact that $\boldsymbol{H}_{pot}$ is singular. The same can be said about the choice of the constitutive parameters. The problem, however, is more pronounced for cloth and hair simulations than say for solid elasticity simply because the objects involved typically have less mass.

In summary, we note that, for large time steps, we will always be dealing with a nearly singular system at every Newton step, including at the solution. We also note that this problem is inherent to *any* elastic energy with translational invariance. In other words, it is not specific to any particular choice of constitutive model.

As a consequence, we are forced either to take smaller time steps or to design a method that can handle nearly singular systems gracefully. As shown by Griewank and Osborne [1983], the standard Newton method is at best linearly convergent near a singular solution, which suggests that more sophisticated methods are required.

One class of methods that has been shown to work well for ill-conditioned systems is so-called tensor methods, where the quadratic approximation used in Newton's method is replaced by a quartic model, [Schnabel and Chow, 1991; Bouaricha, 1997; Bader and Schnabel, 2007].

Another class of methods is based on cubic regularization as first proposed by Nesterov and Polyak [2006], and later developed further by [Cartis et al., 2011a,b; Gould et al., 2012]. However, while these methods have favorable theoretical guarantees in terms of the worst case number of function evaluations, it is not clear that they are competitive with other methods in practice.

6.7 Non-convex optimization

As previously mentioned, the elastic energy is in general non-convex. Independent of the ill-conditioning discussed above, a robust solver must therefore also be able to handle non-convex problems. However, writing a generic robust solver for a non-convex optimization problem is a non-trivial task; even in the unconstrained case. In this section, we discuss some explorations and observations related to this problem, but ultimately there is still room for additional work.

A common way in graphics of dealing with the non-convexity is by modifying $\boldsymbol{H}_{pot}$ at the time of construction to ensure that it is always positive (semi)-definite. Unfortunately, such modifications make it difficult to construct the

incremental potential since there are no guarantees that the modified Hessian corresponds to an energy. Furthermore, since those modifications are typically made to preserve translational invariance, the resulting systems remain ill-conditioned for large time steps.

As an alternative to modifying the Hessian of each element when assembling the stiffness matrix, we can consider more general methods for non-convex optimization. This typically involves adding either a line search or a trust-region to Newton's method. However, line search methods can be problematic due to the fact that we also have a nearly singular problem. In particular, the nearly singular matrix can generate search directions with very large magnitudes. When this happens, a tiny step may be required along the search direction. Unfortunately, this easily leads to catastrophic cancellation in the numerical computations. In particular, we encountered this when implementing Newton's method with a basic back-tracking line search. It is also a problem for the widely used (and more sophisticated) line search method proposed by Moré and Thuente [1994] (used in Mathematica among many other applications). In fact we conjecture that this is one of the reasons we had to increase the internal precision in order to generate Fig. 6.3.

To address this cancellation problem Hager and Zhang [2005] proposed a different line search method that effectively replaces the problematic subtraction by an evaluation of the derivative along the line search direction. Unfortunately, their paper is incomplete in the sense that a number of subtle details necessary for implementing the method are missing. However, the method is implemented as part of the CG_DESCENT algorithm, [Hager and Zhang, 2006], for which the authors provide source code. The CG_DESCENT method is fundamentally a nonlinear conjugate gradient method, which is expected to have slower convergence than Newton's method close to a local (non-degenerate) minimum, but it has the advantage that only function and gradient evaluations (of the incremental potential) are necessary.

To evaluate the method, the implementation by the authors was incorporated into a plug-in for Mathematica for this thesis. This allowed it to be used in the same way as Mathematica's built-in function FindMinimum. Given the source code, it was also possible to look specifically at the line search, but it was clear that there were numerous changes compared to the original paper, and it was not entirely clear exactly what the logic was. In the end, CG_DESCENT was therefore just evaluated as a black box.

The result of applying CG_DESCENT to the hinge problem in Section 6.5 is shown in Fig. 6.4. For this specific plot, the tolerance for the gradient of the objective function (i.e., the gradient of the incremental potential, which is f) was set to 10^{-8}. However, the solver seems to get stuck with a residual

around 10^{-6} and, given the erratic behavior, it seems likely that it is encountering some other problems due to floating point arithmetic. It is possible that a deeper understanding of the CG_DESCENT code could unravel the source of this problem, but this has not been done.

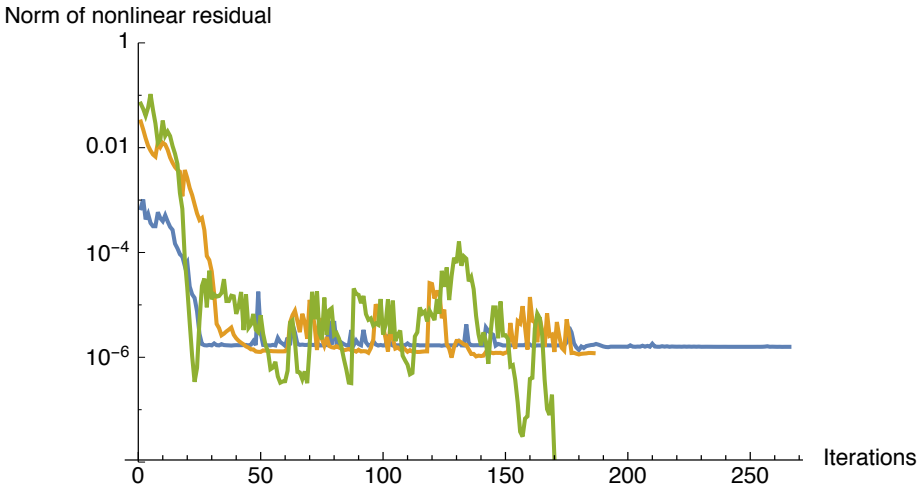


Figure 6.4: *The norm of the nonlinear residual when using the CG_DESCENT function for three different time steps.*

Another method considered for the problem at hand is the "Inertia Controlling Modified Conjugate Gradient" (ICMCG) method proposed by Zhou et al. [2009]. As the name implies, it is also a nonlinear conjugate gradient method, but with some modifications. In particular, it evaluates the full Hessian to ensure optimal convergence rate, then modifies it (if necessary) to guarantee that a good search direction is obtained, and finally combines it with a line-search to get global convergence. The modification effectively introduces an implicit trust-region, so this method is really a hybrid between conjugate gradient and Newton's method on one hand, and line-search vs. trust-region on the other hand.

Similar to CG_DESCENT, the paper leaves out some subtle implementation details. In particular, since it has the flavor of an inexact Newton method, [Dembo et al., 1982], it requires a forcing sequence, but the choice of this sequence is not spelled out in the paper. For implementation purposes, we used the one discussed by An et al. [2007]. Regardless of this choice, the method as presented suffers from floating point problems when evaluating the stopping criterion near the solution due to cancellation. However, the bigger problem is that the conjugate gradient method suffers from floating point precision issues for very ill-conditioned systems, Meurant [2006]; even if they are otherwise positive definite. In particular, after constructing the Krylov space corresponding to the eigenvalues that are not close to zero, the

construction of subsequent search directions can easily fail to be conjugate to previous search directions. This in turn leads many of the invariants in the CG method to fail at which point the behavior is unpredictable. At this point this method can therefore not be said to be robust. However, this failure has more to do with the ill-conditioning of the problem than non-convexity.

6.8 Summary

Originally the work in Chapter 4 and Chapter 5 was motivated by problems observed when trying to take large time steps in cloth simulations. This lead to a suspicion that something in the underlying cloth model was leading to instabilities when subjected to large deformations. At the time there were multiple potential problems. These included the fact that existing models would assume small deformations and flat rest configurations, they would approximate the force gradients, and most of them would introduce modifications to ensure symmetric positive definite matrices.

In the preceding chapters we have addressed all those concerns such that the model considered at this point is well suited for large deformation and given the choice of discretization, the force gradients are exact and consistent with the non-convexities of the underlying hyperelastic energy functions. Yet, the problem largely remains, so we must conclude that the fundamental problem most likely stems from none of these issues. Rather the real problem is that the nonlinear problem we are trying to solve when taking large time steps is inherently ill-conditioned. The non-convexity is certainly something that needs to be taken into account, but it is the ill-conditioning that has ultimately broken all the attempted solvers.

Many papers talk about thin shell problems being ill-conditioned due to the small thickness of the shell relative to the lateral extent. However, it is important to realize that this is not the problem here. Nor does it have to do with badly shaped elements in the discretization. Instead it is the fact that the energy and forces are translationally invariant in the absence of any constraints that causes problems.

In addition to this key observation we have also seen concretely how much can be gained from a variational treatment using the incremental potential compared to a more standard rootfinding approach. Moreover, it has become clear that the optimization routines tend to be numerically much more resilient in the presence of ill-conditioning than the corresponding root finding algorithms.

The design and application of a truly robust method for handling non-convex and ill-conditioned problems remains future work, but we feel that the insights gained here constitute an important step in the right direction. For the remainder of this thesis, however, we note that all the problems discussed in this chapter obviously can be avoided by taking small(er) time steps.

CHAPTER 7

Contact modeling

Having dealt with unconstrained dynamics in the previous chapter, we now turn to constrained dynamics. There are many ways of handling collision response, but not all of them are numerically stable. A nice concise overview of different methods for addressing contact in graphics can be found in [Bertails-Descoubes et al., 2011]. Here, we focus on implicit constraint-based methods in maximal coordinates.

As we saw in the previous chapter, the singularity of the hyperelastic energy Hessian effectively poses a limitation on the size of stable time steps in an unconstrained simulation. In this chapter, we start with a didactic example to show that collision response can pose an even more severe limitation if not handled correctly. We then proceed to establish the model we use for handling contact between two objects in the following chapters. This includes contact forces to avoid interpenetration and friction forces to model both static and dynamic Coulomb friction.

7.1 Motivation

Consider the collision of a discretized three-node rod or piece of cloth in 2D as it hits a wedge (see Fig. 7.1). The standard approach to prevent intersections is to apply a position-based correction that enforces zero-displacement along a collision normal [Baraff, 1989; Mirtich and Canny, 1995; Bridson et al., 2002]. However, by resolving collisions in this way, we ignore changes in internal energy, effectively applying an instantaneous impulse response,

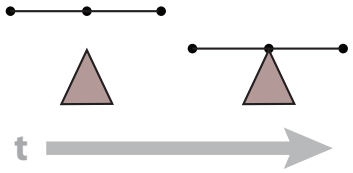


Figure 7.1: *A simple three-node rod hitting a wedge.*

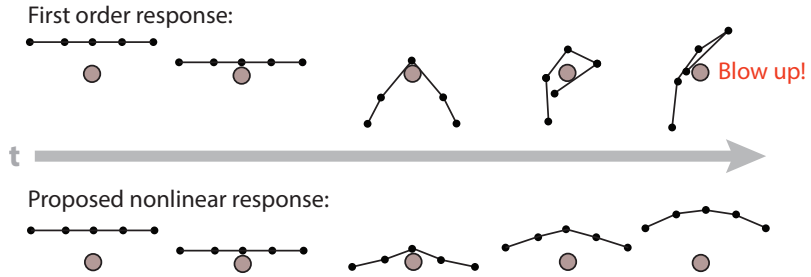


Figure 7.2: *Left to right in time. Top: a collision resolved by first-order–modeled response remains physics-oblivious; the resulting correction generates a large, localized, non-physical deformation leading to instability. Bottom: adaptive nonlinear response obtains a stable, global response.*

[Moreau, 1988]. As a consequence large, highly localized, non-physical deformations are generated.

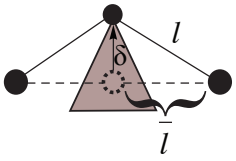
In fact, notice that stretching is dominant in thin materials. As we have seen in previous chapters, the stretch stiffness scales linearly with the thickness of the object, h, while the bend stiffness scales with h^3. Given the thickness of cloth (< 1mm) and hair (≈ 0.1mm), the stretch stiffness is therefore many orders of magnitude greater than the bending stiffness. While we expect these thin materials to potentially bend a great deal, we clearly do not wish to exercise their stretching modes unintentionally. Yet, this is exactly what the standard approaches do.

To build this missing "physical awareness" into collision response, first-order models have been developed. *Linearly compliant* collision response methods as in [Baraff and Witkin, 1998; Duriez et al., 2004, 2006; Kaufman et al., 2008; Otaduy et al., 2009; Allard et al., 2010; Zheng and James, 2011; Bertails-Descoubes et al., 2011; Daviet et al., 2011] apply *compliance* (inverse stiffness) to inexpensively communicate collision-induced strain across force stencils, effectively filtering collision response to obtain smoother post-collision configurations. Linearly compliant methods thus can balance between the difficult tradeoffs of stability and computational efficiency.

Unfortunately, when the unknowns in the discretization are the positions of the vertices (i.e., when using maximal coordinates), we face a severe limita-

tion. This is one of the key insights in this thesis : *first-order models are not sufficient to capture the physical response for thin elastic materials in maximal coordinates*. In particular, the collision response of thin materials such as hair and cloth is strongly nonlinear, which means that first-order collision modeling can generate the same problematic localized collision response as an impulsive model.

Consider the simple case of two adjacent elements, with little or no initial bending, colliding with an edge as in Fig. 7.2. In these configurations, when we apply a first-order, linearly compliant collision method, the resulting initial collision response varies little (to not at all) from the standard physics-oblivious method.



To understand why this occurs, reconsider the collision of our three-node rod in a straight, unstretched configuration with edge lengths $\bar{l}$ and l at undeformed and current configurations respectively. We now impose a normal displacement of δ due to a collision. Our first observation is that such displacements in the normal direction are *entirely* unaccounted for by expansions of stretching force up to second-order in this configuration. To see this, note that the Green-Lagrange strain for a one-dimensional rod[1] is quadratic in normal displacement :

$$\boldsymbol{E} = \tfrac{1}{2}(l^2/\bar{l}^2 - 1) = \tfrac{1}{2}\delta^2/\bar{l}^2$$

The stretching energy as a function of the normal displacement is therefore a quartic monomial :

$$W_{stretch} = \tfrac{1}{2}k\boldsymbol{E}^2 = \tfrac{1}{4}k\delta^4/\bar{l}^4.$$

It follows that the stretch force is a cubic monomial in δ, so a second-order approximation will be exactly zero.

Alternately, consider the stretching energy used in rod and mass-spring models, [Spillmann and Teschner, 2007; Bergou et al., 2008, 2010] :

$$W_{stretch} = \tfrac{1}{2}k(l/\bar{l} - 1)^2 = \tfrac{1}{2}k(\sqrt{\delta^2 + \bar{l}^2}/\bar{l} - 1)^2$$

Although the stretching energy is no longer a quartic monomial in δ, at an unstretched configuration, the force remains oblivious to normal displacements up to second-order (see Fig. 7.3, left).

[1] Let $\boldsymbol{F} = \boldsymbol{RU}$ be the polar decomposition of the deformation gradient, where $\boldsymbol{R}$ is a proper orthogonal tensor and $\boldsymbol{U}$ is the right stretch tensor. It then follows that $\boldsymbol{E} = \frac{1}{2}(\boldsymbol{U}^2 - \boldsymbol{I})$. In the 2D example here, the stretch factor is simply $l/\bar{l}$.

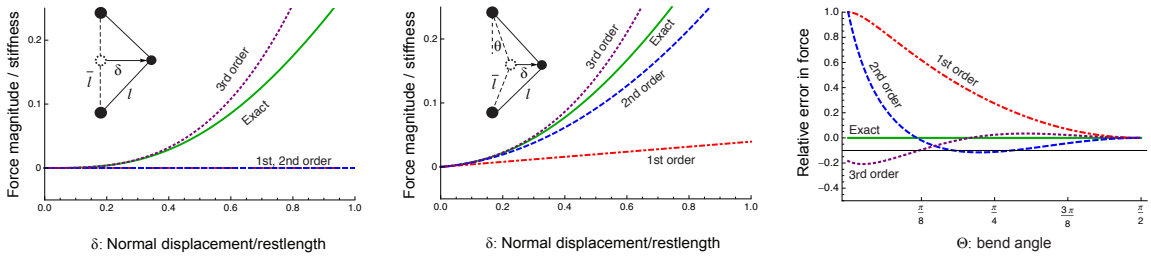


Figure 7.3: *We plot the magnitude of rod/mass-spring stretching forces modeled by respectively first-, second-, and third-order force approximations for expansions about a straight (left) and a bent (middle) configuration, compared against the ground truth stretching force evaluation. In both cases, we observe that first-order modeling with respect to normal displacement δ underestimates the force while, in the case of straight configurations (left), any expansion less than third-order entirely ignores normal displacement. On the other hand, by plotting the relative error of these approximations with respect to the ground truth force evaluation at a fixed $\delta = \frac{1}{2}$, we see that, as bending increases, lower-order models give a correspondingly better approximation (right).*

This problem is not restricted to collision forces; frictional forces also lead to the same behavior. This follows from rotating our perspective to see that the same geometry we considered above for collisions holds for friction as well: friction forces are applied along a contact's tangent plane so that crosswise frictional forces impose *normal* displacements for a hair and inplane displacements for a piece of cloth.

The problem in both cases is not isolated to straight configurations. Even as the rod or surface bends, corresponding first- and second-order force approximations continue to significantly underestimate the collision response of stretching modes (see Fig. 7.3, middle) and thus permit too much strain.

In summary, we observe that collision response methods using lower-order approximations are effectively physics-oblivious for small-bend collisions. On the other hand, as bending increases, first-order modeling gives a correspondingly better approximation (see Fig. 7.3, right).

From the above, we conclude that, while a full-blown nonlinear solution may be overkill, there are many situations where nonlinearity is absolutely essential. As a consequence, the remainder of this chapter focuses on how to establish a fully nonlinear collision response model. In the next chapter, we then focus on a simple algorithm that can flexibly adjust the amount of nonlinearity through an adaptive approach. The goal of that algorithm is to retain the cost of the standard linearly compliant methods when a first-order model is sufficient, while introducing only a minimum of additional

overhead to obtain just enough nonlinearity when higher-order modeling is required.

7.2 Stiffness and compliance

The combination of the mass matrix and the energy Hessian, which first appeared in Eq. (6.2), is used repeatedly in the following. For convenience, we refer to this as the *stiffness* matrix and define

$$\boldsymbol{K}(\mathbf{q}) \equiv \boldsymbol{M} + \tau^2 \boldsymbol{H}_{pot}(\mathbf{q}) \, .$$

It is not uncommon to refer to the energy Hessian by itself as the stiffness matrix, but, in this thesis, we use the above definition.

The notion of *compliance* is also essential for our handling of contact. As mentioned above, compliance is the inverse of stiffness, so we define the *discrete compliance matrix* as

$$\boldsymbol{D}(\mathbf{q}) \equiv (\boldsymbol{K}(\mathbf{q}))^{-1} \, .$$

Associated with the stiffness and the compliance matrices, we introduce the abbreviated notation

$$\boldsymbol{K}^{(i)} \equiv \boldsymbol{K}\left(\mathbf{q}_k + \delta^{(i)}\right)$$

and

$$\boldsymbol{D}^{(i)} \equiv \boldsymbol{D}\left(\mathbf{q}_k + \delta^{(i)}\right) .$$

For convenience, we also use

$$f^{(i)} \equiv f\left(\delta^{(i)}\right)$$

7.3 Constraint functions

In this section, we start by defining some basic notation for constraint functions.

Following Harmon et al. [2008], let $C(\mathbf{q})$ be a real-valued constraint function for a single contact, with $C(\mathbf{q}) < 0$ whenever $\mathbf{q}$ is inadmissible. This function is sometimes also referred to as the *gap function* because it typically computes the gap between two features.[2]

[2]The sign convention in the literature for the gap function is often chosen opposite the one here such that a positive gap corresponds to an interpenetration.

When m simultaneous contacts are present, we denote the set of these contacts by $\mathcal{C}$, with $m = |\mathcal{C}|$. The corresponding set of constraint functions are then C_1 through C_m. These functions are conveniently combined into a vector-valued function $\boldsymbol{C}(\mathbf{q}) = [C_1(\mathbf{q}), \ldots, C_m(\mathbf{q})]^T \in \mathbb{R}^m$. The set in configuration space given by $\boldsymbol{C}(\mathbf{q}) = 0$ is sometimes referred to as the *constraint manifold*. For inequality constraints, this set partitions the configuration space into admissible and inadmissible configuration, whereas for equality constraints, the constraint manifold *is* the set of admissible configurations.

For the special case of holonomic constraints, which can be written as $C(\mathbf{q}) = 0$, it follows by differentiation that $\nabla C\dot{\mathbf{q}} = 0$ (if this didn't hold, the system would be drifting off of the constraint manifold). This is sometimes referred to as a hidden constraint. Baraff and Witkin [1998] only consider holonomic constraints and Harmon et al. [2008] only consider the associated hidden constraint $\nabla C\dot{\mathbf{q}} = 0$. However, we are interested in both equality and inequality constraints.

Independent of whether the constraint is holonomic or not, if the constraint depends on scripted variables, which are time varying then it is referred to as a *rheonomic* constraint. Otherwise, it is *scleronomic*. In hair and cloth simulation, where the scripted objects are typically animated characters, most of the constraints with collision objects are rheonomic. When considering energy conservation, this distinction is important because rheonomic constraints effectively can introduce arbitrary amounts of energy into a system.

7.4 Constrained integration

Using the d'Alembert–Lagrange principle, we can easily introduce equality constraints into the equations of motion through a set of Lagrange multipliers :

$$\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{\mathbf{q}}_i}\right) - \frac{\partial \mathcal{L}}{\partial \mathbf{q}_i} = \sum_{j=1}^{m} \frac{\partial C_j}{\partial \mathbf{q}_i}\lambda_j, \quad \forall i$$

From this, it follows that the forces exerted by the constraints are

$$\boldsymbol{f}_{constraint} = \nabla \boldsymbol{C}^T \lambda \, .$$

Note that this force is always aligned with the gradient of the constraint function. Checking the units of this equation, we see that λ must have units of energy. Nonetheless, it is often being referred to as an impulse.

Applying the backward Euler time discretization to the constrained equation of motion leads to

$$\boldsymbol{M}\delta = \tau \boldsymbol{M}\dot{\mathbf{q}}_k - \tau^2 \nabla W_{pot}(\mathbf{q}_k + \delta) + \tau^2 \left(\nabla \boldsymbol{C}(\mathbf{q}_k + \delta)\right)^T \lambda \, . \tag{7.1}$$

Let $\boldsymbol{C}(\delta)$ be a shorthand notation for $\boldsymbol{C}(\mathbf{q}_k + \delta)$ In practice, we are then primarily concerned with the case $\boldsymbol{C}(\delta) = \boldsymbol{G}^T\delta$, where $\boldsymbol{G}$ is a constant matrix. For convenience, we also absorb τ^2 into the Lagrange multipliers to get the "discrete Lagrange multipliers" (with units of mass times displacement). This way, the last term in Eq. (7.1) simplifies to $\boldsymbol{G}\lambda$ such that the discretized constrained Euler-Lagrange equations become

$$\boldsymbol{M}\delta = \tau\boldsymbol{M}\dot{\mathbf{q}}_k - \tau^2\nabla W_{pot}(\mathbf{q}_k + \delta) + \boldsymbol{G}\lambda \tag{CDEL}$$

Just as Eq. (DEL) was a set of nonlinear equations, so is Eq. (CDEL), but now λ is an additional unknown. As before, we can use the incremental potential, but now it is subject to the specified constraints. In this case, Eq. (CDEL) is the Lagrangian (in the optimization sense of the word) of the constrained minimization problem.

Using the root finding approach, we alternatively define the function f_c for which we want to find the roots :

$$f_c(\delta, \lambda) \equiv f(\delta) - \boldsymbol{G}\lambda \,.$$

Here, $f(\delta)$ is the corresponding function for Eq. (DEL) defined earlier. We can then write one step in Newton's method for this function as

$$\nabla f_c\left(\delta^{(i)}, \lambda^{(i)}\right)\begin{bmatrix} d\delta^{(i)} \\ d\lambda^{(i)} \end{bmatrix} = -f_c\left(\delta^{(i)}, \lambda^{(i)}\right) \,.$$

Expanding this out gives

$$\begin{bmatrix} \boldsymbol{K}^{(i)} & -\boldsymbol{G} \end{bmatrix}\begin{bmatrix} d\delta^{(i)} \\ d\lambda^{(i)} \end{bmatrix} = -f^{(i)} + \boldsymbol{G}\lambda^{(i)} \,. \tag{7.2}$$

This system has more unknowns than equations, but we also need to enforce that the solution is feasible. I.e., that the constraints are satisfied. For equality constraints, we should therefore enforce $\boldsymbol{C}(\delta^{(i+1)}) = \boldsymbol{G}^T\delta^{(i+1)} = 0$. Given that $\delta^{(i+1)} = \delta^{(i)} + d\delta^{(i)}$, it follows that

$$\boldsymbol{G}^T d\delta^{(i)} = -\boldsymbol{G}^T\delta^{(i)} \,.$$

For each iteration, we therefore get the following saddle point problem :

$$\begin{bmatrix} \boldsymbol{K}^{(i)} & -\boldsymbol{G} \\ -\boldsymbol{G}^T & 0 \end{bmatrix}\begin{bmatrix} d\delta^{(i)} \\ d\lambda^{(i)} \end{bmatrix} = \begin{bmatrix} -f^{(i)} + \boldsymbol{G}\lambda^{(i)} \\ \boldsymbol{G}^T\delta^{(i)} \end{bmatrix} \,. \tag{7.3}$$

These equations are exactly the KKT conditions for the minimization of the incremental potential subject to equality constraints. In fact, Newton's

method in this case amounts to *sequential quadratic programming* (SQP) for solving a nonlinear optimization problem with equality constraints, [Nocedal and Wright, 2006, Chapter 18.1]. This method is convergent if $\boldsymbol{G}\boldsymbol{K}^{(i)}\boldsymbol{G}^T$ is positive definite for all i. In other words, $\boldsymbol{K}^{(i)}$ has to be positive definite on the constraint null space. Otherwise, the method has to be augmented by a line search or a trust region to guarantee convergence. Even if $\boldsymbol{K}^{(i)}$ is always positive definite, the reduced system may not be, so SQP methods generally require line searches or trust regions to converge just as Newton's method does. However, to simplify our exposition here, we ignore this issue.

To motivate our approach for solving more complicated systems with inequality constraints, we won't solve Eq. (7.3) directly. Instead, we first note that the unconstrained increment for the i'th iteration obtained by ignoring contact forces is given by

$$d\delta_u^{(i)} \equiv -\boldsymbol{D}^{(i)} f^{(i)} \,. \tag{7.4}$$

Thus, from Eq. (7.2), it follows that

$$d\delta^{(i)} = d\delta_u^{(i)} + \boldsymbol{D}^{(i)}\boldsymbol{G}\lambda^{(i+1)}$$

or, equivalently,

$$\delta^{(i+1)} = \delta^{(i)} + d\delta_u^{(i)} + \boldsymbol{D}^{(i)}\boldsymbol{G}\lambda^{(i+1)} \,. \tag{7.5}$$

In the remainder of our derivations, the quantity $\delta^{(i)} + d\delta_u^{(i)}$ appears often, so we define the *unconstrained update*

$$\delta_u^{(i+1)} \equiv \delta^{(i)} + d\delta_u^{(i)} \tag{7.6}$$

Using this, we can write the constraint satisfaction at the end of the time step as

$$\boldsymbol{C}(\delta^{(i+1)}) = \boldsymbol{C}(\delta_u^{(i+1)} + \boldsymbol{D}^{(i)}\boldsymbol{G}\lambda^{(i+1)}) = 0 \tag{7.7}$$

Assuming that $\delta_u^{(i+1)}$ has been computed based on Eq. (7.6) and Eq. (7.4), then the only unknown in Eq. (7.7) is $\lambda^{(i+1)}$. For the equality constraints considered so far, we have $\boldsymbol{C}(\mathbf{q}) = \boldsymbol{G}^T\mathbf{q}$, so inserting this into Eq. (7.7) shows that finding $\lambda^{(i+1)}$ amounts to solving

$$\boldsymbol{G}^T\boldsymbol{D}^{(i)}\boldsymbol{G}\lambda^{(i+1)} = -\boldsymbol{G}^T\delta_u^{(i+1)} \,. \tag{7.8}$$

The matrix $\boldsymbol{G}^T\boldsymbol{D}^{(i)}\boldsymbol{G}$ on the left is sometimes referred to as the *Delassus operator* based on work done by Etienne Delassus, [Delassus, 1917]. In the graphics world, it is also known as the constraint anticipation matrix, Baraff [1996], while in linear algebra it's the Schur complement.

7.5 Inequality constraints

Introducing inequality constraints complicates matters greatly in terms of the mathematical machinery required for a formal treatment. In particular, the equations of motion now have to be written as variational inequalities, which leads to differential inclusions instead of differential equations. The solutions to differential inclusions are generally set valued and the analysis is non-smooth. A simple example illustrating this for a single point mass is shown in [Leine and Aeberhard, 2007]. A more comprehensive overview is provided in [Moreau, 1999]. However, when all is said and done, it can still be written as the incremental potential subject to a set of constraints, and Newton's equations remain the KKT conditions for this system. In other words Eq. (7.2) plus the constraint equation(s).

To handle the set valued nature of the solutions, we define $\mathcal{R}(\delta)$ as the set of collision responses, λ, such that all constraints $\boldsymbol{C}(\mathbf{q}_k + \delta)$ are satisfied :[3]

$$\mathcal{R}(\delta) \equiv \{\lambda \in \mathbb{R}^m \,:\, \boldsymbol{C}(\mathbf{q}_k + \delta(\lambda)) \geq 0\}$$

It should be noted that this is an implicit definition because δ is a function of λ. Writing $\lambda \in \mathcal{R}(\delta)$ therefore involves some kind of a solve.

Using this notation, we can rewrite Eq. (7.7) as

$$\lambda^{(i+1)} \in \mathcal{R}(\delta_u^{(i+1)} + \boldsymbol{D}^{(i)}\boldsymbol{G}\lambda^{(i+1)}) \tag{7.9}$$

Here, $\lambda^{(i+1)}$ is the only unknown and, in the case of equality constraints, the solution to this inclusion equation is given by Eq. (7.8). With inequality constraints, the set $\mathcal{R}(\delta)$ is more complicated and it is much harder to compute a valid response. For purposes of exposition in the following, we assume that some kind of active set strategy is used for handling inequality constraints. At each step, this means that a set of constraints has been identified that should be handled as equality constraints. These can then be treated as outlined in the previous section.

Summarizing the process so far, it can be viewed as a type of predictor-corrector method similar to the method in [Kaufman et al., 2008], but extended to the nonlinear setting. For each Newton iteration, we perform the following three steps :

- *Predict :* Compute $\delta_u^{(i+1)} = \delta^{(i)} - \boldsymbol{D}^{(i)} f^{(i)}$ to obtain the predicted configuration update.

[3] In general, the full set of constraints may include both equality and inequality constraints. For convenience, we only include the inequality constraints in the definition here. The extension is obvious.

- *Correct :* Solve $\lambda^{(i+1)} \in \mathcal{R}(\delta_u^{(i+1)} + \boldsymbol{D}^{(i)}\boldsymbol{G}\lambda^{(i+1)})$ to obtain the corresponding contact forces.
- *Combine :* Compute $\delta^{(i+1)} = \delta_u^{(i+1)} + \boldsymbol{D}^{(i)}\boldsymbol{G}\lambda^{(i+1)}$ to obtain the combined update due to both dynamics and contacts.

We return to an example with inequality constraints in Section 7.8.

7.6 Nonlinear collision constraints

When detecting contact between two objects, there are generally two approaches. One looks at the configuration at the beginning of a time step and determines if the two objects are within a given *proximity* of each other. The other considers the trajectory of the two objects throughout the time step and detects if they ever intersect. This is also referred to as *continuous-time collision detection* or CCD for short.

Assuming that all of the objects are simplicial meshes (such as triangle meshes or poly-line segments), the collision detection is typically broken into collisions between the various primitives. In this context, the primitives are vertices, edges and faces, and the corresponding types of collisions are typically vertex-face (VF) and edge-edge (EE). If the primitives are assumed to have some associated thickness, then additional pairs of primitives have to be considered (e.g., vertex-vertex (VV)). There is an extensive literature about how to perform collision detection efficiently. The reader may consult e.g. Ericson [2004] or Teschner et al. [2005] for more details.

Generating constraint functions for all collisions in a system is known as *contact sampling*. To construct a single constraint function, consider two points $\boldsymbol{p}^a$ and $\boldsymbol{p}^b$ in $\mathbb{R}^3$ that are colliding, and let $\hat{\boldsymbol{n}} \in \mathbb{R}^3$ be the associated normalized contact normal (oriented from $\boldsymbol{p}^b$ to $\boldsymbol{p}^a$). Later, we also need two orthonormal vectors, $\hat{\boldsymbol{d}}_1$ and $\hat{\boldsymbol{d}}_2$, spanning the tangent plane such that $\Phi = (\hat{\boldsymbol{n}}, \hat{\boldsymbol{d}}_1, \hat{\boldsymbol{d}}_2)$ forms a complete basis for $\mathbb{R}^3$ around the point of contact.

If we prescribe that the two points must remain at least some distance d (which may be zero) apart, then we get

$$C = \hat{\boldsymbol{n}} \cdot (\boldsymbol{p}^a - \boldsymbol{p}^b) - d$$

This function effectively computes the *signed* distance between $\boldsymbol{p}^a$ and $\boldsymbol{p}^b$ and requires that it be at least d. In general, $\boldsymbol{p}^a$, $\boldsymbol{p}^b$, and $\hat{\boldsymbol{n}}$ are all functions of the current configuration.

Each colliding point will typically either be at a vertex, on an edge, or on a face. As such, we can write both of them as a linear combination of neighboring vertices using barycentric coordinates. This generalizes if the underlying surface is a subdivision surface rather than a triangle mesh, since the colliding point can still be written as a linear combination of a larger set of points. In both cases, we have

$$\boldsymbol{p}^a = \sum_{j \in \mathcal{S}_{p_a}} \gamma_j^a \boldsymbol{x}_j, \quad \boldsymbol{p}^b = \sum_{j \in \mathcal{S}_{p_b}} \gamma_j^b \boldsymbol{x}_j \,,$$

where $\mathcal{S}_{p_a}$ and $\mathcal{S}_{p_b}$ are the sets of vertices defining the feature (vertex, edge or face) on which $\boldsymbol{p}^a$ and $\boldsymbol{p}^b$ respectively reside. The set $\mathcal{S} = \mathcal{S}_{p_a} \cup \mathcal{S}_{p_b}$ is called the *stencil* of the contact.

Let $\gamma_j = \gamma_j^a$ for $j \in \mathcal{S}_{p_a}$ and $\gamma_j = -\gamma_j^b$ for $j \in \mathcal{S}_{p_b}$. If $\mathcal{S}_{p_a}$ and $\mathcal{S}_{p_b}$ are not disjoint, then $\gamma_j = \gamma_j^a - \gamma_j^b$ for $j \in \mathcal{S}_{p_a} \cap \mathcal{S}_{p_b}$. The *gap vector*, $\boldsymbol{g}$, is then given by

$$\boldsymbol{g} \equiv \boldsymbol{p}^a - \boldsymbol{p}^b = \sum_{j \in \mathcal{S}} \gamma_j \boldsymbol{x}_j \,.$$

Note that barycentric coordinates (and subdivision weights) form a partition of unity, so we have

$$\sum_{j \in \mathcal{S}_{p_a}} \gamma_j^a = \sum_{j \in \mathcal{S}_{p_b}} \gamma_j^b = 1$$

and, therefore,

$$\sum_{j \in \mathcal{S}} \gamma_j = 0$$

The time at which the barycentric coordinates are evaluated differs with the type of constraint. Baraff and Witkin [1998] evaluate these coordinates at the *beginning* of the time step for proximity constraints. However, following Harmon et al. [2008] and Otaduy et al. [2009], they are typically evaluated at the time of collision for collision constraints. In the latter case, it should be noted that the coordinates are functions of the (unknown) end positions. This makes it complicated to compute the constraint gradient since a change in the end positions may cause the collision to cease to exist. Computing the gradient in this case is not a well-defined operation. In practice, the time of collision is therefore often assumed to be fixed and typically evaluated based on an initial *unconstrained* solution for the dynamics. The time of collision in this context is the time at which $C(\mathbf{q}(t_c)) = 0$. If $d = 0$, then this typically involves solving a cubic polynomial, [Provot, 1997], while it becomes a sixth

order polynomial if $d > 0$. If there are multiple roots within a time step, then the earliest is typically chosen and the others are ignored.

The time at which the contact normal is evaluated also differs. For cloth-object collisions, Baraff and Witkin [1998] evaluate it at the *end* of the time step to ensure that the cloth feature ends up outside the collision object. However, for collisions between two deformable features, it is typically evaluated in the limit as $t \to t_c^-$, where t_c^- indicates that the limit is taken from the left (i.e., going *into* the collision). The limit consideration is needed because the objects may be infinitely thin, so as they go through each other the direction from one to the other will flip. It should be noted though that this problem goes away if we consider each object to have some thickness (i.e., if $d > 0$ in our constraint formulation).

7.7 Linearized collision constraints

The constraint function introduced in the previous section is highly non-linear. In practice, we therefore consider the linearization of the constraint function with respect to time. To this end, it is convenient to introduce the *relative deformation gradient*, which is the gradient of the gap vector with respect to the configuration variables :

$$\Gamma(\mathbf{q}) \equiv \begin{bmatrix} \Gamma^f & \Gamma^s \end{bmatrix} \equiv \nabla \boldsymbol{g} \,.$$

We note that, for deformable objects like cloth and hair, if $\boldsymbol{x}_j$ is the j'th vertex in the configuration vector $\mathbf{q}$, then we have

$$\nabla \boldsymbol{x}_j = \boldsymbol{e}_j^T \otimes \boldsymbol{I}_3 \,,$$

where $\boldsymbol{e}_j$ is the j'th canonical basis vector in $\mathbb{R}^n$ and $\boldsymbol{I}_3$ is a 3×3 identity matrix. I.e., Γ has a very simple structure with as many non-zero blocks as there are vertices in the collision stencil, and each one of them is simply a 3×3 identity matrix scaled by the corresponding weight for the contact point. As noted above, $\sum_j \gamma_j = 0$ when summing over the full stencil, which leads to a zero row-sum in Γ. However, if some of the vertices are scripted, then the row-sum in Γ^f by itself is not zero.

We also note that the relative velocity of the two points in contact is given by $\boldsymbol{u} = \Gamma \dot{\mathbf{q}}$, from which it follows that the *normal velocity* associated with the contact is $\boldsymbol{u}_n = \hat{\boldsymbol{n}}^T \Gamma \dot{\mathbf{q}}$. The tangential sliding velocity, $\boldsymbol{u}_t \in \mathbb{R}^2$, can be found in a similar way.

Γ^T can be thought of as a map from the local cordinates of a single contact to generalized coordinates. We use this to define the *generalized contact normal*

as

$$\mathbf{n} \equiv \begin{bmatrix} \mathbf{n}^f \\ \mathbf{n}^s \end{bmatrix} \equiv \Gamma^T \hat{\boldsymbol{n}} \,.$$

When a constraint is active, then the part of the generalized contact normal corresponding to deformable vertices, $\mathbf{n}^f$, represents a direction in configuration space along which motion is disallowed for equality constraints and only allowed in one direction for inequality constraints. Furthermore, any valid constraint force or impulse must act along $\mathbf{n}^f$.

Generalized tangent vectors for a point of contact can be constructed in a similar way, and we note that if $\boldsymbol{y}$ is an impulse applied to point $\boldsymbol{p}^a$ and an equal but opposite impulse is applied to point $\boldsymbol{p}^b$, then $\Gamma^T \boldsymbol{y}$ is the resulting generalized impulse.

Using the above definitions, we have

$$\begin{aligned} C(\mathbf{q}) &= \hat{\boldsymbol{n}}^T \boldsymbol{g} - d \\ \nabla C &= (\nabla \hat{\boldsymbol{n}})^T \boldsymbol{g} + \hat{\boldsymbol{n}}^T \nabla \boldsymbol{g} \\ &= (\nabla \hat{\boldsymbol{n}})^T \boldsymbol{g} + \hat{\boldsymbol{n}}^T \Gamma \\ &= (\nabla \hat{\boldsymbol{n}})^T \boldsymbol{g} + \mathbf{n}^T \\ \dot{C} &= \nabla C \dot{\mathbf{q}} \end{aligned}$$

Assuming that the time of collision, t_c, is given, the Taylor expansion of $C(t) = C(\mathbf{q}(t))$ around t_c is therefore

$$\begin{aligned} C(t) &= C(t_c) + \dot{C}(t_c)(t - t_c) + \mathcal{O}((t - t_c)^2) \\ &= \left(\left((\nabla \hat{\boldsymbol{n}})^T \boldsymbol{g} + \mathbf{n}^T \right) \dot{\mathbf{q}} \right) \Big|_{t=t_c} (t - t_c) + \mathcal{O}((t - t_c)^2) \end{aligned}$$

In general, the contact normal can be written as $\boldsymbol{n} = \boldsymbol{u} \times \boldsymbol{v}$, where $\boldsymbol{u}$ and $\boldsymbol{v}$ are either two edges of a triangle face coming into contact or they are the two unconnected edges coming into contact. The gradient of this normalized cross product can be written as

$$\nabla \hat{\boldsymbol{n}} = \frac{(\boldsymbol{u} \times \hat{\boldsymbol{n}}) \hat{\boldsymbol{n}}^T \nabla \boldsymbol{v} - (\boldsymbol{v} \times \hat{\boldsymbol{n}}) \hat{\boldsymbol{n}}^T \nabla \boldsymbol{u}}{\|\boldsymbol{n}\|}$$

At the time of contact, the gap vector, $\boldsymbol{g}$, is either zero or parallel to $\boldsymbol{n}$, so since $(\boldsymbol{u} \times \boldsymbol{n}) \cdot \boldsymbol{n} = (\boldsymbol{v} \times \boldsymbol{n}) \cdot \boldsymbol{n} = 0$ for any vectors $\boldsymbol{u}$ and $\boldsymbol{v}$, it follows that

$$\left((\nabla \hat{\boldsymbol{n}})^T \boldsymbol{g} \right) \Big|_{t=t_c} = 0 \,.$$

Consequently, the Taylor expansion of $C(t)$ simplifies to

$$C(t) = \left(\mathbf{n}^T \dot{\mathbf{q}}\right)\Big|_{t=t_c^-} (t - t_c) + \mathcal{O}((t - t_c)^2) \,,$$

which leads to the *linearized constraint function* for the position constraint :

$$C^L(t) \equiv \left(\mathbf{n}^T \dot{\mathbf{q}}\right)\Big|_{t=t_c^-} (t - t_c) \geq 0 \tag{7.10}$$

Typically, we want to enforce the constraint at the end of a timestep to ensure that the new configuration satisfies the constraint, [Goldenthal et al., 2007]. Let $\delta = (\delta^f, \delta^s)$ represent the change in $\mathbf{q}$ within one time step. In practice, we *assume* piecewise linear trajectories, so it follows that $\dot{\mathbf{q}}$ is constant within the time step such that $\dot{\mathbf{q}}(t_c^-) = \delta/\tau$, where τ is the length of the time step.

Although equations in the following are stated in terms of δ, it should be noted that the combination of the linearization of the constraint function and the fact that we always evaluate it at the end of a time step effectively turns it into a velocity level constraint. Eq. (7.10) simply states that the post-impact relative velocity of the two contacting points should be non-negative.

The constraint in Eq. (7.10) can be scaled arbitrarily, so, for convenience, we scale it by $\tau/(t_{end} - t_c)$ such that the linearized constraint evaluated at the end of the time step can be written as

$$\mathbf{n}^T \delta \geq 0 \,. \tag{7.11}$$

In practice, this constraint is further scaled to ensure that $\mathbf{n}$ has unit length as this improves the conditioning of subsequent linear systems. If there are scripted vertices involved in the constraint, then we can use the partitioning of the variables to re-write the above inequality as

$$(\mathbf{n}^f)^T \delta^f \geq -(\mathbf{n}^s)^T \delta^s \,.$$

It should be noted that the quantities on the right-hand side here are known and can be represented by a single constant $s \equiv -(\mathbf{n}^s)^T \delta^s$. This leads to the constraint that is used by Kaufman et al. [2014] for a single contact point :

$$(\mathbf{n}^f)^T \delta^f - s \geq 0 \tag{7.12}$$

However, since this is equivalent to Eq. (7.11), we use the former.

Technically, the above derivation is only valid when the contact normal is evaluated at the time of collision, which means that it does not apply to the scheme used by Baraff and Witkin [1998] where the contact normal is evaluated at the end of the time step. However, in practice, they only consider

cloth-vertex against object-face collisions, so the constraint function in this case is linear in $\mathbf{q}^f$ and we end up with the same expression as in Eq. (7.12), albeit with the weights evaluated at the beginning of the time step and the contact normal at the end of the time step.

Multiple constraints can easily be represented by letting

$$\boldsymbol{N} = \begin{bmatrix} \mathbf{n}_1 & \dots & \mathbf{n}_m \end{bmatrix}.$$

At this point, the combined set of constraints can be written as

$$\boldsymbol{N}^T \delta \geq 0.$$

While this constraint is simple, it's important to realize that it is based on a number of approximations compared to the desired constraint. In particular, we have assumed

- Fixed time of collision (and only one such time).
- Linear trajectories of all vertices.
- Linearized constraint function.
- Constraint only enforced at end of time step.

It should also be noted that the combination of the linearization of the constraint function and the assumption about linear trajectories effectively turns the original position level constraint into a velocity level constraint.

If the constraint is not linearized (but the time of collision is still assumed to be given), then Harmon [2010] showed that the constraint function is in general not only nonlinear but also non-convex, which makes the subsequent optimization problems substantially more difficult.

7.8 Non-penetration model

Associated with each collision constraint is a collision response force denoted by $\boldsymbol{r} \in \mathbb{R}^3$. This force is usually broken into two components : A contact force, $\boldsymbol{r}_n$, to ensure non-penetration, and a friction force, $\boldsymbol{r}_t$, such that $\boldsymbol{r} = \boldsymbol{r}_n + \boldsymbol{r}_t$. The corresponding impulses for contact k is denoted by $\alpha_k \in \mathbb{R}$ and $\beta_k \in \mathbb{R}^2$, and the generalized forces is denoted by $\mathbf{r}$, $\mathbf{r}_n$, and $\mathbf{r}_t$, respectively.

Collectively, the contact impulses are denoted by $\alpha = (\alpha_1, \dots, \alpha_m)$. Based on the Signorini-Fischera conditions, [Kikuchi and Oden, 1988], the contact

force is generally determined through a nonlinear complementarity problem :

$$0 \leq \alpha \perp \boldsymbol{C}(\mathbf{q}(t, \alpha)) \geq 0. \tag{SFC}$$

Here, $\boldsymbol{x} \perp \boldsymbol{y}$ denotes the complementarity condition $x_i y_i = 0$ for all $i \in \{1, \ldots, m\}$. Since both vectors in the above case only have non-negative entries this is equivalent to saying that $\boldsymbol{x}^T \boldsymbol{y} = 0$ or simply that the vectors are orthogonal.

What the equation states is basically that :

- the contact forces must be non-negative (no adhesion)
- the configuration must be admissible
- there cannot be a gap and a contact force at the same time.

We explicitly write $\boldsymbol{C}(\mathbf{q}(t, \alpha))$ to emphasize the fact that the configuration at a given point in time depends on the contact forces in the system.

In the following, we simplify the problem by using the contact constraint derived in the previous section, which means that $\boldsymbol{C}(\delta) = \boldsymbol{N}^T \delta$. In other words, the contact constraints have been linearized, but it is in general a different linearization than the one we perform for the dynamics, and it is assumed to be fixed.

With this simplification, the relation between the contact impulses and the global contact forces is (c.f. Section 7.4)

$$\mathbf{r}_n = \boldsymbol{N}\alpha \, .$$

The complementarity condition on its own is not enough to determine the contact impulses since it does not constrain the magnitude of the contact impulses. To do so, we must include knowledge about the dynamics as in Section 7.4. Since the dynamics itself is nonlinear, the resulting system remains nonlinear, even though we linearized the constraints. Hence, we must solve the system as in Section 7.4 with Newton's method. However, given that we now have inequality constraints, each step in Newton's method corresponds to solving a set of KKT equations.

In particular, the dynamics for a single iteration is given by Eq. (7.2) and, to satisfy the constraints, we must have $\alpha^{(i+1)} \in \mathcal{R}(\delta^{(i+1)})$. Using Eq. (SFC) to define the feasible set, we end up with a *mixed linear complementarity problem* (MLCP) :

$$\begin{aligned} \boldsymbol{K}^{(i)} d\delta^{(i)} &= -f^{(i)} + \boldsymbol{N}\alpha^{(i+1)} \, , \\ 0 \leq \alpha^{(i+1)} &\perp \boldsymbol{N}^T (\delta^{(i)} + d\delta^{(i)}) \geq 0. \end{aligned} \tag{7.13}$$

Recalling from Eq. (7.4) that the unconstrained increment is given by

$$d\delta_u^{(i)} = -\boldsymbol{D}^{(i)} f^{(i)}$$

and that $\boldsymbol{D} = \boldsymbol{K}^{-1}$, we can rewrite the problem as

$$\begin{aligned} \boldsymbol{K}^{(i)} \left(d\delta^{(i)} - d\delta_u^{(i)} \right) &= \boldsymbol{N}\alpha^{(i+1)} \,, \\ 0 \leq \alpha^{(i+1)} \perp \boldsymbol{N}^T (\delta^{(i)} + d\delta^{(i)}) &\geq 0. \end{aligned}$$

If we furthermore define

$$\Delta\delta^{(i)} \equiv d\delta^{(i)} - d\delta_u^{(i)} \,,$$

then

$$\delta^{(i)} + d\delta^{(i)} = \delta^{(i)} + \Delta\delta^{(i)} + d\delta_u^{(i)} = \delta_u^{(i+1)} + \Delta\delta^{(i)}$$

and our problem above becomes

$$\begin{aligned} \boldsymbol{K}^{(i)} \Delta\delta^{(i)} &= \boldsymbol{N}\alpha^{(i+1)} \,, \\ 0 \leq \alpha^{(i+1)} \perp \boldsymbol{N}^T \Delta\delta^{(i)} &\geq -\boldsymbol{N}^T \delta_u^{(i+1)} \,. \end{aligned}$$

Starting with an initial guess of $\delta^{(0)} = 0$ and considering only the first Newton iteration, this effectively recovers the contact formulation in Otaduy et al. [2009]. The only difference is due to the fact that the contact constraints in that paper are linearized slightly differently. However, for what follows, the important point is that the framework considered here is a natural generalization of the formulation used by Otaduy et al. [2009].

7.9 Restitution

A subtle "feature" of the LCP formulation for contact is that, for any active constraint, the solution is such that the corresponding inequality is actually satisfied as an equality. In practice, this means that the relative velocity for all active constraints at the end of the time step is zero. I.e. two points of contact at the end of a time step do not break apart. A different way of stating this is to say that all collisions are treated inelastically. See Smith et al. [2012] for additional discussion. For cloth and hair, this is a reasonable approximation of reality, but for rigid body collisions it is often nice to be able to include a coefficient of restitution. This can be added by introducing the generalized reflection operator from Smith et al. [2012] at the cost of having to solve a sequence of LCPs, but we do not discuss this any further here.

The fact that all collisions end up being treated inelastically is a consequence of our choice of integration scheme. In fact, we have the following lemma :

Lemma 4. A fully *inelastic impact* will only be obtained by one-step numerical integrators that satisfy an implicit-Euler–type velocity update of the form $\dot{\mathbf{q}}^{t+1} = a\delta$, $a \in \mathbb{R}$.

Proof. Recall that we have $\mathbf{q}^{t+1} = \mathbf{q}^t + \delta$ and that imposing a linearized non-negative displacement constraint requires $0 \leq \alpha_k \perp \mathbf{n}_k^T \delta \geq 0$. Whenever a contact force is applied, we have $\alpha_k > 0$ and thus $\mathbf{n}_k^T \delta = 0$. A fully inelastic impact is given by a response satisfying the velocity-level condition $\mathbf{n}_k^T \dot{\mathbf{q}}^{t+1} = 0$. By substitution this can only be satisfied by an implicit-Euler–type update, where $\dot{\mathbf{q}}^{t+1}$ is given by a scaling of δ. □

As concrete examples, consider that the implicit Euler and implicit midpoint velocity updates are, respectively, $\dot{\mathbf{q}}_E^{t+1} = \frac{1}{\tau}\delta$ and $\dot{\mathbf{q}}_M^{t+1} = \frac{2}{\tau}\delta - \dot{\mathbf{q}}^t$. Then, by substitution, we have a *fully inelastic impact* for implicit Euler since $\mathbf{n}_k^T \dot{\mathbf{q}}_E^{t+1} = 0$ and, on the other hand, a *fully elastic* impact for implicit midpoint, corresponding to a coefficient of restitution equal to one, since $\mathbf{n}_k^T \dot{\mathbf{q}}_M^{t+1} = -\mathbf{n}_k^T \dot{\mathbf{q}}^t$.

7.10 Friction model

In addition to the forces preventing interpenetration, an important aspect of contact handling is friction. Capturing both stick (static friction) and slip (dynamic friction) is essential for reproducing many of the phenomena that we see on a daily basis (e.g., clumping of hair). In the following, we focus on the full Coulomb friction model for this purpose.

For a given contact point, k, in addition to the contact normal, $\hat{\boldsymbol{n}}$, let $\hat{\boldsymbol{d}}_1$ and $\hat{\boldsymbol{d}}_2$ be two orthogonal vectors in the tangent plane of the contact. The generalized tangent basis is then

$$\boldsymbol{T}_k \equiv \begin{bmatrix} \boldsymbol{T}^f \\ \boldsymbol{T}^s \end{bmatrix} \equiv \begin{bmatrix} \Gamma^T \hat{\boldsymbol{d}}_1 & \Gamma^T \hat{\boldsymbol{d}}_2 \end{bmatrix} \in \mathbb{R}^{n \times 2} .$$

The sliding velocity at the point of contact follows as $\boldsymbol{u}_t = \boldsymbol{T}_k^T \dot{\mathbf{q}} \in \mathbb{R}^2$.

The maximal dissipation principle introduced by Moreau [1970, 2011], and later extended by Goyal et al. [1991], requires that the friction force satisfies the following maximization :

$$\max_{\boldsymbol{r}_t} \left\{ -\boldsymbol{r}_t^T \boldsymbol{u}_t \; : \; \|\boldsymbol{r}_t\| \leq \mu \|\boldsymbol{r}_n\| \right\} \tag{7.14}$$

where $\boldsymbol{r}_n \in \mathbb{R}$ is the normal component of the collision response (corresponding to α_k in the previous section) and $\boldsymbol{r}_t \in \mathbb{R}^2$ is the tangential (frictional) part of the collision response. Since the product of force and velocity is power, the expression chooses the direction of the friction force that

maximizes the negative power applied by the force to the system (i.e., the dissipation). For isotropic friction, this means that the friction force will be oriented opposite the sliding velocity vector, but for anisotropic friction, the friction force and the sliding velocity may not be co-linear.

This formulation is equivalent to other formulations of Coulomb's law (see e.g. Bertails-Descoubes et al. [2011] or Stewart [2011]). Effectively, we note that the constraint in Eq. (7.14) amounts to a cone constraint for the combined collision response.

We should note that the above expression does not distinguish between static and dynamic friction in the sense that μ is assumed to be known apriori and constant. In a more sophisticated model, one would have $\mu = \mu(\|\boldsymbol{u}_t\|)$.

To discretize Eq. (7.14), we introduce the discrete friction force $\beta_k = (\beta_1, \beta_2)_k^T \in \mathbb{R}^2$ and replace $\dot{\mathbf{q}}$ by the approximation given by our implicit time discretization. I.e. $\dot{\mathbf{q}} \approx \delta/\tau$. Combined, β_k is therefore given by

$$\beta_k = \underset{\beta_k}{\operatorname{argmin}} \left\{ \tfrac{1}{\tau} \beta_k^T \boldsymbol{T}_k^T \delta \;:\; \|\beta_k\| \leq \mu \alpha_k \right\}. \tag{DMD}$$

Clearly, the factor $1/\tau$ does not affect the minimization and can therefore be left out. However, it should be noted that the solution does depend on the full configuration, including any scripted vertices in the collision stencil. We refer to the above equation as the *discrete maximal dissipation* principle (DMD for short).

The friction cone in Eq. (DMD) is sometimes approximated by a friction pyramid, [Trinkle et al., 1997; Stewart, 2000; Otaduy et al., 2009] :

$$\beta_k = \underset{\beta_k}{\operatorname{argmin}} \left\{ \tfrac{1}{\tau} \beta_k^T \boldsymbol{T}_k^T \delta \;:\; \|\beta_k\|_\infty \leq \mu \alpha_k \right\}.$$

This effectively linearizes the constraint. When the dynamics is also linearized, this means that the friction problem can be written as a linear complementarity problem. The pyramid in Otaduy et al. [2009] is oriented such that one of the sides is aligned with the best guess for the sliding direction. In practice, this means that $\hat{\boldsymbol{d}}_1$ should be aligned with this direction, while $\hat{\boldsymbol{d}}_2 = \hat{\boldsymbol{n}} \times \hat{\boldsymbol{d}}_1$. The above formulation is for a four-sided pyramid, but the approximation can obviously be improved by adding more facets, although this comes at the cost of additional constraints, [Stewart, 2000; Kaufman et al., 2008].

When using $\|\beta_k\|_\infty \leq \mu \alpha_k$ as the constraint, the friction cone is contained within the pyramid, while the polygon suggested by Stewart [2000] is inscribed within the friction cone. In case of a four-sided pyramid, the latter is

equivalent to $\|\beta_k\|_1 \leq \mu\alpha_k$. In either case, the constraint can be written as a set of linear constraints.

Specifically, for $\|\beta_k\|_\infty \leq \mu\alpha_k$, we can write out all the constraints as

$$\begin{aligned} |\beta_{1,k}| &\leq \mu\alpha_k \,, \\ |\beta_{2,k}| &\leq \mu\alpha_k \end{aligned}$$

This is equivalent to

$$\begin{aligned} \beta_{1,k} &\leq \mu\alpha_k \,, \quad & \beta_{1,k} &\geq -\mu\alpha_k \,, \\ \beta_{2,k} &\leq \mu\alpha_k \,, \quad & \beta_{2,k} &\geq -\mu\alpha_k \,. \end{aligned}$$

or

$$\begin{aligned} \beta_{1,k} &\leq \mu\alpha_k \,, \quad & -\beta_{1,k} &\leq \mu\alpha_k \,, \\ \beta_{2,k} &\leq \mu\alpha_k \,, \quad & -\beta_{2,k} &\leq \mu\alpha_k \,. \end{aligned}$$

In matrix form, we can write it as

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ -1 & 0 \\ 0 & -1 \end{bmatrix} \beta_k \leq \mu\alpha_k \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \,.$$

Thus, we have the same number of friction force components (two), but we have four constraints instead of one.

If instead we follow Stewart [2000] and write the linearized constraint for a four-sided inscribed pyramid, then we first need to redefine

$$\boldsymbol{T}_k \equiv \begin{bmatrix} \Gamma^T \hat{\boldsymbol{d}}_1 & \Gamma^T \hat{\boldsymbol{d}}_2 & -\Gamma^T \hat{\boldsymbol{d}}_1 & -\Gamma^T \hat{\boldsymbol{d}}_2 \end{bmatrix} \in \mathbb{R}^{n\times 4} \,.$$

At this point, the constraint then becomes

$$\begin{aligned} \boldsymbol{e}^T \beta_k &\leq \mu\alpha_k \,, \\ \beta_k &\geq 0 \,, \end{aligned}$$

where β_k now has four components and $\boldsymbol{e} = (1,1,1,1)^T$. The advantage of this formulation is that additional facets can be added to the pyramid by adding more directions to the definition of $\boldsymbol{T}_k$ and corresponding components to β_k. With this formulation, we can write all the friction constraints together as

$$\begin{aligned} \boldsymbol{E}^T \beta &\leq \mathrm{diag}(\mu)\alpha \,, \\ \beta &\geq 0 \,, \end{aligned}$$

where $\beta = \begin{bmatrix}\beta_1^T & \dots & \beta_m^T\end{bmatrix}^T$ and $\boldsymbol{E}$ is a matrix where column k has ones in the rows corresponding to the entries in β for contact k and zeros in all other rows. This is the formulation used by Kaufman [2009].

We note that this formulation requires twice as many components in β and five times as many constraints as the original quadratic formulation.

7.11 Contact force coupling

Thus far, we have tacitly ignored the fact that the solutions for α and β are coupled in an intricate way because the friction depends on the contact force and vice versa. While the two problems *can* be combined into a single minimization problem, [Kaufman, 2009, Section 5.5.3], this problem corresponds to a non-convex QP which in general is NP-hard to solve. Yet, as noted by Kaufman et al. [2008], accurate coupling between friction impulses, contact impulses, and deformation is extremely important when taking large time steps since inaccurate handling of this can cause system instabilities. Developing a practical solution for this problem is the focus of the next chapter.

CHAPTER 8

Contact solver

In this chapter, we investigate how to write an efficient solver for constrained dynamics with both contact constraints and friction. We start by considering an extension of the *staggered projections* approach originally introduced by Kaufman et al. [2008] and show what challenges that leads to for large-scale nonlinear elasticity. Next, we introduce a different approach by handling all the contacts in a Gauss-Seidel fashion while solving each individual contact problem accurately with respect to the smooth friction cone. We build that into a solver that **A**dapts the **D**egree **O**f **N**onlinearity in the **I**mpact **S**olve (ADONIS) to arrive at an effective contact solver for thin rods and hair.

8.1 Nonlinear staggered projections

Kaufman et al. [2008] proposed to find contact and frictions forces using a sequence of *staggered projections*. The essence of this approach is to solve alternatingly for contact forces and friction forces, where each solve amounts to a projection onto a convex set. The combination of the projections forms a contractive mapping, which is therefore guaranteed to converge[1].

The original algorithm presented in [Kaufman et al., 2008] does not deal with nonlinear elasticity, and therefore does not include any Newton iterations. However, the principal challenge for our problems lies in solving the large-scale QPs associated with the projection operations. Also, the friction

[1] The convergence rate, however, depends on the specifics of the problem.

model considered by Kaufman et al. [2008] is based on a linearized friction cone. Working with the smooth friction cone introduces another level of complexity.

In the following, we start by showing how to write the contact forces and friction forces within each Newton iteration as the solution of two coupled QP problems. This is essentially the staggered projection formulation, but with full compliance.

8.2 QP formulation of non-penetration

The mixed linear complementarity problem in Eq. (7.13) corresponds exactly to the KKT conditions for a quadratic program (QP). As such, the solution to Eq. (7.13) is equivalent to the solution of a QP. To see this, consider the following generic QP :

$$\begin{aligned} \mathbf{x}^\star = \underset{\mathbf{x}}{\text{argmin}} \quad & \tfrac{1}{2}\mathbf{x}^T \boldsymbol{A}\mathbf{x} + \mathbf{b}^T\mathbf{x} \,, \\ \text{s.t.} \quad & \boldsymbol{B}(\mathbf{x}+\mathbf{d}) \leq 0 \end{aligned} \tag{8.1}$$

where $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ and $\boldsymbol{B} \in \mathbb{R}^{m\times n}$ are matrices and $\mathbf{x}$, $\mathbf{b}$, and $\mathbf{d}$ are vectors in $\mathbb{R}^n$. The star superscript is used to denote the optimal value for an optimization problem. Following [Boyd and Vandenberghe, 2004, Section 5.5.3], the KKT conditions for the problem are :

$$\begin{aligned} \boldsymbol{A}\mathbf{x} + \mathbf{b} + \boldsymbol{B}^T\lambda &= 0 \\ \lambda &\geq 0 \\ \boldsymbol{B}(\mathbf{x}+\mathbf{d}) &\leq 0 \\ \lambda^T\boldsymbol{B}(\mathbf{x}+\mathbf{d}) &= 0 \end{aligned}$$

Letting

$$\begin{aligned} \boldsymbol{A} &= \boldsymbol{K}^{(i)} \,, \quad \mathbf{x} = d\delta^{(i)} \,, \quad \mathbf{b} = f^{(i)} \,, \\ \boldsymbol{B} &= -\boldsymbol{N}^T \,, \quad \mathbf{d} = \delta^{(i)} \,, \quad \lambda = \alpha^{(i+1)} \,, \end{aligned} \tag{8.2}$$

we then see that Eq. (7.13) is equivalent to the KKT conditions for the following QP :

$$\begin{aligned} d\delta^{(i)} = \underset{\mathbf{x}}{\text{argmin}} \quad & \tfrac{1}{2}\mathbf{x}^T\boldsymbol{K}^{(i)}\mathbf{x} + \mathbf{x}^T f^{(i)} \\ \text{s.t.} \quad & \boldsymbol{N}^T(\mathbf{x}+\delta^{(i)}) \geq 0 \end{aligned} \tag{8.3}$$

We note that this QP is not simply the incremental potential subject to the desired constraints. Rather, it is the QP equivalent to *one step* of Newton's method for minimizing the incremental potential subject to the constraints.

For this reason, it is sometimes also known as the *local QP*. Within this problem, the quadratic matrix, $\boldsymbol{K}^{(i)}$, is constant and can be readily computed, which makes it feasible to solve this QP, although a large-scale QP with affine constraints remains computationally expensive.

Once a solution to Eq. (8.3) has been found, it is straight forward to use Eq. (7.13) to find the contact forces :

$$\mathbf{r}_n^{(i+1)} = \boldsymbol{N}\alpha^{(i+1)} = \boldsymbol{K}^{(i)} d\delta^{(i)} + f^{(i)} .$$

This does not immediately provide $\alpha^{(i+1)}$, but if $\boldsymbol{N}$ has full rank, then $\alpha^{(i+1)}$ can be recovered using a least squares approach by premultiplying by $\boldsymbol{N}^T$ and solving for $\alpha^{(i+1)}$.

8.3 QP formulation of friction forces

In Section 7.10, the friction model for a single point of friction was established. To handle multiple points of contact, we maximize the total dissipation, which is simply the sum of the dissipation at all of the contact points. Let

$$\boldsymbol{T} = \begin{bmatrix} \boldsymbol{T}_1 & \dots & \boldsymbol{T}_m \end{bmatrix}, \quad \beta = \begin{bmatrix} \beta_1^T & \dots & \beta_m^T \end{bmatrix}^T .$$

The combined set of friction forces is then given by

$$\beta = \underset{\beta}{\operatorname{argmin}} \left\{ \beta^T \boldsymbol{T}^T \delta \; : \; \|\beta_k\| \leq \mu\alpha_k, \; 1 \leq k \leq m \right\}. \tag{8.4}$$

where we have omitted the immaterial constant $1/\tau$ from the objective function. At this point, we can write the global friction force as

$$\mathbf{r}_t = \boldsymbol{T}\beta$$

In general we therefore have that the combined collision response force is given by :

$$\mathbf{r} = \mathbf{r}_n + \mathbf{r}_t = \boldsymbol{N}\alpha + \boldsymbol{T}\beta$$

From Section 7.4, we have :

$$\delta^{(i+1)} = \delta_u^{(i+1)} + \boldsymbol{D}^{(i)} \boldsymbol{G} \lambda^{(i+1)} ,$$

where $\boldsymbol{G}\lambda^{(i+1)}$ effectively is the collision response. Substituting the above expression this leads to

$$\delta^{(i+1)} = \delta_u^{(i+1)} + \boldsymbol{D}^{(i)} (\boldsymbol{N}\alpha^{(i+1)} + \boldsymbol{T}\beta^{(i+1)}) , \tag{8.5}$$

Assuming that the contact forces have already been computed as outlined in the previous section, this expression can be inserted into Eq. (8.4) to get the following optimization problem for the friction forces :

$$\begin{aligned} \beta^{(i+1)} = \underset{\beta}{\operatorname{argmin}} \quad & \beta^T \boldsymbol{T}^T \boldsymbol{D}^{(i)} \boldsymbol{T} \beta + \beta^T \boldsymbol{T}^T \left(\delta_u^{(i+1)} + \boldsymbol{D}^{(i)} \boldsymbol{N} \alpha^{(i+1)} \right) . \\ \text{s.t.} \quad & \|\beta_k\| \leq \mu \alpha_k, \quad 1 \leq k \leq m \end{aligned} \tag{8.6}$$

This is the QP for friction forces. However, we note that, for the smooth friction cone, the constraints are quadratic. In other words, it is a quadratically constrained QP (QCQP). By approximating the smooth friction cone by a linearized friction pyramid as outlined in Section 7.10, the quadratic constraints can be linearized, leading to a simpler QP.

Previously, when we derived the MLCP for contact in Eq. (7.13), we only included the contact forces in the expression for the constraint forces. At this point, we can use Eq. (8.5) to also include the friction forces. Effectively, this means that the substitution in Eq. (8.2) for $\mathbf{b}$ has to be updated to $\mathbf{b} = f^{(i)} - \boldsymbol{T}\beta^{(i+1)}$. As a result, we get the following QP for the displacement :

$$\begin{aligned} d\delta^{(i)} = \underset{\mathbf{x}}{\operatorname{argmin}} \quad & \tfrac{1}{2}\mathbf{x}^T \boldsymbol{K}^{(i)} \mathbf{x} + \mathbf{x}^T (f^{(i)} - \boldsymbol{T}\beta^{(i+1)}) . \\ \text{s.t.} \quad & \boldsymbol{N}^T (\mathbf{x} + \delta^{(i)}) \geq 0 \end{aligned} \tag{8.7}$$

8.4 Dual formulation of contact forces

The problem solved in Eq. (8.7) provides the change in configuration for a time step, which we refer to as the *primal problem*. However, as part of the constraints for the friction forces in Eq. (8.6), we need to know what the (discrete) contact forces are. Since these are the Lagrange multipliers in the primal problem, and the primal problem is convex (under the assumption that $\boldsymbol{K}^{(i)}$ is positive definite), they can be obtained by solving the corresponding *dual problem*.

To derive the dual problem, we start with the generic QP problem in Eq. (8.1). The procedure we follow is described in [Boyd and Vandenberghe, 2004, Section 5.1.6]. First, let the quadratic objective function in Eq. (8.1) be denoted by

$$\phi(\mathbf{x}) = \tfrac{1}{2}\mathbf{x}^T \boldsymbol{A} \mathbf{x} + \mathbf{b}^T \mathbf{x}$$

The Legendre transform (or convex conjugate) of ϕ is then

$$\phi^*(\mathbf{y}) = \sup_{\mathbf{x}} \left(\mathbf{y}^T \mathbf{x} - \phi(\mathbf{x}) \right) = \tfrac{1}{2}(\mathbf{y} - \mathbf{b})^T \boldsymbol{A}^{-1} (\mathbf{y} - \mathbf{b}) .$$

Here and in the following, we use an asterisk superscript to denote a dual quantity. The Lagrangian[2] for Eq. (8.1) is

$$\mathcal{L}(\mathbf{x}, \lambda) = \phi(\mathbf{x}) + \lambda^T \boldsymbol{B}(\mathbf{x} + \mathbf{d})$$

Note that the last term in this expression can be thought of as a penalty function. Subject to the constraint that $\lambda \geq 0$ when the original constraints in Eq. (8.1) are not satisfied (i.e., $\boldsymbol{B}(\mathbf{x} + \mathbf{d}) > 0$), this term is always non-negative, and it is only zero if the constraints are satisfied.

From this Lagrangian, the dual objective function follows through a sequence of simple transformations, [Boyd and Vandenberghe, 2004, Equation 5.11] :

$$\begin{aligned}
g(\lambda) &= \inf_{\mathbf{x}} \mathcal{L}(\mathbf{x}, \lambda) \\
&= \inf_{\mathbf{x}} \Big(\phi(\mathbf{x}) + \lambda^T \boldsymbol{B}(\mathbf{x} + \mathbf{d}) \Big) \\
&= \lambda^T \boldsymbol{B}\mathbf{d} + \inf_{\mathbf{x}} \Big(\phi(\mathbf{x}) + \lambda^T \boldsymbol{B}\mathbf{x} \Big) \\
&= \lambda^T \boldsymbol{B}\mathbf{d} - \sup_{\mathbf{x}} \Big(-\phi(\mathbf{x}) - \lambda^T \boldsymbol{B}\mathbf{x} \Big) \\
&= \lambda^T \boldsymbol{B}\mathbf{d} - \phi^*(-\boldsymbol{B}^T \lambda) \\
&= \lambda^T \boldsymbol{B}\mathbf{d} - \tfrac{1}{2}(-\boldsymbol{B}^T \lambda - \mathbf{b})^T \boldsymbol{A}^{-1}(-\boldsymbol{B}^T \lambda - \mathbf{b}) \\
&= \lambda^T \boldsymbol{B}\mathbf{d} - \tfrac{1}{2}(\boldsymbol{B}^T \lambda + \mathbf{b})^T \boldsymbol{A}^{-1}(\boldsymbol{B}^T \lambda + \mathbf{b}) \\
&= -\tfrac{1}{2}\lambda^T \boldsymbol{B}\boldsymbol{A}^{-1}\boldsymbol{B}^T \lambda - (\mathbf{b}^T \boldsymbol{A}^{-1} - \mathbf{d}^T)\boldsymbol{B}^T \lambda - \tfrac{1}{2}\mathbf{b}^T \boldsymbol{A}^{-1}\mathbf{b}
\end{aligned}$$

The dual problem is obtained by maximizing $g(\lambda)$ subject to $\lambda \geq 0$ or, equivalently, by minimizing $-g(\lambda)$. When doing this, the last constant term doesn't matter, so the dual problem becomes

$$\begin{aligned}
\min_{\lambda} \quad & \tfrac{1}{2}\lambda^T \boldsymbol{B}\boldsymbol{A}^{-1}\boldsymbol{B}^T \lambda + (\boldsymbol{A}^{-1}\mathbf{b} - \mathbf{d})^T \boldsymbol{B}^T \lambda \, . \\
\text{s.t.} \quad & \lambda \geq 0
\end{aligned} \tag{8.8}$$

Using the same substitutions as in Eq. (8.2), but with the update to include the friction forces, we first note that

$$\begin{aligned}
\boldsymbol{A}^{-1}\mathbf{b} - \mathbf{d} &= \boldsymbol{D}^{(i)} \Big(f^{(i)} - \boldsymbol{T}\beta^{(i+1)} \Big) - \delta^{(i)} \\
&= -\Big(\delta^{(i)} - \boldsymbol{D}^{(i)} f^{(i)} + \boldsymbol{D}^{(i)} \boldsymbol{T}\beta^{(i+1)} \Big) \\
&= -\Big(\delta^{(i)} + d\delta_u^{(i)} + \boldsymbol{D}^{(i)} \boldsymbol{T}\beta^{(i+1)} \Big) \\
&= -\Big(\delta_u^{(i+1)} + \boldsymbol{D}^{(i)} \boldsymbol{T}\beta^{(i+1)} \Big) \, .
\end{aligned}$$

[2] Not to be confused with the Lagrangian for our mechanical system.

Inserting this in Eq. (8.8) then gives us

$$\begin{aligned} \alpha^{(i+1)} = \underset{\alpha}{\text{argmin}} \quad & \tfrac{1}{2}\alpha^T \boldsymbol{N}^T \boldsymbol{D}^{(i)} \boldsymbol{N}\alpha + \alpha^T \boldsymbol{N}^T \left(\delta_u^{(i+1)} + \boldsymbol{D}^{(i)} \boldsymbol{T}\beta^{(i+1)}\right) . \\ \text{s.t.} \quad & \alpha \geq 0 \end{aligned} \tag{8.9}$$

8.5 Coupled QPs for contact and friction

Summarizing the results so far, we have two coupled QPs, one for the contact forces given by Eq. (8.9) and one for the friction forces given by Eq. (8.6). Restating both of them here for easy comparison, we see that they are very similar :

$$\begin{aligned} \alpha^{(i+1)} = \underset{\alpha}{\text{argmin}} \quad & \tfrac{1}{2}\alpha^T \boldsymbol{N}^T \boldsymbol{D}^{(i)} \boldsymbol{N}\alpha + \alpha^T \boldsymbol{N}^T \left(\delta_u^{(i+1)} + \boldsymbol{D}^{(i)} \boldsymbol{T}\beta^{(i+1)}\right) , \\ \text{s.t.} \quad & \alpha \geq 0 \end{aligned} \tag{8.10}$$

$$\begin{aligned} \beta^{(i+1)} = \underset{\beta}{\text{argmin}} \quad & \beta^T \boldsymbol{T}^T \boldsymbol{D}^{(i)} \boldsymbol{T}\beta + \beta^T \boldsymbol{T}^T \left(\delta_u^{(i+1)} + \boldsymbol{D}^{(i)} \boldsymbol{N}\alpha^{(i+1)}\right) . \\ \text{s.t.} \quad & \|\beta_k\| \leq \mu\alpha_k, \quad 1 \leq k \leq m \end{aligned} \tag{8.11}$$

Both of these QPs can be interpreted as projections onto convex sets (given by the feasible regions), and the staggered projection algorithm proceeds by alternating between solving these two QPs. Due to the nonlinearity of the global problem, the projections here are in compliance norm rather than in the inverse mass norm as originally stated in [Kaufman et al., 2008], but otherwise they are identical.

Each iteration provides a better estimate for $\alpha^{(i+1)}$ and $\beta^{(i+1)}$. The process starts by solving for $\alpha^{(i+1)}$ using either zero as an initial guess or the value from the previous Newton iteration or even the previous time step.

Unfortunately, despite the elegance of this formulation, the actual implementation of a solver based on these equations is difficult for the problems we are interested in. While the stiffness matrix is usually sparse, the same is not true for the Delassus operators in Eq. (8.10) and Eq. (8.11) (i.e., $\boldsymbol{N}^T \boldsymbol{D}^{(i)} \boldsymbol{N}$ and $\boldsymbol{T}^T \boldsymbol{D}^{(i)} \boldsymbol{T}$). Simply forming the QPs can therefore be difficult.

In the original work by Kaufman et al. [2008], the largest example considered had less than 2000 degrees of freedom, so there it was still feasible to form $\boldsymbol{D}^{(i)}$ explicitly. Even if it was dense. For large scale cloth simulation, this is no longer the case.

8.6 Iterative contact solver (ADONIS)

Rather than attempting to solve for *all* contact forces at once and then for *all* friction forces at once, we propose a different approach here. In particular, we focus on how to solve these systems within the popular framework of iterative contact-collision methods, specifically Gauss-Seidel methods.

We extend the standard contact iteration approach by observing that we can iterate through the full set of contacts in multiple Gauss-Seidel passes, while updating the discrete compliance matrix until a convergence criterion is satisfied. The key here is that the Gauss-Seidel passes happen *inside* the Newton iterations rather than the other way around. This ensures fine-grained global coupling between all the contacts, which is essential for a stable response.

To facilitate the discussion, we organize the components of the collision response per contact, k, in

$$\lambda_k = (\alpha_k, \beta_k^T)^T \in \mathbb{R}^3.$$

Similarly, we denote the full contact basis by

$$\boldsymbol{G}_k = \begin{bmatrix} \boldsymbol{N}_k & \boldsymbol{T}_k \end{bmatrix} \in \mathbb{R}^{n\times 3}$$

Concatenating the total response in $\lambda \in \mathbb{R}^{3m}$ and the response basis in $\boldsymbol{G} = (\boldsymbol{G}_1, ..., \boldsymbol{G}_m) \in \mathbb{R}^{n\times 3m}$, the total discrete frictional contact force is $\boldsymbol{G}\lambda$.

With the above discretizations, the combined satisfaction of Eq. (SFC) and Eq. (DMD) per contact k is equivalently determined by seeking a λ_k satisfying the inclusion

$$\lambda_k \in \mathcal{R}_k(\delta) \;\Leftrightarrow\; \begin{cases} \min\limits_{\beta_k} \{\beta_k^T \boldsymbol{T}_k^T \delta \;:\; \|\beta_k\| \le \mu_k \alpha_k\}, \\ 0 \le \alpha_k \perp \mathbf{n}_k^T \delta \ge 0. \end{cases}$$

Here, we use $\mathcal{R}_k$ to denote the solution set to the single-point frictional contact problem.

In general, the stencils of contacts' generalized friction and normal bases overlap. Thus, we have to consider the globally coupled, joint satisfaction of Eq. (SFC) and Eq. (DMD) over all contacts. This is denoted by a total response λ satisfying the inclusion

$$\lambda \in \mathcal{R}(\delta) \;\Leftrightarrow\; \lambda_k \in \mathcal{R}_k(\delta) \quad \forall k \in \{1, \dots, m\}, \tag{8.12}$$

where $\mathcal{R}$ is the set of all response solutions globally satisfying the frictional contact conditions, [Stewart, 2011].

For convenience, we designate $\bar{\boldsymbol{G}}_k$ and $\bar{\lambda}_k$, respectively, as the complement matrix and vector formed by zeroing out the three columns in $\boldsymbol{G}$ and the three entries in λ corresponding to the frame basis vectors of contact k. We then overload addition for complement pairs to ensure consistent block-vector addition so that $\lambda = \lambda_k + \bar{\lambda}_k$.

To begin, we then first note from Eq. (7.5) and Eq. (7.6) that the displacement at the next Newton iterate is given by

$$\delta^{i+1} = \delta_u^{i+1} + \boldsymbol{D}^i[\boldsymbol{G}_k\lambda_k^{i+1} + \bar{\boldsymbol{G}}_k\bar{\lambda}_k^{i+1}],$$

where δ_u^{i+1} can be precomputed at the beginning of each Newton iteration.

To solve Eq. (8.12) at *each Newton iteration* i, we iterate in Gauss-Seidel fashion *per contact* k, solving directly for λ_k^{i+1} (and thus implicitly for δ_k^{i+1}) by solving

$$\lambda_k^{i+1} \in \mathcal{R}_k(\delta_u^{i+1} + \boldsymbol{D}^i[\boldsymbol{G}_k\lambda_k^{i+1} + \bar{\boldsymbol{G}}_k\bar{\lambda}_k^{i+1}]) \tag{8.13}$$

and then setting $\lambda^{i+1} \leftarrow \bar{\lambda}_k^{i+1} + \lambda_k^{i+1}$.

Each such iteration subproblem can most usefully be viewed as simply solving a single point frictional contact problem, where $\delta_u^{i+1} + \boldsymbol{D}^i\bar{\boldsymbol{G}}_k\bar{\lambda}_k^{i+1}$ is the predicted displacement and λ_k^{i+1} is the contact's unknown response force. As such, (8.13) can then be solved or approximated by a plethora of available numerical methods customized for the resolution of systems subject to a single contact. Examples of such methods include those proposed by Alart and Curnier [1991]; Stewart [2001]; Duriez et al. [2006]; Bonnefon and Daviet [2011]; Bertails-Descoubes et al. [2011]; Stewart [2011]. In our implementation, we employ the robust hybrid solver of Daviet et al. [2011], for which code has been publicly released.

Our approach effectively applies a simple extension of the standard contact iteration by Jean and Moreau [1992] by observing that we can iterate through the full contacting system in multiple Gauss-Seidel passes while updating the discrete compliance matrix until convergence.

After each Gauss-Seidel solve, we update the displacement estimate :

$$\delta^{i+1} \leftarrow \delta_u^{i+1} + \boldsymbol{D}^i\boldsymbol{G}\lambda^{i+1}.$$

The convergence criterion for the Newton solver is then tested and, if too large, we update the nonlinear terms and proceed to solve the next Newton iteration by multiple Gauss-Seidel passes.

The resulting ADONIS method is summarized in Algorithm 3 below.

Algorithm 3: *ADONIS*

Input: $\mathbf{q}^t, \dot{\mathbf{q}}^t, \tau$

$\delta \leftarrow$ `solve DEL`$(\mathbf{q}^t, \dot{\mathbf{q}}^t, \tau)$ ▷ Initial unconstrained solve
$\mathbb{K} \leftarrow$ `CollisionDetection`$(\delta, \mathbf{q}^t)$
while not converged **do** ▷ Newton loop
  $\boldsymbol{D} \leftarrow [\boldsymbol{M} + \tau^2 \boldsymbol{H}(\mathbf{q}^t + \delta)]^{-1}$
  $\delta_u \leftarrow \delta - \boldsymbol{D}[\boldsymbol{M}\delta - \tau \boldsymbol{M}\dot{\mathbf{q}}^t + \tau^2 \nabla W_{pot}(\mathbf{q}^t + \delta)]$
  `gs_itr` $\leftarrow 0$
  while `contact_err` $>$ `contact_tol` & `gs_itr` $<$ `gs_max` **do** ▷ GS loop
    for k in $\mathbb{K}$ **do**
      $\lambda_k \leftarrow$ `solve:` $\lambda_k \in \mathcal{R}_k(\delta_u + \boldsymbol{D}[\boldsymbol{G}_k \lambda_k + \bar{\boldsymbol{G}}_k \bar{\lambda}_k])$ ▷ Single point contact solve
      $\lambda \leftarrow \bar{\lambda}_k + \lambda_k$
    end for
    `gs_itr` $\leftarrow$ `gs_itr` $+ 1$
  end while
  $\delta \leftarrow \delta_u + \boldsymbol{D}\boldsymbol{G}\lambda$.
end while
$\dot{\mathbf{q}}^{t+1} \leftarrow \frac{1}{\tau}\delta$
$\mathbf{q}^{t+1} \leftarrow \mathbf{q}^t + \delta$
return $(\mathbf{q}^{t+1}, \dot{\mathbf{q}}^{t+1})$

To evaluate the algorithm, we start by considering a number of algorithmic simplifications specific to hair. In particular, we focus on thin elastic rods (hair) using the discrete elastic rod model, [Bergou et al., 2008, 2010]. In the next chapter, we then show that the resulting algorithm is stable for substantially larger time steps than existing algorithms, while capturing important visual characteristics of hair due to the accurate handling of both contact and friction. Ultimately, this leads to the important conclusion that it is essential to deal with the nonlinearity inherent in collision response for thin objects like both hair and cloth.

8.7 Contact groups

For hair simulation, each hair strand represents a separable domain in the absence of contact. Thus, each strand can be integrated independently of other hair strands. In the presence of contacts, at each time step, we decompose the contact graph formed by detected contacts into its connected components. Each of these maximally connected subgraphs similarly forms an independent system, a *contact group*, that we can integrate separately, independently, and (when computational resources allow) in parallel.

For clarity of presentation and the avoidance of further subscripting, we do not include this detail in our pseudocode. However, the chief difference is simply that each contact group (or "island") independently follows the pseudocode after collision detection and is computed in parallel.

In the following, the size of a contact group is measured as the number of collisions that contribute to the group. In the preceding text, this has been referred to by the variable m.

For cloth simulation, the contacts can still be partitioned into independent groups, but since the cloth itself cannot easily be broken into many independent regions, the benefit in this case is less significant.

8.8 Application of the compliance matrix

Step 4 in ADONIS requires computation of the compliance matrix, which as noted before is generally dense. However, we only need the action of the compliance matrix when applied to a vector. This can be implemented reasonably efficiently as long as the set of such vectors is not too large.

For hair simulation, the situation is simplified by the fact that the Hessian associated with each strand is banded (due to the 1D nature of a strand of hair). Consequently, the LU decomposition of the Hessian is also banded, [Golub and Loan, 1983], and can be stored at the same cost as the Hessian. Specifically, for discrete elastic rods, the energy Hessian is a banded matrix with a width of eleven. We can therefore employ a banded LU factorization method `dgbtrf`[3] for LU factorization of these Hessians. After this factorization, the application of the compliance matrix to a vector amounts to a forward and backward substitution.

This process comes into play in the Gauss-Seidel loop where we repeatedly need to compute $\boldsymbol{DG}_k$ for all k. However, none of these quantities change between Gauss-Seidel iterations, so they can all be precomputed as shown in Algorithm 4.

The potential issue in this context is that while $\boldsymbol{G}_k$ is sparse, $\boldsymbol{DG}_k$ is dense. Thus, even though we don't have to store the dense compliance matrix explicitly, we do have to store the filtered contact basis vectors, and these vectors are dense. This cost obviously scales linearly in the total number of contacts, but it should also be noted that the dimension of the contact basis vectors scales with the size of the associated contact group. Thus large

[3] `http://www.netlib.org/lapack/double/dgbtrf.f`

contact groups can require a substantial amount of memory for storing the filtered contact basis vectors. We revisit this issue in Section 9.2.3.

Algorithm 4: *AssembleCompliance*

Input: $\mathbb{K}$

for k in $\mathbb{K}$ **do**
  $\{r, s\} \leftarrow$ `GetContactingObjectsIndices`(k)
  if $r = s$ or s is a collision mesh **then** ▷ Self collision or mesh collision
    $\mathbf{P}_k \leftarrow$ `BlockLUSolve`$(r, \boldsymbol{L}_r, \boldsymbol{U}_r, \boldsymbol{G}_k)$
  else ▷ Hair-hair collision
    $\mathbf{P}_k \leftarrow$ `BlockLUSolve`$(r, \boldsymbol{L}_r, \boldsymbol{U}_r, \boldsymbol{G}_k)$
    $\mathbf{P}_k \leftarrow$ `BlockLUSolve`$(s, \boldsymbol{L}_s, \boldsymbol{U}_s, \mathbf{P}_k)$
  end if
end for
return $\mathbf{P} = \{\mathbf{P}_1, ..., \mathbf{P}_m\}$

8.9 Termination

Motivated by our observations in Section 7.1, we use the *geometric* objective of reducing non-physical stretch as our convergence criterion. For close to inextensible materials such as hair, we judge whether "sufficient nonlinearity" has been applied by considering axial strain. At iteration i, we compute a geometric ∞-norm measure of the maximum stretch-factor over all edges $\boldsymbol{e}_{r,j}$ belonging to rods indexed by r :

$$\mathrm{sf}^i = \max_{r,j} \; |\boldsymbol{e}^i_{r,j}| / |\bar{\boldsymbol{e}}_{r,j}| - 1,$$

where $\bar{\boldsymbol{e}}$ indicates edge rest length.

We adaptively stop each time step solve when a sufficient reduction in stretch indicates that the collision response has been communicated across the domain. Note that standard practice suggests $\mathrm{sf}^i < 10^{-1}$, [Bridson et al., 2002] . For stiff, but not close to inextensible materials, we could instead apply a bound on a discrete strain-rate measure :

$$\mathrm{sr}_i = \max_{j,k} \; (|\boldsymbol{e}^{\mathrm{i}}_{r,j}| - |\boldsymbol{e}^{\mathrm{t}}_{r,j}|)/h.$$

For cloth, we generally expect the two orthotropic directions corresponding to warp and weft to be near inextensible. This means that the directional strain in those two directions should be low while the shear strain may potentially be large. A simple measure for cloth is therefore to look at $J_3^2 = \mathrm{tr}^2(\boldsymbol{AEA})$ since $\mathrm{tr}(\boldsymbol{AEA})$ is the sum of the eigenvalues of $\boldsymbol{AEA}$, i.e.,

the principal strains of the directional strain tensor. As we have seen in Section 4.5.2, this is cheap to compute.

Within each Newton iteration, the Gauss-Seidel loop is terminated whenever the ∞-norm over all single point contact subproblem residuals is below a chosen threshold. With our choice of using the method by Daviet et al. [2011] for the single point contact solves, this residual is given by the Fischer-Burmeister functional for the subproblem. In practice, we use $\texttt{contact\_tol} = 10^{-6}$.

8.10 A localized modified-Newton strategy

Consider now that each of the above Newton iterations is applied with the specified goal of building a better distribution of contact responses to reduce stretch. At the start of each such Newton iteration, we thus restrict our update of Hessian terms to rods within each contacting system whose stretch is larger than the requested tolerance. All other Hessians are left unchanged from the prior iteration. This *localized* modified-Newton step allows us to focus computation on improving nonlinear terms of domains that are poorly modeled while the following Gauss-Seidel iterations then redistribute forces and displacements over the whole contacting system via the improved model. This localization is effectively what allows us to **A**dapt the **D**egree **O**f **N**onlinearity in the **I**mpact **S**olve. I.e., it is a one of the key ingredients in ADONIS.

This approach could also be used with a different strategy for the contact solve (e.g. staggered projections), but it depends critically on being able to partition the system into independent subsystems. Thus, this particular aspect of the algorithm does not extend easily to cloth.

8.11 Collision detection

The collision detection used for the evaluation in the next chapter is based on either proximity detection at the beginning of the time step or continuous time collision detection (CCD) as described in Section 7.6. For hair-mesh collisions, we always use CCD. The corresponding collision constraints are also built as described in Section 7.6. However, we have to take care to compute the contribution from scripted objects correctly (see Eq. (7.12)).

Finally we note that broad-phase updates, narrow-phase collision detection

queries, and contact point processing to build constraints are all performed in parallel.

Algorithm 5: `CollisionDetection`

Input: $\delta, \mathbf{q}^t$
1: **if** `rod_ccd` **then**
2: $\quad \mathbb{K} \leftarrow$ rod-rod CCD on the trajectory from $\mathbf{q}^t$ to $\mathbf{q}^t + \delta$
3: **else**
4: $\quad \mathbb{K} \leftarrow$ rod-rod proximity collision-detection at $\mathbf{q}^t$
5: **end if**
6: $\mathbb{K} \leftarrow \mathbb{K} \cup$ rod-mesh CCD on the trajectory from $\mathbf{q}^t$ to $\mathbf{q}^t + \delta$
7: **return** $\mathbb{K}$

8.12 Choice of unconstrained guess

Finally, we should note that it would be cheaper to initialize each of our contact solves with the inexpensive solution to Eq. (LIE) rather than the more expensive, fully nonlinear guess we generate from solving Eq. (DEL). However, doing so introduces large jitters and popping to the simulation. These are especially distracting as simulations try to come to a rest. Moreover, the *unconstrained* solutions to Eq. (DEL) are not a bottleneck to compute for hair simulation, as each strand can be solved independently in parallel. For cloth simulation, the unconstrained solution is more expensive to solve since the whole system is typically coupled.

In both cases, the discussion from Chapter 6 applies here.

8.13 Summary

Combining all the various components presented in the previous sections, we end up with the pseudocode given in Algorithm 6 for applying ADONIS to hair or rod simulations.

Algorithm 6: *ADONIS for hair/rods*

for r in $\{1, ..., \ell\}$ **do** in parallel
  $\delta_r \leftarrow$ `solve DEL`$(\mathbf{q}_r^t, \dot{\mathbf{q}}_r^t, \tau)$
  $\boldsymbol{H}_r \leftarrow \boldsymbol{H}_r(\mathbf{q}_r^t + \delta_r)$
  $\{\boldsymbol{L}_r, \boldsymbol{U}_r\} \leftarrow$ `LUFactorize`$\left(\boldsymbol{M}_r + \tau^2 \boldsymbol{H}_r\right)$
end for
$\mathbb{K} \leftarrow$ `CollisionDetection`$(\delta, \mathbf{q}^t)$
$\{\boldsymbol{G}, \mathbf{s}\} \leftarrow$ `BuildConstraints`$(\mathbb{K}, \delta, \mathbf{q}^t)$
$\mathbb{S} \leftarrow \mathbb{K}$
$\lambda \leftarrow$ `InitializeContactForce`$()$
while $\mathbb{S} \neq \emptyset$ **do**
  $\mathbf{P} \leftarrow$ `AssembleCompliance`$(\mathbb{S})$
  for x in $\{1, ..., \ell\}$ **do** in parallel
    $\mathbf{b}_r \leftarrow h\boldsymbol{M}_r \dot{\mathbf{q}}_r^t - h^2 \nabla V_r(\mathbf{q}_r^t + \delta_r) + h^2 \boldsymbol{H}_r \delta_r$
    $\delta_r \leftarrow$ `LUSolve`$(\boldsymbol{L}_r, \boldsymbol{U}_r, \mathbf{b}_r)$
  end for
  `gs_itr` $\leftarrow 0$
  while (`ContactError`$(\lambda) >$ `contact_tol`
    & `gs_itr` $<$ `gs_max`) **do**
    for k in $\mathbb{K}$ **do**
      $\mathbf{c} \leftarrow \sum_{j \in \mathbb{K} \neq k} \mathbf{P}_j \lambda_j$
      $\lambda_k \leftarrow$ `solve:` $\lambda_k \in \mathcal{R}_k(\delta + \mathbf{P}_k \lambda_k + \mathbf{c})$
    end for
    `gs_itr` $\leftarrow$ `gs_itr` $+ 1$
  end while
  $\delta \leftarrow \delta + \sum_{j \in \mathbb{K}} \mathbf{P}_j \lambda_j$
  $\mathbb{S} \leftarrow \emptyset$
  for r in $\{1, ..., \ell\}$ **do** in parallel
    if `Stretch`$(r) >$ `stretch_tol` **then**
      $\boldsymbol{H}_r \leftarrow \boldsymbol{H}_r(\mathbf{q}_r^t + \delta_r)$
      $\{\boldsymbol{L}_r, \boldsymbol{U}_r\} \leftarrow$ `LUFactorize`$\left(\boldsymbol{M}_r + \tau^2 \boldsymbol{H}_r\right)$
      $\mathbb{S} \leftarrow \mathbb{S} \cup$ `GetContactsForRod`(r)
    end if
  end for
end while
$\dot{\mathbf{q}}^{t+1} \leftarrow \frac{1}{h}\delta$
$\mathbf{q}^{t+1} \leftarrow \mathbf{q}^t + \delta$
return $(\mathbf{q}^{t+1}, \dot{\mathbf{q}}^{t+1})$

CHAPTER 9

Evaluation of contact solver

To understand the performance and behavior of the proposed contact solver, we ran a range of hair simulation examples over a variety of collision scenarios. The results are described in detail in this chapter. As part of this effort we wish to understand how the behavior of ADONIS compares with that of both standard zeroth-order, impulse response (obtained by setting $\boldsymbol{D}^i = \boldsymbol{M}^{-1}$) and linearly compliant response. Thus, we consider these methods below as well.

We first consider and quantify stability gains obtained by adaptive nonlinearity (Section 9.1). This leads us to an analysis of the runtime performance (Section 9.2.1), where we observe that contact solves, which scale in the number of contacts, dominate the cost of our simulations. Hence, we note that our algorithm likewise scales in the number of contacts processed.

However, this is only the beginning of the story—the number of contacts processed in each scene clearly affects the simulation output. In Section 9.2.3, we explore the tradeoff between the final cost and visual quality of simulations as a function of the number of contacts processed. We control the latter by considering variations in both the seeding density of rooted rods (rods/cm^2) and the number of contacts sampled by collision detection.

In our experiments, we vary seeding density by increasing the number of rods rooted over a fixed surface area. As we do so, we see that the number of contacts resolved grows in a superlinear fashion, suggesting that we face a severe computational challenge as we scale towards contacting rod assemblies at reported human hair densities.

For a fixed seeding density, we can also increase the number of contacts sampled at the cost of lengthening runtime. We investigate what the advantages of doing this are, and observe that increased contact sampling leads to the capture of more local features in our simulations.

9.1 Case Study 1: Single Rod Collisions

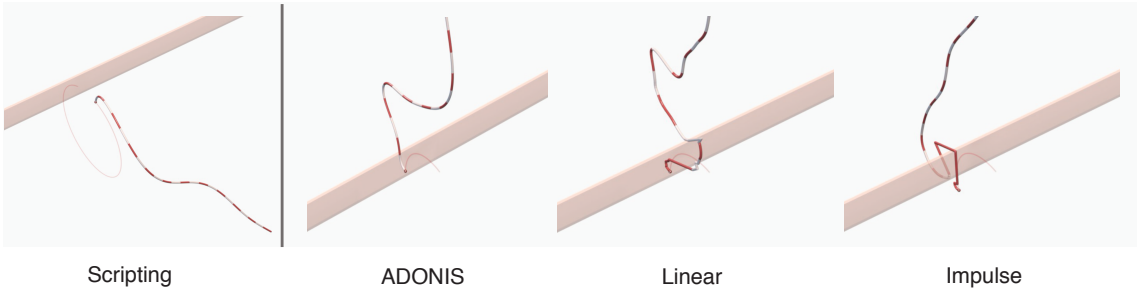


Figure 9.1: Whip-it stability test. *To test the relative stability of response methods we rotate a scripted handle connected to a rod so that it repeatedly whips the rod against the edge of a thin wall obstacle. At a time step size of 3 ms this results in, from left to right, a smoothly varying collision response from the proposed adaptive nonlinear algorithm, while the linearly compliant and impulse methods both obtain large, localized, non-physical deformations.*

We begin with a simple example to consider how the stability behavior of an isolated colliding rod varies as we change the collision response method. This example is designed specifically to exercise the worst-case scenario of repeated small-bend collisions to better understand the extremity of nonlinear collision response behavior. We rotate a scripted handle connected to an elastic rod so that it repeatedly and vigorously whips the rod against the edge of a thin wall obstacle. See Fig. 9.1 for set-up and simulation snapshots. The physical parameters for the rod used in this example are: material density $\rho = 1.3\ \mathrm{g/cm^3}$, elastic modulus $Y = 10^{10}\ \mathrm{g/(cm \cdot s^2)}$, shear modulus $G = 3.4 \times 10^9\ \mathrm{g/(cm \cdot s^2)}$, and radius $r = 60\ \mu\mathrm{m}$. For a cloth simulation the equivalent example would be a sharp object (like a finger) hitting the middle of a piece of the cloth.

To understand stability gains, we plot the stability regions of all three response algorithms over varying time step size and rotational whipping speed. To determine, stability we stipulate success as a completed run over a time-period of five seconds, during which axial extension does not exceed

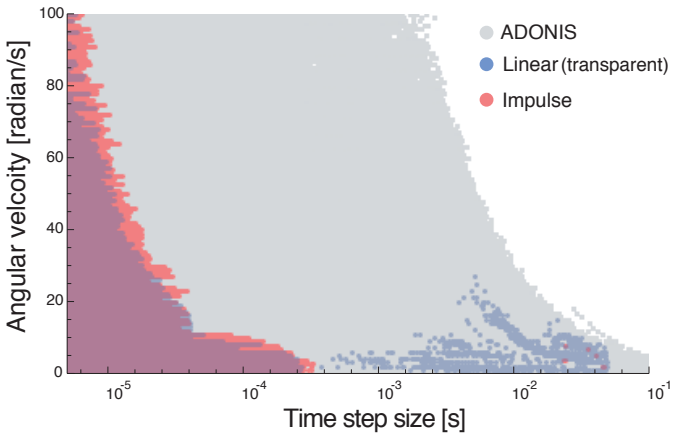


Figure 9.2: *To understand the overall stability behavior of these algorithms, we plot their respective stability regions as we vary time step (x-axis) in log-scale from 5 µs to 100 ms and rotational whipping speeds (y-axis). For each successful simulation, we plot a corresponding grey marker for the adaptive nonlinear method; transparent blue for linearly compliant response; and red for impulse response. Here, we observe a generally two orders of magnitude gain in maximum stable time step size for ADONIS.*

an unusually forgiving tolerance of 50% rest length (a typical tolerance for simulation examples is 1% rest length). Fig. 9.2 depicts the stable regions. Observe the gain of two orders of magnitude in maximum stable time step size for ADONIS. The stability regions of the linear and impulse response algorithms largely overlap, confirming that, in practice, for small-bend collisions, linear compliance obtains the same effective response as that of an unfiltered impulse.

9.2 Case Study 2: Hair Balls

We now shift our focus from a single to a dense assembly of rods. Starting with a sphere of roughly human head proportions (18 cm diameter), we uniformly seed curly rods over 50% of the surface, and script the sphere through a sequence of rotations about three orthogonal axes, alternating rotations with rest phases. An example of this with 16K rods is shown in Fig. 9.3. The scripting details are given in the bottom part of Fig. 9.4. This scenario exercises the rod assembly through a full range of tossing, tumbling, and spinning. Fast collisions are initiated at both the start and end of each rotation phase, and the pauses in between are sufficient for settling into slower contacting behavior.

We run our simulations in geometrically increasing sequences, ranging from 1K up to 64K rods in our largest example. This corresponds to a

Figure 9.3: *In this example, we simulate a "hairball" consisting of 16K rods affixed to a sphere scripted through a series of rapid rotations.*

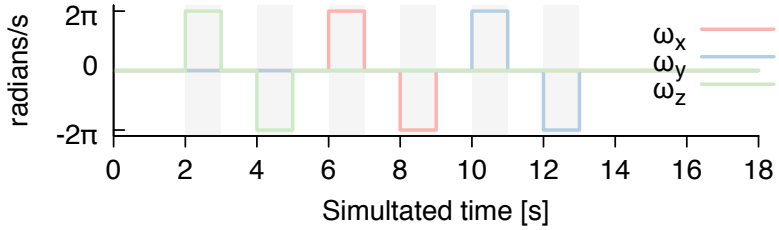


Figure 9.4: *All hair ball examples are scripted through a set of rotations around the three main axes, as shown here where ω represents the angular velocity around the specified axis.*

maximum seeding density of $\sim$ 125 rods/cm^2. In comparison, the average full head of human hair has $175 - 300$ hairs/cm^2, [Robbins, 2012]. We select reported human hair parameters for these experiments : material density $\rho = 1.32$ g/cm^3, viscosity $\mu_v = 5 \times 10^7$ g/(cm $\cdot$ s), gravity $g = 981$ cm/s^2, elastic modulus $Y = 3.9 \times 10^9$ g/(cm $\cdot$ s^2), shear modulus $G = 3.4 \times 10^9$ g/(cm $\cdot$ s^2), rod-rod frictional coefficient $\mu = 0.2$, rod-sphere frictional coefficient $\mu = 0.1$, and rod radius $r = 37$ μm.

In all the hair ball examples, each rod is discretized to 119 DoFs total, 30 vertices and an additional 29 twist DoFs per rod. Except where otherwise noted, we apply proximity based collision detection for rod-rod contact sampling and employ a time step of $\tau = 10$ ms. To better understand the effects of increased contact sampling, we applied two different proximity radii, 2.5 μm and 25 μm, respectively. As we explore below, varying proximity radii has some clear tradeoffs. In the following sections, based on the resulting behaviors we obtain, we distinguish between these two simulation types as respectively *smooth* and *tangled*. For the smooth and tangled simulations, we set `gs_max` to 1050 and 150 respectively. Here and in the following, the convergence criterion for ADONIS limits stretching to sf $< 1\%$.

Statistics for the smooth and tangled simulations were obtained respectively with Intel Xeon E5-4650 @ 2.7GHz (8 core Sandy Bridge-EP, 4 sockets) and Intel Xeon E5-2680v2 @ 2.8GHz (10 core Ivy Bridge-EP, 2 sockets) systems,

with the exception of the 32K tangled simulation, which ran on an Intel Xeon E5-2650 @ 2GHz (8 core Sandy Bridge-EP, 2 sockets).

9.2.1 Timing breakdown and scaling

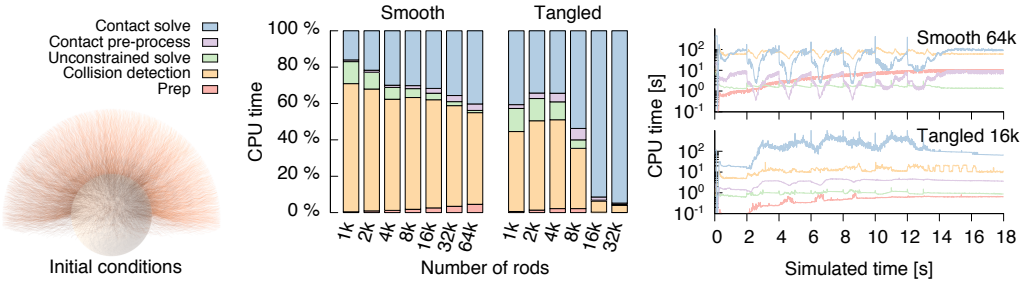


Figure 9.5: *Left: Initial conditions for the hair ball examples (8K rods). Middle: CPU breakdown statistics across all hair ball simulations as we increase the number of rods/seeding density. Right: CPU breakdown statistics by time step for the 64K rod smooth (top) and 16K rod tangled (bottom) simulations.*

First, we show the breakdown of CPU time for these simulations in Fig. 9.5, middle. The cost of collision resolution becomes dominant as we move towards increased contact sampling and higher rod densities. Indeed, for the most complex scenarios, collision resolution is dominant at every time step (see Fig. 9.5, right). Recall that, at each time step, we can have multiple contact problems, each corresponding to the resolution of an independent contact group. In turn, the total cost of contact resolution at each time step depends on the number and size of these groups. In Fig. 9.6, we plot the time required to solve individual contact groups as a function of the number of contacts in the group, across all groups encountered in the smooth (top) and tangled (bottom) simulation sequences, and observe close to linear scaling.

At each Newton iteration, we attempt to solve the contact problem in Eq. (8.12). A solution for this optimization should be expected to scale nonlinearly in the number of contact variables. Initially, it is therefore surprising that we observe linear scaling overall. However, recall that to solve Eq. (8.12) we employ the Gauss-Seidel solver which regularly saturates at its upper iteration limit. We conjecture that this saturation is the source of the observed linear scaling.

9.2.2 Sufficient nonlinearity

A key question to understand the algorithm's overall performance is exactly how much additional work is required when linear compliance is insuffi-

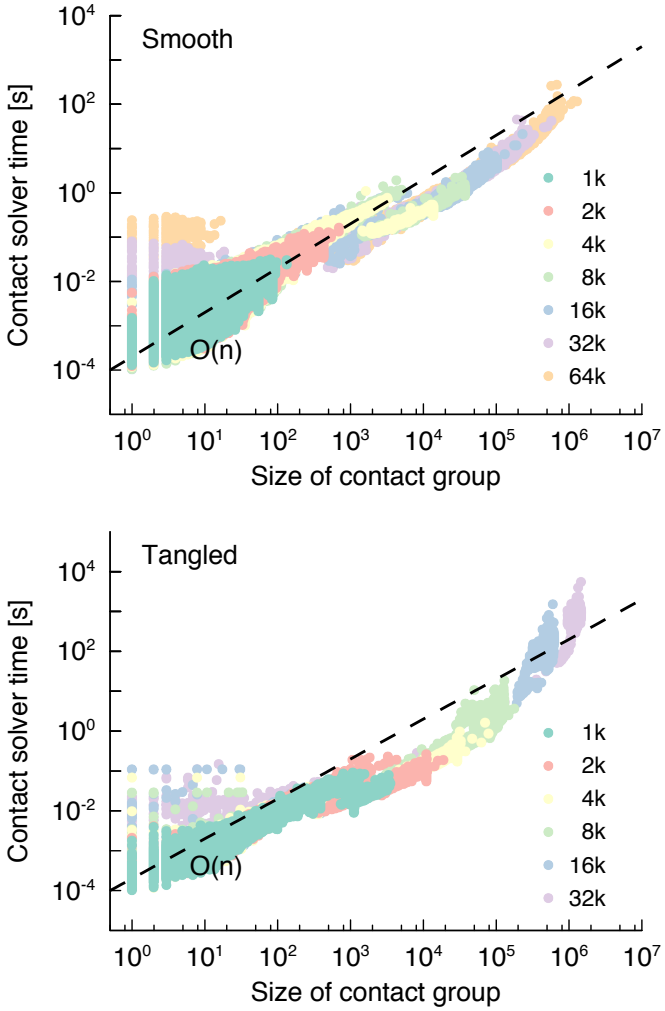


Figure 9.6: *Contact solver scaling statistics. Here, we plot the time required to solve a contact group, as a function of the number of contacts, across all groups encountered in the smooth (top) and tangled (bottom) simulation sequences. Linear scaling is plotted for reference.*

cient. Across our simulations, we find that the average number of constrained Newton iterations, per contact solve, remains close to one. However, this does not offer a sufficiently detailed view of how our algorithm adapts to nonlinearity over time. Nor does it help us understand the potential cost of the resulting additional Newton iterations. In Fig. 9.7, we instead plot the average number of constrained Newton iterations required by our algorithm per time step *weighted by contact problem size*. As we have seen above, larger contact problems are more costly, which is why the weighting is appropriate. We correspondingly observe here and in Fig. 9.8 that the vast

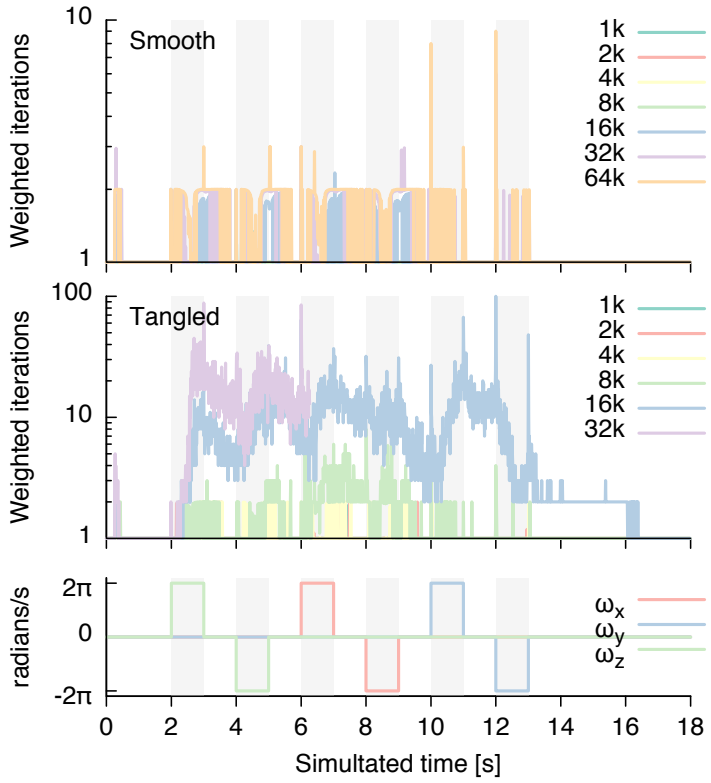


Figure 9.7: *Sufficient nonlinearity. To understand the potential cost of the resulting additional Newton iterations, we plot the average number of constrained Newton iterations applied by ADONIS, weighted by contact problem size per time step for smooth (top) and tangled (middle) simulations. Spikes in iteration counts correspond to instants in time when the scripting (bottom) exhibits discontinuities either by initiating new rotations or abruptly coming to a halt.*

majority of contact problems encountered require small numbers of iterations.

9.2.3 Turning the contact numbers knob

When trying to describe the complexity of a hair simulation, it is tempting simply to consider the number of hair strands. However, this can be very misleading; both in terms of computational cost and in terms of qualitative results. A more appropriate metric is the number of contacts processed. We explore this by considering variations in seeding density and contact sampling, and consider the tradeoffs between final runtime cost and the resulting visual characteristics.

Fig. 9.9 shows how the visual characteristics of the smooth hair ball exam-

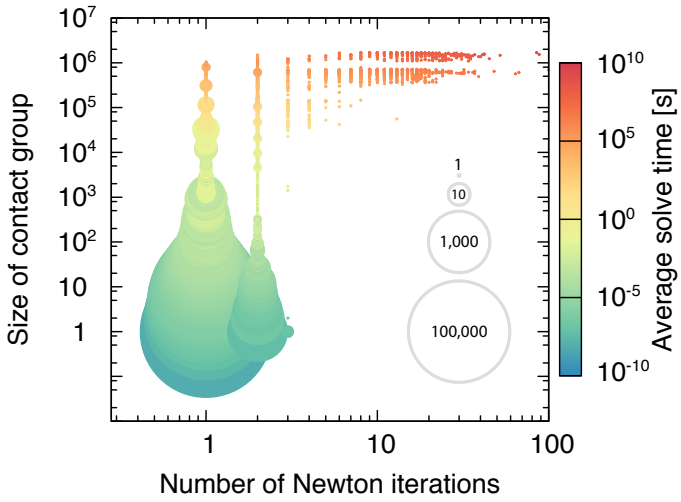


Figure 9.8: *The distribution of Newton iterations per contact group size and corresponding average solve times, over all hair ball simulation sequences. Disc areas indicate numbers of occurrences of problems at each group size and iteration count; grey circles on the right demonstrate the scaling between disc size and number. The color of all plotted discs shows the average solve time on a logarithmic scale, from which we note that the vast majority of problems are small and inexpensive to solve.*

Figure 9.9: *Increasing the number of hair strands has a relatively predictable effect on the overall appearance of the simulations.*

ples shifts as we vary the seeding density in the hair ball examples while keeping seed area fixed. Correspondingly, in Fig. 9.10, left, we show that the total number of contacts in the system grows superlinearly with the number of rods.

As noted above, however, simply counting hairs is insufficient to determine the number of contacts that must be processed. By itself, it also does not say much about the visual quality of the resulting simulation. In practice, we can vary the amount of contacts we detect by changing our rod discretization, our collision detection method, or even parameters within our collision detection method. Here, we consider the latter and look at the change in behavior and contact numbers for the two applied proximity radii we use in rod-rod collision detection. As we see in a side-by-side comparison in Fig. 9.11, increased contact sampling leads to the capture of local features, e.g., lock and ringlet-like structures. However, due to the extremely tight

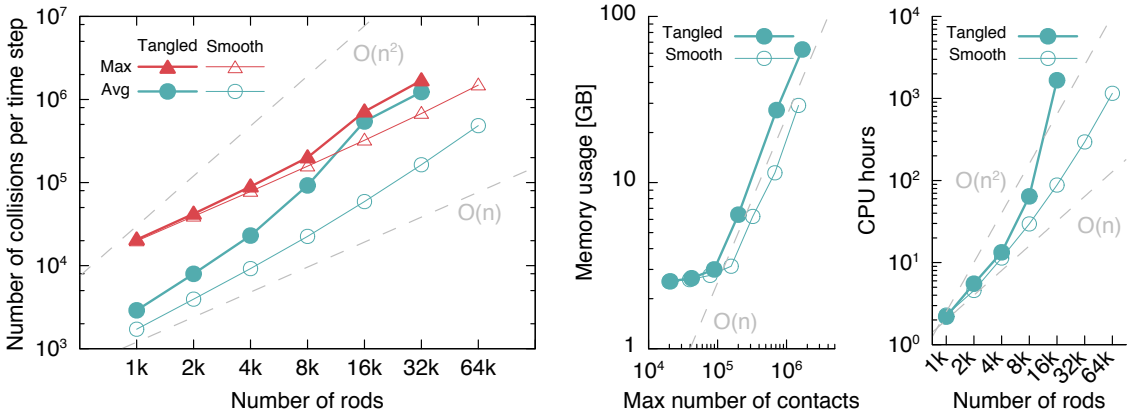


Figure 9.10: *Left: The number of contacts grows superlinearly in the number of rods/seeding density. Right: A summary of maximum memory utilization and CPU scaling across hair ball simulations.*

confines of these twists, this comes at the cost of a 10-fold and greater increase in the number of contacts we must process, which in turn leads to an increase in the total run time and memory usage (see Fig. 9.10, right.)

Figure 9.11: *Differing contact numbers and resulting simulated behaviors are obtained by changing the proximity radii we use in rod-rod collision detection from* 25 *µm (left) to* 2.5 *µm (right). Visually, this has quite a dramatic effect.*

9.2.4 Stability in rod assemblies

We saw above in Section 9.1 that, when stressed by high-speed collisions, ADONIS enables progress at stable time steps orders of magnitude larger than existing methods. However, it is not obvious if this stability advantage is maintained when we consider less violent motions in rod assemblies. To investigate this we consider the scripted hair ball example, but instrumented to examine the stability behavior of all three resolution methods as we vary both time step and seeding density. Since we wish to compare sta-

bility and performance across time step sizes, we employ CCD for rod-rod contact sampling. This ensures that larger time step simulations do not gain the unfair advantage of smaller contact groups due to missed collisions, or "tunneling."

rods	method	100ms	10ms	5ms	1ms	0.1ms
1K	ADONIS	x	3h42m	3h28m	4h09m	10h58m
1K	Linear	x	x	x	6h38m	11h09m
1K	Impulse	x	x	x	x	13h27m
2K	ADONIS	x	6h29m	3h46m	5h11m	23h49m
2K	Linear	x	x	x	8h30m	23h35m
2K	Impulse	x	x	x	x	28h51m
4K	ADONIS	x	43h55m	21h47m	19h40m	55h19m
4K	Linear	x	x	x	x	55h37m
4K	Impulse	x	x	x	x	76h34m

Figure 9.12: *A comparison of stable time step sizes and runtimes for response methods on the hair ball example. As we scale to larger seeding densities we see a stability gain for ADONIS of one to two orders of magnitude. Entries in the table give either the runtime to completion or an* x *to indicate a failed simulation. These simulations were all run on an Intel Xeon X5650 @ 2.67GHz (4 core Westmere-EP, 1 socket).*

We observe in Figure 9.12 that ADONIS is consistently stable at larger time step sizes in these examples, at least an order of magnitude larger than linear compliance and two orders of magnitude larger than impulsive response. *Furthermore, this increased stability does not impose additional cost*. Rather, we see that nonlinear adaptivity maintains the fastest runtimes across simulations when allowed to take large, stable time steps.

9.3 Case study 3: Combing, Flinging and Tangling

In the hair ball tests, we mostly considered rod assemblies at slower speeds (albeit with abrupt speed changes). Here, we start to stress ADONIS with a trio of examples, where the simulation of combing, flinging, and tangling behaviors are made possible by adaptive nonlinearity. In all of the following examples, we once again employ rod-rod CCD. The goal here is to show that the algorithm is robust even with quite challenging simulations and that the simulation cost remains reasonable.

9.3.1 Comb out

The combing stress test shown in Fig. 9.13 subjects thin rods to collisions and tangling that exercise the strongly nonlinear collision response of rods. We

Figure 9.13: *Collisions and tangling, as in this combing stress test, exercise the strongly nonlinear collision response of rods.*

comb through rods as they are tightly rotated and coiled about two rotors. Simulation details: $\tau = 4$ ms; 5,200 rods, 119 DoFs each; runtime: 24h28m on a Xeon E7-8870.

9.3.2 Debris fling

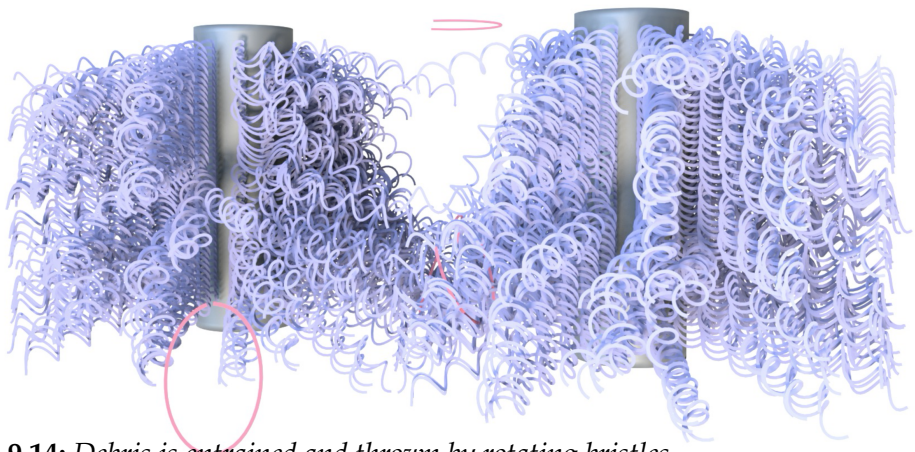


Figure 9.14: *Debris is entrained and thrown by rotating bristles.*

In the test shown in Fig. 9.14, dropped debris is entrained and thrown by

rapidly rotating stiff bristles. Simulation details: $\tau = 8$ ms; 1040 bristle rods, 399 DoFs each; 26 dropped rods, 79 DoFs each; runtime: 44m on a MacBook Pro 2011, Intel Core i7 @ 2GHz.

9.3.3 Rod catch

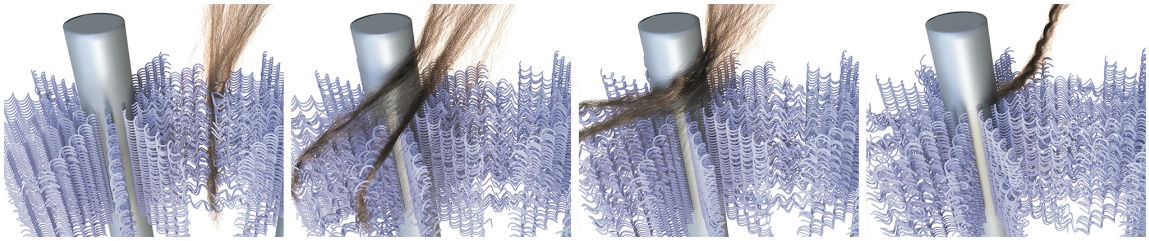


Figure 9.15: *Thin rods are caught and pulled into two separate hanks by stiff, rotating elastic bristles. The rods are wound about each other so that when pulled out they are braided together.*

In the last example, thin rods are caught and pulled into two separate hanks by stiff, rotating elastic bristles. The rods are then wound about each other so that, when pulled back out of the rotating bristles, they are braided together. See Fig. 9.15. Simulation details: $\tau = 8$ ms; 1040 bristle rods, 399 DoFs each; 1000 thin rods, 239 DoFs each; runtime: 30h14m on a Xeon E7-8870.

9.4 Limitations

The focus in this chapter has been on impact problems. Frictional contact problems in general are often broken into problems dealing with impact and those dealing with stable, persistent contact. For problems with persistent or slow moving contacts, the results above show that ADONIS retains comparable computational advantages of existing contact resolution methods. However, it also retains the same weaknesses : the Gauss-Seidel solver typically does not converge. We conjecture that this lack of convergence is responsible for a notable artifact. We observe that locks formed in our simulations may fall apart on their own, over time. As previously noted, the simulation of stable frictional contact assemblies depends on the accurate resolution of the underlying contact problem, [Kaufman et al., 2008]. We therefore suspect that the unraveling we observe could be addressed in future work by replacing the current Gauss-Seidel solver employed in steps 7-13 of ADONIS with a fully convergent solver for the contact and friction forces.

Another limitation of ADONIS is that it utilizes a relatively large amount of memory to resolve and store contact graphs during solves. The actual amount of memory used grows linearly in the number of contacts and thus superlinearly in seeding density. This issue was brought up in Section 8.8 and Fig. 9.10, right, shows that we do in fact observe this behavior. Depending on the available resources, this may or may not be a concern.

Finally, while the method proposed here generally offers robust stability for low cost, and is able to run our simulations at large time steps relative to existing methods, we have done nothing to address the problems outlined in Section 6.6. Thus the unconstrained problem remains ill-conditioned for large time steps, which ultimately limits the size of time steps in our simulations.

9.5 Summary

Considering the geometry of thin body collisions, we started in Chapter 7 by observing that the degree of nonlinearity in collision response for stretching modes varies greatly with configuration. Noting that capturing this nonlinearity is essential for the stable progress of simulations at practical time step sizes, we constructed a simple algorithm in Chapter 8 that applies first-order modeling in most solves. When linearity is sufficient, this proposed method incurs marginal cost over existing linearly compliant methods. In essence, it identifies instances where a frugal application of additional computation is most needed, which enables us to take time steps several orders of magnitude larger than previously possible. Importantly, this does not mean that *every* response is resolved with a small number of Newton iterations: as we have seen above in Section 9.2, in a *small but critical* subset of collision events, large numbers of iterations are necessary. Without these additional nonlinear iterations, the time step restriction becomes severe. The crucial point is that these instances are infrequent, and so when amortized across a simulation, we incur minimal additional cost.

CHAPTER 10

Multigrid methods

Having established in the previous chapters that nonlinearity is essential for solving contact problems with thin objects like hair and cloth, the next goal is to extend the contact solver from hair to cloth. As already noted, this presents a number of challenges.

The primary difficulty is that cloth problems do not allow us to easily partition the contact problem as we did in Section 8.7. As a consequence, the product of the compliance matrix, $\boldsymbol{D}$, and the contact basis, $\boldsymbol{G}$, forms a dense matrix in $\mathbb{R}^{n\times 3m}$, which can become prohibitive to store for systems with many contacts. In practice, such a situation easily arises if one has multiple layers of clothing (like a jacket on top of a shirt or a multilayer dress).

The secondary problem is that, without the partitioning, there is no easy way to adapt on the nonlinearity locally. As such, every nonlinear iteration must include the full system. Using Gauss-Seidel with its attendant slow convergence therefore becomes less attractive.

One potential approach for overcoming this is to reconsider the staggered projection algorithm presented in Chapter 8. The complication with that approach is the need to solve several large QPs subject to affine or even quadratic constraints.

There are multiple ways of solving large QPs, but many of them invariably involve solving large linear systems. This includes methods based on sequential quadratic programming using active sets as well as interior point methods. Thus, we are compelled to solve large linear systems efficiently.

One of the most appealing methods for solving such large linear systems is the multigrid method because its computational complexity is often linear in the number of unknowns. However, multigrid algorithms that obtain full computational efficiency can be difficult to design for new applications, especially when constraints are introduced.

In the remainder of this thesis, we focus on how to design such multigrid methods for cloth simulations. In particular, we develop the so-called *smoothed aggregation* (SA) method, [Míka and Vaněk, 1992], for cloth simulation. This is an algebraic multigrid (AMG) method, which eventually represents the foundation for solving large contact QPs. While the extension to QPs and the application to frictional contact problems remains outside the scope of this thesis, we show that this method can provide significant speedups for existing cloth simulation systems.

10.1 Multigrid basics

Conventional multigrid methods, whether geometric or algebraic, tend to perform poorly for thin-shell applications. Geometric methods furthermore require structured meshes, and existing multigrid approaches tend to have difficulty with collisions. To understand the difficulties inherent in cloth simulation and develop a more advanced algebraic multigrid method with improved performance, we begin by carefully considering the basic multigrid principles. Many of these principles are available in the basic tutorial presented in [Briggs et al., 2000], the additional material developed in [Trottenberg et al., 2000], and the theory given in [McCormick, 1984] and [Vassilevski, 2008].

Although multigrid methods have much broader applicability (including indefinite systems), our focus here is on the symmetric positive definite (SPD) case because it allows for a simpler development and clarification of the basic principles underlying multigrid methodology.

Every multigrid method has two essential components : *smoothing* and *coarsening*. The smoother is responsible for removing oscillatory errors (to be defined below) in the solution, while the coarsening strategy effectively provides a way to deal with smooth errors. However, in order for the multigrid scheme to work optimally, these two components must be carefully designed to complement each other.

Geometric multigrid (GMG) methods rely on the ability to coarsen a grid geometrically and to (explicitly or implicitly) define discretizations on coarser grids, as well as interpolation operators between the grids. Unfortunately,

geometric multigrid methods can be difficult to develop for problems with unstructured grids, complex geometries, and widely varying coefficients and anisotropies. As a convenient alternative to GMG methods, AMG and its cousin SA were developed to provide automatic processes for coarsening based solely on the target matrix. AMG coarsens a grid *algebraically* based on the relative size of the entries of the matrix to determine strong connections, thereby forming a hierarchy of grids from the finest, on which the original problem is defined, down to the coarsest, which typically consists of just a few degrees of freedom. The standard AMG coarsening process produces coarse grids whose degrees of freedom are subsets of those on the fine grid (represented by identity rows in the interpolation matrix). Thus, while AMG is an algebraic approach, a geometric representation of coarse-grid nodes in the continuum is still easily determined.

For linear finite element discretizations of Poisson's equation on regular 2D grids, the parameters for AMG can be selected to produce the usual geometric coarsening with linear interpolation. In this case, the coarse-grid matrix is essentially what finite elements would produce by re-discretization on the coarse grid. AMG and GMG solvers would then have similar interpolation, restriction, and coarse-grid components. It is thus often safe to make assumptions about the convergence of a standard GMG approach by looking at the convergence of an AMG implementation. Yet AMG can automatically produce effective coarse levels for many problems that do not lend themselves to geometric approaches.

Smoothed aggregation by contrast is an advanced aggregation-based method founded on algebraic multigrid principles. When coarsening the grid, these methods form agglomerates (by grouping fine-grid nodes) that each become a node of the coarse grid. The points that go into agglomerates are also formed based on relative strength between elements of the matrix. However, for standard SA, coarse nodes do not correspond to single fine-grid nodes. So, for vertex-centered discretizations, it is generally not possible to assign geometric meaning to the coarse grids that SA produces, especially for systems of PDEs. Smoothed aggregation also tends to coarsen more aggressively than AMG and GMG, so the coarse matrices and interpolation operators generally must work harder to obtain efficiency comparable to that of AMG and GMG.

To better understand the interplay between smoothing and coarsening, we start by characterizing smoothers in Section 10.2. Assuming that coarse grids and interpolation operators are given, we construct a basic multigrid scheme in Section 10.3. In Section 10.4, we analyze the requirements for these multigrid schemes to achieve good convergence and, based on this convergence

analysis, we go back and determine how to actually design effective coarse grids and interpolation operators in Section 10.5. Finally, in Section 10.7, we discuss how the discretization of the underlying problem can also affect the performance of multigrid methods.

10.2 Smoothing

Assume that the matrix $A^h \in \mathbb{R}^{n \times n}$ is SPD, where the generic "mesh" parameter h is used to indicate that it corresponds to our biggest or finest "grid". We use geometric terms here to simplify the discussion, although the concepts carry over to the use of algebraic multigrid methods when geometry is not necessarily assumed or even present. Given this terminology, our target fine-grid matrix equation is simply

$$A^h u^h = f^h, \tag{10.1}$$

where f^h is a given source term, and u^h is the unknown. Such linear systems can be solved using any of a number of simple iterative methods like Jacobi or Gauss-Seidel relaxation. For simplicity, we consider Richardson's iteration given by

$$v^h \leftarrow v^h - \frac{1}{\|A^h\|}\left(A^h v^h - f^h\right), \tag{10.2}$$

where $\|\cdot\|$ denotes the matrix norm induced by the Euclidean vector norm $\|\cdot\|$. (By virtue of our SPD assumption, $\|A^h\| = \max_{1 \le i \le n} \lambda_i$, where the λ_i are the eigenvalues of A^h.) By the notation in (10.2), we mean that the current approximation, v^h, to the solution, u^h, of (10.1) is replaced by the expression to the right of the arrow. Unfortunately, Richardson's iteration, as well as most other relaxation methods like Jacobi, Gauss-Seidel, and even conjugate gradients, usually stall well before they reach an acceptable approximation to the solution when applied to discretizations of partial differential equations. The cause of stalling comes from the residual's inability to see "smooth" error, by which we mean that the matrix applied to it yields a residual that is small compared to the error itself. Correcting smooth error by a term that involves the residual, as these methods do, would therefore accomplish very little. Worse yet, while these methods may work well for a couple of iterations when the initial error has oscillatory components, this fast elimination of oscillatory error exposes the remaining smooth error that works to stall all subsequent iterations. This limitation is a common smoothing property of most conventional iterative methods applied to discretizations of partial differential equations. However, as we shall see, this

smoothing property provides the motivation and foundation for the coarse-grid correction component of multigrid algorithms.

To clarify the source of this difficulty, we begin by studying the algebraic error, $\boldsymbol{e}^h \equiv \boldsymbol{v}^h - \boldsymbol{u}^h$, in more detail and we introduce ways to measure its (algebraic) smoothness. To this end, first consider the following simple relationship between the error and the residual :

$$\boldsymbol{A}^h\boldsymbol{e}^h = \boldsymbol{A}^h\left(\boldsymbol{v}^h - \boldsymbol{u}^h\right) = \boldsymbol{A}^h\boldsymbol{v}^h - \boldsymbol{f}^h.$$

If we use this relationship in (10.2) and subtract $\boldsymbol{u}^h$ from both sides of the result, we end up with the error propagation expression

$$\boldsymbol{e}^h \leftarrow \boldsymbol{G}^h\boldsymbol{e}^h, \tag{10.3}$$

where $\boldsymbol{G}^h = \boldsymbol{I} - \frac{1}{\|\boldsymbol{A}^h\|}\boldsymbol{A}^h$. $\boldsymbol{G}^h$, which is called the error propagation matrix because it governs how the iteration transforms the error, facilitates our analysis of the relaxation process that follows. Letting $\langle\cdot,\cdot\rangle$ denote the Euclidean inner product, we note that Richardson iteration is convergent because, we have that $0 < \langle\frac{1}{\|\boldsymbol{A}^h\|}\boldsymbol{A}^h\boldsymbol{e}^h, \boldsymbol{e}^h\rangle \leq \langle\boldsymbol{e}^h, \boldsymbol{e}^h\rangle$, from which it follows that $\|\boldsymbol{G}^h\| < 1$.

For a given error, $\boldsymbol{e}^h$, two quantities are particularly useful for measuring its algebraic smoothness :

$$\mathcal{M}_w(\boldsymbol{e}^h) = \frac{\langle\boldsymbol{A}^h\boldsymbol{e}^h, \boldsymbol{e}^h\rangle}{||\boldsymbol{A}^h||\langle\boldsymbol{e}^h, \boldsymbol{e}^h\rangle}$$

and

$$\mathcal{M}_s(\boldsymbol{e}^h) = \frac{\langle\boldsymbol{A}^h\boldsymbol{e}^h, \boldsymbol{A}^h\boldsymbol{e}^h\rangle}{||\boldsymbol{A}^h||\langle\boldsymbol{A}^h\boldsymbol{e}^h, \boldsymbol{e}^h\rangle}.$$

The *weak* measure, $\mathcal{M}_w(\boldsymbol{e}^h)$, is just the Rayleigh quotient of $\boldsymbol{e}^h$ relative to $\|\boldsymbol{A}^h\|$, and the *strong* measure, $\mathcal{M}_s(\boldsymbol{e}^h)$, is the relative Rayleigh quotient of $\left(\boldsymbol{A}^h\right)^{\frac{1}{2}}\boldsymbol{e}^h$. In either case, we call errors with a large measure *algebraically oscillatory* (or simply oscillatory), while errors with small measure are called *algebraically smooth* (smooth).

When $\boldsymbol{A}^h$ is symmetric, as we have assumed here, any error can be written as a linear combination of the eigenvectors, $\{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n\}$, of $\boldsymbol{A}^h$: $\boldsymbol{e}^h = \sum_{i=1}^n \alpha_i\boldsymbol{v}_i$. Smooth vectors are rich in eigenvectors associated with the low part of the spectrum, while oscillatory vectors are rich in the high part. This is the basis for using the terms *smooth* and *oscillatory*. For example, if the error is exactly an eigenvector, so that $\boldsymbol{e}^h = \boldsymbol{v}_i$ associated with eigenvalue λ_i, then $\mathcal{M}_w(\boldsymbol{e}^h)$ = $\mathcal{M}_s(\boldsymbol{e}^h)$ = $\frac{\lambda_i}{\|\boldsymbol{A}^h\|}$. Since $\|\boldsymbol{A}^h\| = \max_{1\leq i\leq n}\lambda_i$, then this measure assesses

how large λ_i is relative to its largest possible value. For many applications, this value can be very small at the lower end of the spectrum (e.g., $O(h^2)$ for discrete elliptic operators). For general linear combinations of eigenvectors, these measures judge how relatively small or large the coefficients, α_i, are for the opposite parts of the spectrum.

The weak and strong measures are important because they identify errors that cannot be reduced by relaxation and therefore must be reduced by coarse-grid correction. To begin to see this, note that relaxation reduces the current error, $\boldsymbol{e}^h$, in the Euclidean norm by the factor $\|\boldsymbol{G}^h\boldsymbol{e}^h\|/\|\boldsymbol{e}^h\|$. Using the definition of the matrix norm to conclude that

$$0 \leq \langle \boldsymbol{A}^h\boldsymbol{e}^h, \boldsymbol{A}^h\boldsymbol{e}^h\rangle \leq \langle \boldsymbol{A}^h\boldsymbol{e}^h, \boldsymbol{e}^h\rangle \|\boldsymbol{A}^h\|,$$

it then follows that

$$\begin{aligned}\frac{\|\boldsymbol{G}^h\boldsymbol{e}^h\|^2}{\|\boldsymbol{e}^h\|^2} &= 1 - 2\frac{\langle \boldsymbol{A}^h\boldsymbol{e}^h, \boldsymbol{e}^h\rangle}{\|\boldsymbol{A}^h\|\|\boldsymbol{e}^h\|^2} + \frac{\langle \boldsymbol{A}^h\boldsymbol{e}^h, \boldsymbol{A}^h\boldsymbol{e}^h\rangle}{\|\boldsymbol{A}^h\|^2\|\boldsymbol{e}^h\|^2} \\ &= 1 - (1+\xi)\mathcal{M}_w(\boldsymbol{e}^h),\end{aligned}$$

where $\xi \in]0,1]$. This confirms that relaxation slows *in the Euclidean norm* if and only if the *weak measure* is small. Let $\langle\cdot,\cdot\rangle_{\boldsymbol{A}^h} = \langle \boldsymbol{A}^h\cdot,\cdot\rangle$ and $\|\cdot\|_{\boldsymbol{A}^h} \equiv \sqrt{\langle\cdot,\cdot\rangle_{\boldsymbol{A}^h}}$ denote the energy inner product and its induced norm, respectively. Then an argument similar to the above shows that

$$\frac{\|\boldsymbol{G}^h\boldsymbol{e}^h\|_{\boldsymbol{A}^h}^2}{\|\boldsymbol{e}^h\|_{\boldsymbol{A}^h}^2} = 1 - (1+\chi)\mathcal{M}_s(\boldsymbol{e}^h),$$

where $\chi \in]0,1]$. Thus, relaxation slows *in the energy norm* if and only if the *strong measure* is small.

This correspondence between weak vs. strong measures and Euclidean vs. energy norms carries over to the analysis of coarse-grid correction. As shown in what follows, if the coarse grid adequately approximates errors for which $\mathcal{M}_s(\boldsymbol{e}^h)$ is small, then the so-called multigrid *V-cycles* converge well in the energy norm. For the so-called *two-grid* or *W-cycles* in the Euclidean norm, it is enough to have the coarse grid adequately approximate errors for which $\mathcal{M}_w(\boldsymbol{e}^h)$ is small. This latter requirement is weaker in part because small $\mathcal{M}_s(\boldsymbol{e}^h)$ implies small $\mathcal{M}_w(\boldsymbol{e}^h)$ (which follows because $\langle \boldsymbol{A}^h\boldsymbol{e}^h, \boldsymbol{e}^h\rangle^2 \leq \langle \boldsymbol{A}^h\boldsymbol{e}^h, \boldsymbol{A}^h\boldsymbol{e}^h\rangle\langle \boldsymbol{e}^h, \boldsymbol{e}^h\rangle$), but the converse is not true. More importantly, what we meant by adequate approximation for the strong measure, $\mathcal{M}_s(\boldsymbol{e}^h)$, involves the energy norm, which is a stronger requirement than that for the Euclidean norm used with the weak measure.

10.3 Coarse-grid correction

The coarse-grid correction phase of multigrid enters as a way to exploit the smoothing property of relaxation. The basic idea is that the fine-grid residual equation, which defines the error, is approximated on coarse levels where the error becomes oscillatory and is therefore easily handled by relaxation. The error components thus computed on these coarse levels are then interpolated back to the fine grid to correct the approximation there.

To see this more clearly, we start by assuming that a coarse grid and an interpolation operator have already been constructed. For example, if the fine level, which we call grid h, is a uniform grid on the two-dimensional unit square, then "grid $2h$" might be constructed, in what is considered to be a standard way, by choosing every other grid line in both coordinate directions. (Coarsening can divert substantially from this conventional multigrid construction, even to the point that it does not involve any geometry at all, as is often the case when algebraic multigrid methods are used.) Suppose also that we have already determined an interpolation operator, denoted by $\boldsymbol{I}_{2h}^{h}$, from grid $2h$ to grid h. ($\boldsymbol{I}_{2h}^{h}$ can be constructed based on standard linear interpolation, for example, but the following discussion does not assume this.) Now consider the discrete energy functional given by

$$F(\boldsymbol{v}^h) \equiv \langle \boldsymbol{A}^h \boldsymbol{v}^h, \boldsymbol{v}^h \rangle - 2\langle \boldsymbol{v}^h, \boldsymbol{f}^h \rangle.$$

Minimizing $F(\boldsymbol{v}^h)$ is equivalent to solving (10.1), and a little algebra shows that the best coarse-grid correction to a fixed approximation, $\boldsymbol{v}^h$, in the sense of minimizing $F(\boldsymbol{v}^h - \boldsymbol{I}_{2h}^{h}\boldsymbol{v}^{2h})$, is expressed by

$$\boldsymbol{v}^h \leftarrow \boldsymbol{v}^h - \boldsymbol{I}_{2h}^{h}\left(\left(\boldsymbol{I}_{2h}^{h}\right)^T \boldsymbol{A}^h \boldsymbol{I}_{2h}^{h}\right)^{-1}\left(\boldsymbol{I}_{2h}^{h}\right)^T\left(\boldsymbol{A}^h\boldsymbol{v}^h - \boldsymbol{f}^h\right).$$

This is the form of the coarse-grid correction that we use here and it gives rise to the following so-called *variational conditions* :

$$\boldsymbol{I}_{h}^{2h} \equiv \left(\boldsymbol{I}_{2h}^{h}\right)^T \text{ and } \boldsymbol{A}^{2h} \equiv \boldsymbol{I}_{h}^{2h}\boldsymbol{A}^h\boldsymbol{I}_{2h}^{h},$$

where $\boldsymbol{I}_{2h}^{h}$ represents interpolation from a coarse grid to grid h, $\boldsymbol{I}_{h}^{2h}$ is the restriction (or transfer) operator from the fine to the coarse grid, and $\boldsymbol{A}^{2h}$ is the coarse-grid matrix. (The first definition is called the *Galerkin condition* and it generally is assumed to hold only up to a multiplicative constant.) The basic issue in designing a multigrid method is choosing $\boldsymbol{I}_{2h}^{h}$, which amounts to determining a coarse set of grid points and a method for interpolating functions defined on those points to functions defined on the fine-grid points.

However, this energy-minimization formulation is convenient in that, once this is done, the coarse-grid matrix and the restriction operator are then determined automatically from the variational conditions.

With these constructions in hand, a two-grid version of multigrid, starting with initial guess v^h, is as follows :

- Apply coarse-grid correction : $v^h \leftarrow v^h - I_{2h}^h (A^{2h})^{-1} I_h^{2h} \left(A^h v^h - f^h \right)$.
- Relax once on v^h : $v^h \leftarrow v^h - \frac{1}{\|A^h\|} \left(A^h v^h - f^h \right)$.

This version begins with coarse-grid correction because it simplifies the theory, but multigrid can be constructed with these two phases reversed. The reverse form may be more intuitive because one can then think of relaxation as smoothing the error so that coarse-grid correction can then work effectively, but both forms actually work equally well in that they have the same convergence behavior. We have also included just one relaxation sweep for simplicity, although two or sometimes more are more common in practice. Note that the error propagation matrix for this scheme is given by

$$MG^h \equiv \left(I - \frac{1}{\|A^h\|} A^h \right) T^h,$$

where

$$\begin{aligned} T^h &\equiv I - S^h \\ S^h &\equiv I_{2h}^h (A^{2h})^{-1} I_h^{2h} A^h. \end{aligned} \tag{10.4}$$

Two-grid versions are generally not enough to achieve the optimality afforded by multigrid equipped with a full set of coarse levels. The key to the extension to a multilevel version is to notice that coarse-grid correction involves the solution of

$$A^{2h} w^{2h} = I_h^{2h} \left(A^h v^h - f^h \right)$$

(and then correction to the fine grid via $v^h \leftarrow v^h - I_{2h}^h w^{2h}$). We simply replace this step by a two-grid scheme that first improves the grid $2h$ approximation (namely, $w^{2h} = 0^{2h}$) by a correction from grid $4h$ and one relaxation sweep on grid $2h$. Continuing recursively, the algorithm actually begins on a very coarse grid, grid $H \equiv 2^m h$ say, solves the problem there, and then interpolates the result to grid $2^{m-1}h$ to form an initial guess for one relaxation there, the result of which is in turn interpolated to the next finer grid, and so on until the finest grid is reached and one relaxation sweep is performed. This process is called a $V(0,1)$-cycle because it starts from the fine grid and drops down through the coarser grids to the coarsest without doing any relaxation (hence the 0), then proceeds back up to the finest grid, performing

one relaxation (hence the 1) on each grid along the way. We can represent this $V(0,1)$-cycle by the following expression :

$$\boldsymbol{v}^h \leftarrow \boldsymbol{MV}^h\left(\boldsymbol{v}^h; \boldsymbol{f}^h\right), \tag{10.5}$$

where the $\boldsymbol{v}^h$ on the right is a given grid h approximation and $\boldsymbol{f}^h$ is a given source term. $\boldsymbol{MV}^h\left(\boldsymbol{v}^h; \boldsymbol{f}^h\right)$ is defined recursively as shown in Algorithm 7.

Algorithm 7: *Multigrid V(0,1)-cycle :* $\boldsymbol{MV}^h\left(\boldsymbol{v}^h; \boldsymbol{f}^h\right)$

Input: $h, \boldsymbol{v}^h, \boldsymbol{f}^h$
if $h = H$ **then**
  Solve $\boldsymbol{A}^h\boldsymbol{u}^h = \boldsymbol{f}^h$
  $\boldsymbol{v}^h \leftarrow \boldsymbol{u}^h$
else
  $\boldsymbol{f}^{2h} \leftarrow \boldsymbol{I}_h^{2h}\left(\boldsymbol{A}^h\boldsymbol{v}^h - \boldsymbol{f}^h\right)$ ▷ Restrict residual to coarser grid
  $\boldsymbol{v}^{2h} \leftarrow \boldsymbol{0}$ ▷ Initialize coarse grid solution
  $\boldsymbol{v}^{2h} \leftarrow \boldsymbol{MV}^{2h}\left(\boldsymbol{v}^{2h}; \boldsymbol{f}^{2h}\right)$ ▷ Solve coarser problem
  $\boldsymbol{v}^h \leftarrow \boldsymbol{v}^h - \boldsymbol{I}_{2h}^h\boldsymbol{v}^{2h}$ ▷ Interpolate correction from coarser grid
  $\boldsymbol{v}^h \leftarrow \boldsymbol{v}^h - \frac{1}{\|\boldsymbol{A}^h\|}\left(\boldsymbol{A}^h\boldsymbol{v}^h - \boldsymbol{f}^h\right)$ ▷ Apply relaxation
end ifreturn $\boldsymbol{v}^h$

Below is a list some basic properties of our constructs. These properties result from $\boldsymbol{S}^h$ and $\boldsymbol{T}^h$ being "energy"-orthogonal projections onto the *range* of $\boldsymbol{I}_{2h}^h$ and the *null space* or *kernel* of $\boldsymbol{I}_h^{2h}\boldsymbol{A}^h$, respectively. Keep in mind that the range of $\boldsymbol{I}_{2h}^h$ consists of vectors that can be exactly represented by interpolation from the coarse grid, while the kernel of $\boldsymbol{I}_h^{2h}\boldsymbol{A}^h$ consists of the vectors that cannot be represented at all by the coarse grid. That is, $\boldsymbol{S}^h$ and $\boldsymbol{T}^h$ provide a decomposition into errors that *can* be eliminated by the coarse grid and vectors that *cannot* (and must therefore be dealt with on the fine grid). More specifically we have the following :

- Any grid h vector $\boldsymbol{e}^h$ can be expressed as $\boldsymbol{e}^h = \boldsymbol{s}^h + \boldsymbol{t}^h$, where $\boldsymbol{s}^h \in R(\boldsymbol{I}_{2h}^h)$, $\boldsymbol{t}^h \in N(\boldsymbol{I}_h^{2h}\boldsymbol{A}^h)$, and $\langle \boldsymbol{s}^h, \boldsymbol{t}^h \rangle_{\boldsymbol{A}^h} = 0$ (that is, $\boldsymbol{s}^h$ and $\boldsymbol{t}^h$ are orthogonal in the energy inner product).
- $\boldsymbol{T}^h\boldsymbol{s}^h = \boldsymbol{0}$, $\boldsymbol{S}^h\boldsymbol{t}^h = \boldsymbol{0}$, $\boldsymbol{T}^h\boldsymbol{t}^h = \boldsymbol{t}^h$, $\boldsymbol{S}^h\boldsymbol{s}^h = \boldsymbol{s}^h$, $\boldsymbol{T}^h\boldsymbol{e}^h = \boldsymbol{t}^h$, $\boldsymbol{S}^h\boldsymbol{e}^h = \boldsymbol{s}^h$, $\langle \boldsymbol{T}^h\boldsymbol{w}^h, \boldsymbol{S}^h\boldsymbol{z}^h \rangle_{\boldsymbol{A}^h} = 0 \; \forall \boldsymbol{w}^h, \boldsymbol{z}^h$, and $\|\boldsymbol{S}^h\|_{\boldsymbol{A}^h} = \|\boldsymbol{T}^h\|_{\boldsymbol{A}^h} = 1$.
- $\|\boldsymbol{T}^h\boldsymbol{e}^h\|_{\boldsymbol{A}^h} = \min_{\boldsymbol{u}^{2h}} \|\boldsymbol{e}^h - I_{2h}^h\boldsymbol{u}^{2h}\|_{\boldsymbol{A}^h}$ (that is, coarse-grid correction minimizes the energy norm of the error over all possible corrections in the range of interpolation).

We began this discussion by providing motivation for using relaxation on coarse levels to improve the solver on the fine grid. We then showed how to

construct a multigrid solver based on this concept. Next, we discuss theoretical properties that ensure that this construction leads to an efficient solver.

10.4 Convergence analysis

We first analyze the two-grid scheme, which has the error propagation matrix $\boldsymbol{MG}^h = \boldsymbol{G}^h\boldsymbol{T}^h$. Each cycle converges uniformly in energy for any initial guess if and only if

$$\|\boldsymbol{G}^h\boldsymbol{T}^h\| = \left\| \left(I - \frac{1}{\|\boldsymbol{A}^h\|}\boldsymbol{A}^h \right) \boldsymbol{T}^h \right\|_{\boldsymbol{A}^h}^2 \leq \gamma$$

for some fixed $\gamma \in [0, 1[$. A little algebra shows that this is the same as the condition that the strong measure is not close to zero :

$$\mathcal{M}_s(\boldsymbol{t}^h) \geq \delta$$

for some fixed $\delta > 0$ (more precisely, $\delta \in [\frac{1-\gamma}{2}, 1-\gamma[$) and for all $\boldsymbol{t}^h$ in the range of $\boldsymbol{T}^h$. To understand this two-grid condition a little better, we make a (generally unrealistic) assumption that $\boldsymbol{t}^h$ is an eigenvector of $\boldsymbol{A}^h$ with corresponding eigenvalue λ. Then

$$\mathcal{M}_s(\boldsymbol{t}^h) = \frac{\langle \boldsymbol{A}^h\boldsymbol{t}^h, \boldsymbol{A}^h\boldsymbol{t}^h \rangle}{||\boldsymbol{A}^h||\langle \boldsymbol{A}^h\boldsymbol{t}^h, \boldsymbol{t}^h \rangle} = \frac{\lambda^2}{\lambda\|\boldsymbol{A}^h\|} = \frac{\lambda}{\|\boldsymbol{A}^h\|},$$

so the two-grid condition requires that λ be theoretically comparable (up to δ) to $\|\boldsymbol{A}^h\|$, the largest eigenvalue of $\boldsymbol{A}^h$. Of course, the practical quality of convergence depends on δ, and a value of δ near zero means that γ is near 1, so it signals poor expected numerical performance. Generally, we want error components that $\boldsymbol{T}^h$ cannot eliminate to lie primarily in the upper spectrum of $\boldsymbol{A}^h$.

Unfortunately, counterexamples (such as standard cell-centered coarsening of discrete Poisson equations; c.f., Bramble et al. [1996]) show that this two-grid condition does not suffice to establish uniform convergence of the multilevel version, $\boldsymbol{MV}^h$. However, a condition that is sufficient, (c.f. [Vassilevski, 2008]), is the following property :

Definition 1. *Strong Approximation Property (SAP)* A coarse-grid correction scheme satisfies the Strong Approximation Property if and only if

$$\min_{\boldsymbol{u}^{2h}} \|\boldsymbol{e}^h - \boldsymbol{I}_{2h}^h\boldsymbol{u}^{2h}\|_{\boldsymbol{A}^h}^2 \leq \frac{C}{\|\boldsymbol{A}^h\|}\langle \boldsymbol{A}^h\boldsymbol{e}^h, \boldsymbol{A}^h\boldsymbol{e}^h \rangle$$

for some fixed constant $C < \infty$ and for all grid h vectors $\boldsymbol{e}^h$.

Another way to write this condition is in terms of the multigrid correction matrix and the strong measure :

$$\frac{\|\boldsymbol{T}^h \boldsymbol{e}^h\|_{\boldsymbol{A}^h}^2}{\|\boldsymbol{e}^h\|_{\boldsymbol{A}^h}^2} \leq C\mathcal{M}_s(\boldsymbol{e}^h).$$

Note that this property reduces to the two-grid condition (with $C = \frac{1}{\delta}$) when $\boldsymbol{e}^h$ is in the range of $\boldsymbol{T}^h$, that is, $\boldsymbol{e}^h = \boldsymbol{t}^h$. (Remember that vectors in the range of $\boldsymbol{T}^h$ are precisely those that the multigrid correction leaves untouched and therefore undiminished.)

Theorem 1. *(V-Cycle Convergence)* The strong approximation property is a sufficient condition for uniform convergence of the $V(0,1)$-cycle in the energy norm :

$$\|\boldsymbol{M}\boldsymbol{V}^h\|_{\boldsymbol{A}^h}^2 \leq 1 - \frac{1}{C}.$$

Proof. A little algebra shows that the SAP is equivalent to the following smoothing property :

$$\begin{aligned} \left\| \left(I - \frac{1}{\|\boldsymbol{A}^h\|} \boldsymbol{A}^h \right) \boldsymbol{e}^h \right\|_{\boldsymbol{A}^h}^2 &\leq \|\boldsymbol{e}^h\|_{\boldsymbol{A}^h}^2 - \frac{1}{\|\boldsymbol{A}^h\|} \|\boldsymbol{A}^h \boldsymbol{e}^h\|^2 \\ &\leq \|\boldsymbol{s}^h\|_{\boldsymbol{A}^h}^2 + \left(1 - \frac{1}{C}\right) \|\boldsymbol{t}^h\|_{\boldsymbol{A}^h}^2. \end{aligned} \tag{10.6}$$

Assume now for induction purposes that $\|\boldsymbol{M}\boldsymbol{V}^{2h}\|_{\boldsymbol{A}^{2h}}^2 \leq 1 - \frac{1}{C}$. Remembering that $\boldsymbol{e}^h = \boldsymbol{T}^h \boldsymbol{e}^h + \boldsymbol{S}^h \boldsymbol{e}^h$ and that $\boldsymbol{S}^h \boldsymbol{e}^h$ is in the range of $\boldsymbol{I}_{2h}^h$, then we can write $\boldsymbol{e}^h = \boldsymbol{T}^h \boldsymbol{e}^h + \boldsymbol{I}_{2h}^h \boldsymbol{e}^{2h}$ for some grid $2h$ vector $\boldsymbol{e}^{2h}$. Also, with some effort, it can be shown that

$$\boldsymbol{M}\boldsymbol{V}^h = \left(I - \frac{1}{\|\boldsymbol{A}^h\|} \boldsymbol{A}^h \right) \left(\left(\boldsymbol{I}_{2h}^h \boldsymbol{M}\boldsymbol{V}^{2h} (\boldsymbol{A}^{2h})^{-1} \boldsymbol{I}_h^{2h} \boldsymbol{A}^h \right) + \boldsymbol{T}^h \right), \tag{10.7}$$

so (10.6) and (10.7), together with the smoothing property and recursion assumptions, combine to show that

$$\begin{aligned} \|\boldsymbol{M}\boldsymbol{V}^h \boldsymbol{e}^h\|_{\boldsymbol{A}^h}^2 &\leq \left(1 - \frac{1}{C}\right) \|\boldsymbol{T}^h \boldsymbol{e}^h\|_{\boldsymbol{A}^h}^2 + \|\boldsymbol{I}_{2h}^h \boldsymbol{M}\boldsymbol{V}^{2h} (\boldsymbol{A}^{2h})^{-1} \boldsymbol{I}_h^{2h} \boldsymbol{A}^h \boldsymbol{I}_{2h}^h \boldsymbol{e}^{2h}\|_{\boldsymbol{A}^h}^2 \\ &= \left(1 - \frac{1}{C}\right) \|\boldsymbol{T}^h \boldsymbol{e}^h\|_{\boldsymbol{A}^h}^2 + \|\boldsymbol{M}\boldsymbol{V}^{2h} \boldsymbol{e}^{2h}\|_{\boldsymbol{A}^{2h}}^2 \\ &\leq \left(1 - \frac{1}{C}\right) \left(\|\boldsymbol{T}^h \boldsymbol{e}^h\|_{\boldsymbol{A}^h}^2 + \|\boldsymbol{e}^{2h}\|_{\boldsymbol{A}^{2h}}^2 \right) \\ &= \left(1 - \frac{1}{C}\right) \|\boldsymbol{e}^h\|_{\boldsymbol{A}^h}^2. \end{aligned}$$

This proves our assertion that $\|\boldsymbol{M}\boldsymbol{V}^h\|_{\boldsymbol{A}^h}^2 \leq 1 - \frac{1}{C}$. □

The strong approximation property requires that vectors in the kernel of A^h be exactly eliminated by coarse-grid correction. This follows from noting that if e^h is in the kernel of A^h, then the right-hand side of the SAP vanishes, and thus so too must the left-hand side. However, we are assuming that A^h is SPD, so its kernel consists only of $\mathbf{0}$ and this requirement is, therefore, trivially satisfied by setting $u^{2h} = \mathbf{0}$. On the other hand, the strong approximation property also clearly shows that any vector that is in the near kernel (in the sense of yielding a relatively small residual, $A^h e^h$) must be nearly eliminated by coarse-grid correction. This again follows from noting that if e^h is in the near kernel of A^h, then the right-hand side of the SAP is small, and thus so too must the left-hand side be. It is this observation that provides our motivation for constructing an interpolation operator that adequately approximates near-kernel components. This focus on the near kernel is generally sufficient because algebraically oscillatory vectors automatically satisfy the SAP: by definition of algebraically oscillatory error e^h, $\mathcal{M}_s(e^h) \geq \delta$ for some $\delta \gg 0$, so it follows that

$$\frac{\|T^h e^h\|_{A^h}^2}{\|e^h\|_{A^h}^2} \leq \frac{\|e^h\|_{A^h}^2}{\|e^h\|_{A^h}^2} \leq C\mathcal{M}_s(e^h),$$

where $C = \frac{1}{\delta} \ll \infty$. Finally, since the SAP involves the choice of coarse-grid points and interpolation to the fine grid, it gives us a way to assess the quality of the coarse-grid correction process.

As we have just seen, the strong measure is related to the strong approximation property in the energy inner product. We have a similar relationship for the weak case in the Euclidean norm.

Definition 2. *Weak Approximation Property (WAP)* A coarse grid correction scheme satisfies the Weak Approximation Property if and only if

$$\min_{u^{2h}} \|e^h - I_{2h}^h u^{2h}\|^2 \leq \frac{C}{\|A^h\|} \langle A^h e^h, e^h \rangle.$$

At this point, it should be no surprise that the WAP can be written in terms of the weak measure :

$$\frac{\|T^h e^h\|^2}{\|e^h\|^2} \leq C\mathcal{M}_w(e^h).$$

By arguments similar to those for the strong approximation property, the weak approximation property confirms uniform convergence of the two-grid form of multigrid in the Euclidean norm. It can also confirm convergence of other multigrid cycling schemes, like the so-called *W-cycle* and *μ-cycle*, that are more aggressive than the V-cycle, provided the effort spent on the coarse grids is commensurate with the size of C.

10.5 Designing MG algorithms

The preceding sections assumed that a coarse grid and an interpolation operator were given. In this section, we seek to use the theoretical results that were based on this assumption to go back and determine how to actually design effective coarse grids and interpolation operators.

We first need to point out that, while V-cycle convergence is usually the desired target in applications because of its superior efficiency, the global nature of the energy norm that the SAP is based on makes it more difficult to use as a design tool. The problem stems from the need to determine interpolation in a local *neighborhood* (i. e., a small group of points interconnected by the entries of A^h). Using the energy norm to measure how well interpolation approximates local errors is problematic because A^h, which is present in that norm, generally reaches to points outside of the neighborhood. This lack of locality inhibits the ability to isolate the design of interpolation. For this reason, it is more common in practice to develop interpolation schemes based on the WAP because estimates involving the Euclidean norm can be wholly restricted to individual neighborhoods. This locality provides the basis for the classical development of Smoothed Aggregation (SA).

The practical importance of determining the coarsening process locally cannot be overemphasized. Just as finite elements realized its true potential when it transitioned from a global spectral approach to a local piecewise polynomial methodology, so too is multigrid most effective when coarse-grid corrections are constructed from local approximations of errors based on their local character. The dimension of the space of smooth errors that relaxation cannot effectively reduce is typically a significant fraction of the dimension of the fine-level space. This is generally much too large to permit computation of a global basis for that space. This reasoning is analogous to saying that global spectral discretizations based on eigenvectors or Fourier modes are generally impractical. So the approach taken by algebraic multigrid methods is to attempt to approximate algebraically smooth errors over small sets of points that are interconnected in the matrix by using a significantly smaller number of degrees of freedom. The intent is that these approximations can then be pieced together to provide adequate global approximation to all algebraically smooth errors.

In the following we are particularly interested in smoothed aggregation (SA). At this point we will therefore focus on how the ideas discussed above for coarsening and interpolation ends up motivating the construction of the SA algorithm.

As noted before SA begins by partitioning the fine grid into groups or aggregates of nodes. This is accomplished by aggregating several points that are strongly interconnected in $\boldsymbol{A}^h = \left(a_{ij}^h\right)$. The basic idea is that $|a_{ij}^h|$ is used as a measure to determine how dependent the values at points i and j of an algebraically smooth error, $\boldsymbol{e}^h = \left(e_i^h\right)$, are on each other. The motivation here is that, for scalar elliptic equations, a large value of $|a_{ij}^h|$ relative to $\sqrt{a_{ii}^h a_{jj}^h}$ implies that e_i^h and e_j^h are approximately equal. Once a full set of aggregates have been determined, SA then uses basis functions defined in each aggregate to form linear combinations that approximate fine-grid smooth error. The coefficients in these linear combinations constitute the coarse-grid unknowns. Often, local analysis can be used to guarantee that the WAP holds for a given proper choice of these basis functions, which would then guarantee that the method as constructed would provide at least a good two-grid solver and possibly an effective W-cycle solver (where the coarse grid equations are solved more aggressively than for V-cycle solvers). But this *unsmoothed* version of SA is not likely to exhibit optimal V-cycle performance. To achieve that, smoothing of the interpolation operator may be necessary. The fundamental idea is to improve interpolation and the coarsening it is based on by applying a smoother to $\boldsymbol{I}_{2h}^h$, whence the moniker *smoothed* aggregation. We'll get back to this shortly.

To better understand how SA constructs interpolation based on the WAP, assume that the fine grid has been partitioned into a set of agglomerates, $\mathcal{A} = \{a\}$. For each agglomerate, the aim now is to satisfy the *local* WAP: there exists a constant $C \ll \infty$ such that, for all fine-grid error $\boldsymbol{e}^h$ and every agglomerate $a \in \mathcal{A}$, a coarse-grid representative, $\boldsymbol{u}^{2h}$, exists that satisfies

$$\|\boldsymbol{e}^h - I_{2h}^h \boldsymbol{u}^{2h}\|_a^2 \leq \frac{C|a|}{\|\boldsymbol{A}^h\|} \langle \boldsymbol{A}^h \boldsymbol{e}^h, \boldsymbol{e}^h \rangle,$$

where $\|\cdot\|_a$ denotes the local Euclidean inner norm and $|a|$ is the relative size of a (so that $\sum_{a \in \mathcal{A}} |a| = 1$). Note that the sum of this local WAP over all $a \in \mathcal{A}$ yields the global one.

Now, it would be impossible to test the WAP to see if it holds for all $\boldsymbol{e}^h$, so the idea is instead to choose a set of vectors that hopefully represents all near-kernel components locally. (As we noted above, focusing on such components is appropriate because those that are algebraically oscillatory automatically satisfy the WAP, just as they do the SAP.) The local nature of near-kernel components are known for many problems. As examples, for scalar elliptic equations, they are approximately constant locally and, for linear elasticity, they look locally like *rigid body modes* (constant displacements and rigid rotations). In such cases, we can take these few global vectors,

normalize them in energy, and then restrict them to each agglomerate to determine an effective basis there. This can be done by finding a minimal local basis that adequately (according to the WAP) approximates all linear combinations of these restricted near-kernel components. The actual computation amounts to forming a local matrix, $\boldsymbol{T}$, whose columns consist of these restricted near-kernel components, solving the eigenproblem for $\boldsymbol{T}^T\boldsymbol{T}$, and then letting the basis be the resulting eigenvectors whose eigenvalues are less than $\frac{C|a|}{\|A\|}$. (Remember that the near-kernel components were normalized in energy.) It is important to realize that SA computes a minimal basis, which not only controls complexity, but also ensures that redundancy does not creep into the coarse-grid problem. Any contamination by an exact or near linear dependence of a local basis would lead to artificial singularity or ill-conditioning of the coarse-grid matrix, making the development of the solver on coarser levels very problematic.

As we noted above, to improve interpolation approximation properties, SA applies a smoother to the interpolation operator, [Vaněk, 1992]. To understand the role of the smoothing process in SA, consider the one-dimensional example of a uniform grid of n points on the unit interval. If we consider three neighboring interior points and the vector function that is 1 at these points and 0 elsewhere, such a vector might make up a typical basis element for the coarse level that has about $\frac{n}{3}$ points. That is, we can define the coarse-level space as the set of coefficients of these *piecewise-constant* basis elements. Piecewise-constant functions by themselves provide poor *discretization* spaces for most elliptic equations, and so too do piecewise-constant vectors provide for poor *coarsening*. But a simple averaging of the basis elements given by the stencil $(\frac{1}{4}\ \frac{1}{2}\ \frac{1}{4})$ yields *piecewise-linear* basis elements, which provide higher-order approximation. Averaging of the basis elements is typically what a properly constructed smoother might do. This reasoning provides the motive for smoothing $\boldsymbol{I}_{2h}^h$ in an attempt to improve its approximation order. In other words, for the model 1D problem, piecewise constants satisfy the WAP but not the SAP; yet, smoothing yields piecewise linear functions that do satisfy the SAP; and so the motivation is that smoothing the interpolation operator might provide the approximation properties we need to confirm good V-cycle convergence for other problems.

10.6 Nodal vs. unknown-based coarsening

Even with the above requirements there are many ways to construct the interpolation operator, $\boldsymbol{P}$. Standard GMG and AMG are examples of so-called *unknown-based* multigrid methods, where the degrees-of-freedom (DOF) are

coarsened and interpolated separately. To illustrate this approach, note that a cloth simulation typically involves three *displacement* unknowns. The resulting matrix, $\boldsymbol{A}$, can therefore be written in the form

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{A}_{1,1} & \boldsymbol{A}_{1,2} & \boldsymbol{A}_{1,3} \\ \boldsymbol{A}_{2,1} & \boldsymbol{A}_{2,2} & \boldsymbol{A}_{2,3} \\ \boldsymbol{A}_{3,1} & \boldsymbol{A}_{3,2} & \boldsymbol{A}_{3,3} \end{bmatrix}, \tag{10.8}$$

where each block is of size $n \times n$, with n being the number of nodes used in the discretization. When using an unknown-based method, interpolation and coarsening are constructed based on the block diagonals to form the full interpolation operator:

$$\boldsymbol{P} = \begin{bmatrix} \boldsymbol{P}_{1,1} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{P}_{2,2} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{P}_{3,3} \end{bmatrix}. \tag{10.9}$$

The coarse-grid matrix is then formed using the Galerkin operator $\boldsymbol{P}^T\boldsymbol{A}\boldsymbol{P}$. This works well when the PDE is dominated by the connections within the unknowns. However, for problems like elasticity with strong off-diagonal connections, unknown-based approaches can suffer poor convergence and scaling.

Another choice for applying multigrid to systems of PDEs is the *nodal* approach, where the fine-grid matrix is structured in blocks of the same size as the number of unknowns in the PDE at each grid point. This approach complicates the determination of strength between nodes and, in turn, coarsening, but it provides a bridge to smoothed aggregation. Instead of coarsening a grid by identifying a set of nodes that becomes the coarse grid, SA partitions the grid into aggregates that are strongly interconnected. Akin to how finite elements use local functions, SA then assigns each aggregate a basis of local vectors that can be linearly combined with each other and bases of the other aggregates to adequately (according to the WAP) approximate all fine-grid smooth error. The coefficients in these linear combinations constitute the coarse-grid unknowns, and the vectors themselves represent an approximation of the near-kernel for the coarse problem. This form gives SA the natural ability to interpolate across unknowns, and it has the added benefit of being able to fit a much richer set of errors.

10.7 Designing problems for multigrid

As already noted earlier, there are similarities between multigrid methods and finite element methods. These similarities extend further in the sense

that, just as the appropriate choice of finite elements can improve convergence of the discretization, so too does a good discretization help convergence of the multigrid solver.

To be a little more specific, note that discretization can be viewed as an attempt to approximate an infinite-dimensional continuum by a finite-dimensional grid h space. This would only make sense if the target function could be pinned down somehow, which most discretizations do by relying on some sense of smoothness. For example, continuous piecewise-(bi)linear finite element spaces target solutions of elliptic partial differential equations by relying on boundedness of their second derivatives. This is a sensible goal when the corresponding operator exhibits so-called "full regularity" because it means that the second derivatives of the solution are bounded by the source term. The success of discretization thus depends heavily on the behavior of the differential operator, which in the standard elliptic case amounts to full regularity. Operators that exhibit reduced regularity, such as those with difficulties introduced by discontinuous coefficients or re-entrant corners, can certainly be treated effectively by discretization schemes, but they usually require special handling that carefully addresses these difficulties. Otherwise convergence suffers.

So it is with multigrid. Effective coarse-grid correction depends heavily on the behavior of the matrix : if it comes from standard discretization of a fully regular elliptic differential operator, then standard coarse-grid correction schemes that mimic the discretization itself should be effective. In fact, multigrid aims to approximate the grid h space by the grid $2h$ space. It does this by pinning down the error using relaxation to obtain small relative residuals. For the fully regular case, small residuals mean that these errors vary slowly in any given neighborhood, that is, that the error is locally almost constant. It is this "discrete regularity" property that enables use of standard coarse-grid correction schemes to be used to full effectiveness.

An important point here is that, while we expect good approximation by standard discretization of fully regular differential operators, we can also expect good overall performance of standard multigrid solvers applied to them. Conversely, multigrid performance can degrade when applied to discretizations that are less accurate or operators with reduced regularity. Special treatment in the multigrid approach may be necessary to handle these difficulties. In many cases, it is often more effective to improve the discretization or operator formulation when this is possible.

The discussion here has focussed primarily on elliptic PDEs, specifically linear second-order elliptic equations discretized by continuous piecewise linear or bilinear finite elements (or other traditional discretization meth-

ods with similar approximation properties). It is well known that multigrid *methods* have been very effective for solving problems in this model elliptic class. When regular grids are used, a geometric multigrid method using standard components (linear interpolation, full weighting, and simple relaxation schemes like Richardson iteration) is usually able to solve such problems at a cost that is optimal in the sense that it is proportional to the number of unknowns. When the problem begins to stray from the classical elliptic regime (as in the presence of strong convection, strong anisotropies, or jump-discontinuous coefficients) or when irregular grids are used, standard *algebraic* multigrid methods can usually be applied at optimal cost.

What is less well known is that the multigrid *methodology* has much broader applicability than this model elliptic class. For example, multigrid has been applied successfully to nonlinear problems since its practical origin in Brandt's seminal paper, [Brandt, 1977]. In fact, the so-called *full* multigrid approach is especially effective for nonlinear problems in that it often needs the equivalent of only one or two multigrid cycles on the fine-grid to solve the nonlinear problem. Moreover, the methodology has been extended well beyond the elliptic case and has been highly successful for solving Navier-Stokes equations and other systems in fluid flow, as well as many problems in structural mechanics (c.f., Brandt and Livne [2011]). It has also been successful in solving previously intractable problems that are beyond the realm of differential equations (e.g., geodetic equations [Brandt et al., 1982], quantum chromodynamics [Brannick et al., 2007], and Markov chains [Sterck et al., 2008]). However, these applications require special multigrid treatment that can lead to algorithms that are much more sophisticated than the standard multigrid methods in more common use.

10.8 Summary

In summary, multigrid methodology involves two key components : First, a coarse grid that can approximate the algebraically smooth errors that are left behind by relaxation on the fine grid. Second, an interpolation operator from the coarse grid to the fine grid that adequately represents these errors in accordance with the WAP.

SA, in particular, approximates algebraically smooth errors by choosing aggregates of nodes that are connected strongly enough to enable one or a few basis elements to represent these errors locally, with the WAP guiding the choice of the basis elements that properly approximate these smooth errors.

More generally, to achieve optimal multigrid performance, the following properties are desirable :

- The relaxation process should effectively reduce oscillatory components of the error. More precisely, the errors that relaxation cannot quickly eliminate should provide a pattern that can be effectively exploited in the local construction of interpolation.
- There should be some sense of locality, such as that afforded by the WAP, since that is what makes it possible and practical to determine an adequate representation of the near-kernel components.
- The coarse space should be constructed in neighborhoods of strongly interconnected points and they should carefully avoid (near) redundancy.
- The range of the interpolation operator (from the coarse grid to the fine grid) should represent near-kernel components of the fine-grid operator accurately according to the SAP.
- If the matrix arises from a PDE, the discretization that creates it should be accurate enough to satisfy the continuum version of the SAP.

CHAPTER 11

Smoothed Aggregation

Given the theory presented in the previous chapter, the goal here is to develop an efficient multigrid method for cloth simulation based on smoothed aggregation (SA). We start by carefully considering the challenges posed by the thin-shell equations, and then proceed to present the smoothed aggregation method in detail. Ultimately we use this method as a preconditioner for a conjugate gradient solver.

The two key challenges are coupling between the variables and time varying anisotropy. SA is designed to handle both of these challenges and ends up being superior to simple diagonally preconditioned conjugate gradients even for relatively small problems. SA is also a purely algebraic method, so it is agnostic to the choice of input meshes and relieves the user of explicitly generating hierarchies or choosing meshes such that they are amenable to coarsening. In fact, SA works with completely irregular and potentially even (time) adaptive meshes such as the one in [Narain et al., 2012]. Moreover, it works equally well with triangle and quad meshes.

While we do not solve the full contact model presented in previous chapters, we do incorporate the resulting solver into a production quality cloth solver based on [Baraff and Witkin, 1998]. To that end, we introduce a novel way of handling the filtering step originally introduced by Baraff and Witkin [1998]. The new method, called *Prefiltered Preconditioned CG (PPCG)*, is described in Section 11.5. For highly efficient preconditioners, like the ones proposed here, prefiltering is essential, but, even for simple preconditioners, prefiltering provides significant benefits in the presence of many constraints.

Although this chapter focuses on the cloth model introduced by Baraff and Witkin [1998], we expect most of the results to carry over directly to the cloth models discussed earlier in this thesis or at least provide a clear pathway for how to treat them. In fact, in most cases, the proposed method can be incorporated into other simulators simply by replacing the innermost linear solver.

Figure 11.1: *The smoothed aggregation based method combined with prefiltering as presented in this chapter provides an* $8\times$ *speedup for a walk cycle animation of this character and* $6\times$ *for a run cycle animation. These numbers are compared to a block diagonally preconditioned CG method. The garments consist of a combined* $371,064$ *vertices.* © *Disney*

At the end of this chapter, we present numerical tests of the new approach based on a range of examples. This confirms $6-8\times$ speedups on a fully dressed character with 371k vertices (see Fig. 11.1), and even larger speedups on synthetic examples.

11.1 Challenges for multigrid

As mentioned in Chapter 2, the partial differential equations (PDEs) for thin shell elasticity with large deformations are complicated. Still, most cloth models are approximations of these equations, which means that these are the equations that should guide the design of our multigrid methods. However, even the highly simplified models of a planar elastic membrane undergoing small deformations result in biharmonic equations. For most methods, including multigrid, these equations are substantially more complicated to find solutions for than Poisson-like problems, which is what is often considered in the multigrid literature. It is therefore to be expected that standard methods do not perform optimally.

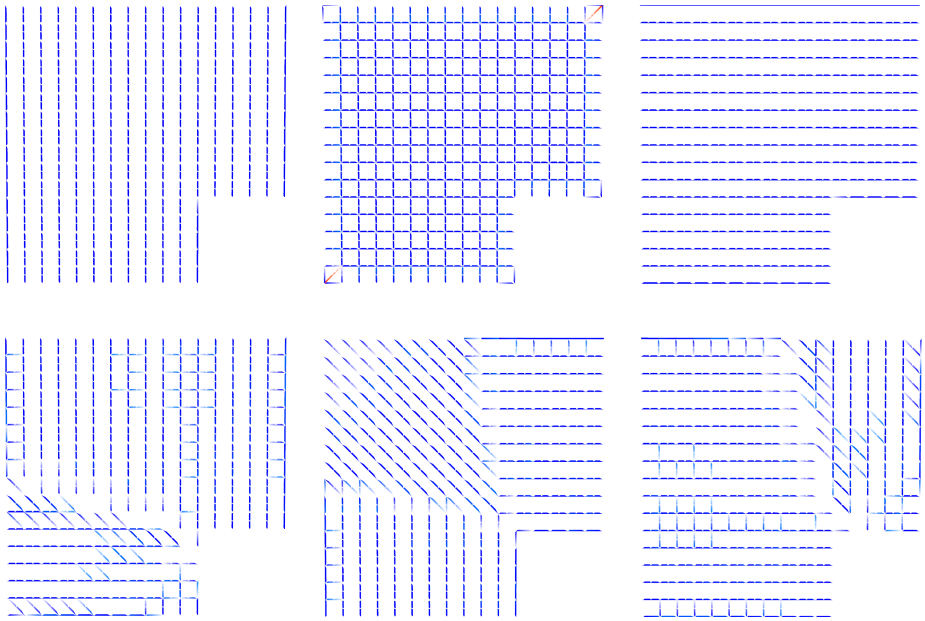
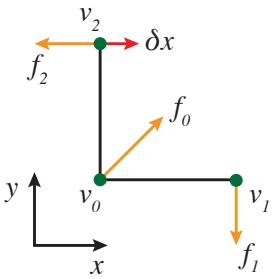


Figure 11.2: *Visualization of the strength of connection within a matrix for each of the three variables, (x, y, z), at two frames of a simulation. Strong negative off-diagonal connections between vertices are shown in UV space as blue lines. The two red lines in the middle of the top row image indicate strong positive off-diagonal connections. For clarity, "weak" connections are omitted. In the undeformed state (top row), the x and z components are each anisotropic but in different directions. In the deformed state (bottom row), different directions of anisotropy appear even within a single variable.*

Two challenges that affect our work directly are anisotropy and strong coupling between variables. Anisotropy is usually associated with constitutive models and, as discussed earlier chapters, cloth is an example of an anisotropic material. However, in the context of multigrid methods,

anisotropy simply means that certain variables are connected more strongly than others in the underlying matrix, and that there is a pattern to the directionality of these strong connections. The notion of a "strong connection" here corresponds to a large off-diagonal value of the associated matrix element relative to the diagonal. What is important to note is that this type of anisotropy occurs even with an isotropic homogeneous elastic material. This behavior is due in part to the Poisson effect. As an illustration, consider the simulation of an L-shaped piece of cloth, where the boundary along the cut-out corner is held fixed while the rest falls under gravity. The corresponding strength of connections in the associated stiffness matrix exhibit not only distinct anisotropies, but also directions of anisotropy that vary in space and time (see Fig. 11.2). Standard methods like semi-coarsening or line relaxation used with geometric multigrid are thus ineffective for this problem.



The second challenge related to strong coupling between variables can easily be seen by considering the simple 2D example shown in the inset figure to the right. In this example, two edges are bent from their rest configuration, which generates the set of bending forces labeled f_0, f_1, and f_2. If we apply a displacement, δx, to the top vertex, v_2, then all the bend forces increase in magnitude. In particular, this means that a change in the x coordinate of v_2 leads to a change in the x coordinate of f_2, but it also leads to a change of the y coordinate of f_1. Because the two changes have the same magnitude, the x and y variables are interpreted to be strongly coupled from a multigrid point of view. Neither *geometric multigrid methods (GMG)*, [Brandt, 1977], nor *algebraic multigrid methods (AMG)*, [Brandt et al., 1985], when designed in a standard way, work optimally when the variables are strongly coupled in this way.

The theory in Section 10.6 suggests that this strong cross-variable coupling would lead to poor performance of *standard* AMG. We confirmed this numerically by running several tests for the L-shaped problem mentioned above. Let the number of vertices in a simulation be n. Standard AMG then performed well for each of the three $n \times n$ blocks associated with the individual unknowns when the coupling between unknowns was deleted in the matrix. However, for the full matrix, its convergence rate was poor even for small problems, and degraded further as the problem sizes increased. Standard AMG simply was unable to provide a significant improvement over diagonally preconditioned conjugate gradients (PCG).

In Section 10.6, it was pointed out that SA deals with coupled variables by

using a nodal approach instead of an unknown-based approach. It also deals with the anisotropy because its coarsening is based directly on the strength of connection within the given matrix. As such, we expect SA to provide a good foundation for our cloth problems.

11.2 Smoothed Aggregation

At this point, consider the details required for implementing an SA solver. The construction in SA of a hierarchy of matrices and the corresponding interpolation operators between successive levels proceeds in three stages: selection of aggregates ($\mathcal{A}_i$ on level $i = 1, 2, \ldots, m$ from fine to coarse grids), forming interpolation operators ($\boldsymbol{P}_i$), and then forming coarse-grid operators ($\boldsymbol{A}_{i+1} = \boldsymbol{P}_i^T \boldsymbol{A}_i \boldsymbol{P}_i$). Since SA is a nodal approach, on any given level i of the hierarchy, $\boldsymbol{A}_i$ is assumed to have n_i nodes, each corresponding to $b_i \times b_i$ blocks. At the finest level, b_1 is the number of unknowns in the original PDE (i.e., 3 displacements in our case). The dimensions of $\boldsymbol{A}_i$ is block $n_i \times n_i$ when considered nodally, and $(n_i \cdot b_i) \times (n_i \cdot b_i)$ when all unknowns are considered.

A key ingredient in smoothed aggregation is an initial set of near-kernel vectors (see Section 10.5) which must be provided by the user. Let these vector constitute the columns of a matrix, $\boldsymbol{K}$. This near-kernel matrix is used below to construct bases for the agglomerates. $\boldsymbol{K}$ must have the property that any near-kernel component $\boldsymbol{e}$ must be adequately approximated (according to the WAP) in each aggregate by a linear combination of the columns of $\boldsymbol{K}$ restricted to that aggregate. For scalar Poisson equations, one near-kernel component (typically the constant vector) is usually enough to obtain good performance. For 2D linear elasticity, three components (typically a constant for each of the two displacement unknowns and a rotation) are usually needed. In Section 11.3, we return to the problem of choosing $\boldsymbol{K}$.

The first step in aggregating nodes is to form a strength-of-connection (SOC) matrix, $\boldsymbol{S}$, which serves multiple purposes. Its primary function is to provide a structure where "strength" between any pair of nodes in the "grid" is stored. This is used to decide which nodes are strongly interconnected so that they can be grouped together into small local aggregates. Another purpose of $\boldsymbol{S}$ is to treat a problem caused by anisotropy. The problem arises because the interpolation should be in the direction of strength, but the smoothing that is used to improve the interpolation operator can smear in the direction of weakness. $\boldsymbol{S}$ can be used to identify the potential for this smearing and filter smoothing by eliminating the weak connections in the matrix.

The SOC matrix is usually chosen with a sparsity pattern that is a subset of the original nodal matrix. This can be advantageous in the implementation because the size of the necessary memory allocation is known at the beginning of the construction process. In general, $\boldsymbol{S}$ is not needed after setup, so it can be discarded after that phase. Usually, the strength between nodes is defined in a way that allows $\boldsymbol{S}$ to be symmetric, and the cost of assembling $\boldsymbol{S}$ is reduced to constructing the upper (or lower) triangular part of the matrix. Classically, the strength between nodes is defined as

$$s_{ij} = \begin{cases} 1, & i = j, \\ 1, & \rho\left(\boldsymbol{A}_{ii}^{-1/2}\boldsymbol{A}_{ij}\boldsymbol{A}_{jj}^{-1/2}\right) > \theta \cdot \rho_{i,\max}, \\ 0, & \text{otherwise}, \end{cases}$$

where $\rho(\cdot)$ denotes the spectral radius of a matrix, $\theta \in (0, 1)$, and

$$\rho_{i,\max} = \max_{j \neq i} \rho\left(\boldsymbol{A}_{ii}^{-1/2}\boldsymbol{A}_{ij}\boldsymbol{A}_{jj}^{-1/2}\right)$$

s_{ij} effectively determines strength relative to other off-diagonals in row i. Also, $\boldsymbol{A}_{ij}$ here refers to the block (of size $b_1 \times b_1$) associated with a nonzero in the matrix between nodes i and j.

Based on $\boldsymbol{S}$, define the set $\mathcal{S}$ to be the *special nodes*, by which we mean those that are not strongly connected to any other node or that correspond to a row in the matrix that is very diagonally dominant. Relaxation leaves little to no error at the special nodes, so they need not be interpolated to or from coarser levels and are therefore gathered into one aggregate that is not affected by interpolation from coarser levels.

Next, to facilitate description of the aggregation process, let the set of fine-level nodes be represented by their indices, $\mathcal{D}_1 = \{1, 2, \ldots, n_1\}$. The next phase then constructs a fine-level partition, $\{\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_{n_2}\}$, of $\mathcal{D}_1 \setminus \mathcal{S}$ into disjoint aggregates:

$$\mathcal{D}_1 \setminus \mathcal{S} = \bigcup_{i=1}^{n_2} \mathcal{A}_i, \quad \mathcal{A}_i \cap \mathcal{A}_j = \emptyset, \quad \forall i \neq j.$$

Each aggregate here forms a node on the coarse level. Given the set of special nodes, this phase is accomplished by two passes through the nodes. In the first pass, an initial set of aggregates is formed and, in the second, all unaggregated nodes are assigned to existing aggregates.

Algorithm 8: *Form aggregates, pass 1*

Input: Set of nodes, $\mathcal{D}$, and SOC matrix, S.

$\mathcal{R} = \mathcal{D}$
$k = 0$
for $i = 1, \ldots, n_1$ **do**
    Form $\mathcal{N}_i = \{j : s_{ij} = 1, i \neq j\}$
    if $\mathcal{N}_i \cap \mathcal{R} = \mathcal{N}_i$ **then**
        $\mathcal{A}_k = \mathcal{N}_i \cup \{i\}$
        $k = k + 1$
        $\mathcal{R} = \mathcal{R} \setminus (\mathcal{N}_i \cup \{i\})$
    end if
end for
$n_2 = k$
return Aggregates, $\mathcal{A}_1, \ldots, \mathcal{A}_{n_2}$, and still un-aggregated nodes $\mathcal{R}$.

The goal of the first pass is to create a set of aggregates from a maximally independent set of strongly connected nodes. One way to do this is outlined here. Each node is examined once in turn, in any logical order. If none of the current node's strongly connected neighbors are in an aggregate, then they join the current node to form a new aggregate. Otherwise, the current node is left alone and the next node is examined similarly. More specifically, let $\mathcal{R}$ be the set of node indices that are not currently assigned to an aggregate. Initially, $\mathcal{R} = \mathcal{D}_1 \setminus \mathcal{S}$. Let $\mathcal{N}_i = \{j : s_{ij} = 1, i \neq j\}$ be the set of points that are strongly connected to point i. The first pass then proceeds as outlined in Algorithm 8.

After the initial set of aggregates is formed, a subset of unaggregated nodes, $\hat{\mathcal{R}}$, remains. The goal now is to assign the nodes in $\hat{\mathcal{R}}$ to aggregates in the list $\mathcal{A}_1, \ldots, \mathcal{A}_{n_2}$. This assignment can be done by looping over each aggregate and assigning to it all nodes left in $\hat{\mathcal{R}}$ that are strongly connected to one of its nodes. (An alternative is to loop over each node in $\hat{\mathcal{R}}$, assigning it to the aggregate that it is most strongly connected to.) All non-special nodes are strongly connected to at least one node, so this step ensures that they will all be aggregated. Each aggregate is represented by a node on the coarse level, so that level will have size n_2. This step is outlined in Algorithm 9.

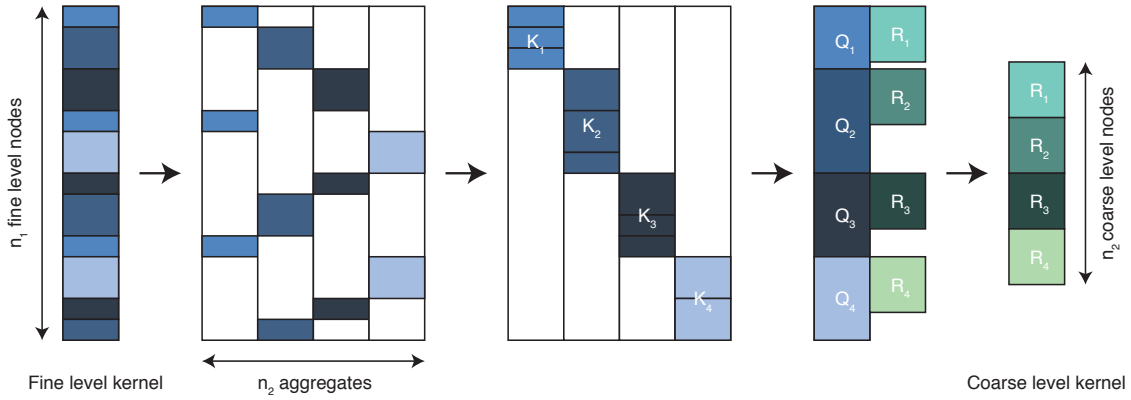


Figure 11.3: *To form the kernel for a coarse level, we start with the kernel for the fine level (far left), where the rows have been tagged according to which aggregate the corresponding node belongs to. All the rows with identical tags are then combined to form the* local *kernels (middle), and a thin QR decomposition is applied to each local kernel. The resulting* $\boldsymbol{Q}$ *matrices form the building blocks for the tentative interpolation operator,* $\hat{\boldsymbol{P}}$*, while the resulting* $\boldsymbol{R}$ *matrices form the building blocks for the coarse-level kernel (far right).* $\hat{\boldsymbol{P}}$ *is obtained by replacing each* $\boldsymbol{K}_i$ *matrix with the corresponding* $\boldsymbol{Q}_i$ *matrix and then permuting back to the original row ordering (second from the left).*

Algorithm 9: *Form aggregates, pass 2*

Input: Aggregates, $\mathcal{A}_1, \ldots, \mathcal{A}_{n_2}$, and still un-aggregated nodes $\mathcal{R}$.

for $i = 1, \ldots, n_2$ **do**
  Let m_i be the number of elements in $\mathcal{A}_i$
  for $j = 1, \ldots, m_i$ **do**
    Form $\mathcal{N}_j$
    Let $P_j = \mathcal{N}_j \cap \hat{\mathcal{R}}$
    for $k \in P_j$ **do**
      $\mathcal{A}_i = \mathcal{A}_i \cup \{k\}$
      $\hat{\mathcal{R}} = \hat{\mathcal{R}} \setminus \{k\}$
    end for
  end for
end for
return An independent set containing all nodes, $\mathcal{A}_1, \ldots, \mathcal{A}_{n_2}$.

Interpolation is constructed in two basic steps. The first involves choosing a *tentative* interpolation operator, $\hat{\boldsymbol{P}}$, while the second step consists of smoothing $\hat{\boldsymbol{P}}$. The tentative interpolation operator is chosen such that the set of near-kernel components, $\boldsymbol{K}$, is in the range of $\hat{\boldsymbol{P}}$, and $\hat{\boldsymbol{P}}$ does not connect neighboring aggregates, so $\hat{\boldsymbol{P}}^T\hat{\boldsymbol{P}} = \boldsymbol{I}$. The construction of $\hat{\boldsymbol{P}}$ is illustrated in Figure 11.3. Conceptually, assume that the nodes are ordered so that they are contiguous within each aggregate and in correspondence to the aggregate ordering. (This ordering is not necessary in practice, but used here

simply to facilitate the discussion.) The near kernel can then be decomposed into n_2 blocks denoted by $\boldsymbol{K}_1, \boldsymbol{K}_2, \ldots, \boldsymbol{K}_{n_2}$ and written in block form as $\boldsymbol{K} = \begin{bmatrix} \boldsymbol{K}_1^T & \boldsymbol{K}_2^T & \cdots & \boldsymbol{K}_{n_2}^T \end{bmatrix}^T$. This representation means that the number of rows of $\boldsymbol{K}_i$ equals the number of nodes in $\mathcal{A}_i$ times the nodal block size for the current level, and the number of columns equals the number, κ, of near-kernel components.

A local QR of each block can now be formed: $\boldsymbol{K}_i = \boldsymbol{Q}_i \boldsymbol{R}_i, \boldsymbol{Q}_i^T \boldsymbol{Q}_i = \boldsymbol{I}$, which yields the matrices $\boldsymbol{Q}_1, \ldots, \boldsymbol{Q}_{n2}$ and $\boldsymbol{R}_1, \ldots, \boldsymbol{R}_{n2}$. The columns of $\boldsymbol{Q}_i$ form a local basis spanning the near kernel in $\mathcal{A}_i$. Given this decomposition, the tentative interpolation operator, $\hat{\boldsymbol{P}}$, is formed via

$$\hat{\boldsymbol{P}} = \begin{bmatrix} \boldsymbol{Q}_1 & 0 & 0 & 0 \\ 0 & \boldsymbol{Q}_2 & 0 & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & \boldsymbol{Q}_{n_2} \end{bmatrix}, \quad \boldsymbol{R} = \begin{bmatrix} \boldsymbol{R}_1 \\ \boldsymbol{R}_2 \\ \vdots \\ \boldsymbol{R}_{n_2} \end{bmatrix}.$$

Here, $\hat{\boldsymbol{P}}^T \hat{\boldsymbol{P}} = \boldsymbol{I}$ by construction.

A coarse near kernel must be determined to allow for recursion to coarser levels. But $\boldsymbol{K} = \hat{\boldsymbol{P}}\boldsymbol{R}$ means that the fine-level near kernel can be exactly represented on the coarse level by simply choosing $\boldsymbol{K}_c = \boldsymbol{R}$. Note then that, with $\boldsymbol{A}_c = \hat{\boldsymbol{P}}^T \boldsymbol{A} \hat{\boldsymbol{P}}$, we have $\boldsymbol{A}_c \boldsymbol{K}_c = \hat{\boldsymbol{P}}^T \boldsymbol{A} \hat{\boldsymbol{P}} \boldsymbol{R} \approx \hat{\boldsymbol{P}}^T \boldsymbol{0} = \boldsymbol{0}$ since $\boldsymbol{A}\boldsymbol{K} \approx \boldsymbol{0}$.

As discussed in Section 10.5, this local non-overlapping use of the near kernel may generally satisfy the weak approximation property, but not the strong one needed to ensure optimal V-cycle performance. To improve accuracy of the interpolation operator, we therefore smooth it by applying the weighted Jacobi error propagation matrix: $\boldsymbol{P} = \left(\boldsymbol{I} - \omega \boldsymbol{D}^{-1} \boldsymbol{A}\right) \hat{\boldsymbol{P}}$. The block diagonal matrix, $\boldsymbol{D}$, whose block diagonal agrees with that of $\boldsymbol{A}$, is used here because it does not change the sparsity pattern of $\boldsymbol{P}$ and it responds better to the local nature of $\boldsymbol{A}$. A typical choice for ω is $\frac{4}{3\rho(\boldsymbol{D}^{-1}\boldsymbol{A})}$, with care needed in estimating $\rho(\boldsymbol{D}^{-1}\boldsymbol{A})$ as discussed in Section 11.3. Smoothed interpolation, while generally causing overlap in the aggregate basis functions so that $\boldsymbol{P}^T \boldsymbol{P} \neq \boldsymbol{I}$, often leads to an optimal V-cycle algorithm. Also, the smoothed near kernel is exactly in the range of smoothed interpolation: $\boldsymbol{P}\boldsymbol{K}_c = (\boldsymbol{I} - \omega \boldsymbol{D}^{-1}\boldsymbol{A})\hat{\boldsymbol{P}}\boldsymbol{K}_c = (\boldsymbol{I} - \omega \boldsymbol{D}^{-1}\boldsymbol{A})\boldsymbol{K}$, which generally preserves and even improves the near-kernel components in $\boldsymbol{K}$. While the finest-level matrix has nodes with b_1 degrees of freedom each, all coarser levels have κ degrees of freedom associated with each finer-level aggregate. The complexity of the coarse level is thus dictated by the number of near-kernel components and the aggressiveness of coarsening (that is, the size of the aggregates). Both choices must be controlled so that the coarse-level matrix has substantially fewer nonzero entries than the next finer-level matrix has.

The above steps outline how a given matrix and near kernel pair $\boldsymbol{A}, \boldsymbol{K}$ are decomposed to form a coarse level, the operators between the two levels, and the appropriate coarse matrix and near kernel. The combined process is summarized in Algorithm 10. The coarsening routine is applied recursively until there is a coarse matrix that can be easily inverted through iteration or a direct solver. Because aggregation coarsens aggressively, the number of levels is usually small, between three to six levels for all our tests.

Algorithm 10: *Form a coarse level in the SA hierarchy*

Input: The matrix and kernel for this level, $\boldsymbol{A}$ and $\boldsymbol{K}$.
1: Precompute inverse $\boldsymbol{D}^{-1}$ of block diagonal of $\boldsymbol{A}$
2: Find spectral radius of $\boldsymbol{D}^{-1}\boldsymbol{A}$
3: Smooth kernel $\boldsymbol{K}$ to match boundary conditions
4: Form a matrix $\boldsymbol{S}$ to determine strength
5: Use $\boldsymbol{S}$ to form aggregates from nodes in $\boldsymbol{A}$
6: For each aggregate form local QR of $\boldsymbol{K}$
7: Use the local Q blocks to form tentative interpolation, $\hat{\boldsymbol{P}}$
8: Smooth the tentative interpolation to get $\boldsymbol{P}$
9: Use $\boldsymbol{P}$ to form $\boldsymbol{A}_c = \boldsymbol{P}^T\boldsymbol{A}\boldsymbol{P}$
10: Use local R blocks to form coarse kernel $\boldsymbol{K}_c$
11: **return** Interpolation and restriction operators, $\boldsymbol{P}, \boldsymbol{R}$ as well as the coarse matrix and kernel, $\boldsymbol{A_c}, \boldsymbol{K_c}$.

11.3 Null space

In the previous section, we tacitly assumed that the near kernel for the fine-grid problem is known. While this is not always the case, near-kernel components can be obtained for many problems by examining the underlying PDE as mentioned briefly before. For example, for elasticity, if we ignore boundary conditions, then it is well-known that a rigid-body mode (constant displacement or rotation) has zero strain and, therefore, zero stress. So rigid-body modes that are not prevented by boundary conditions from being in the domain of the PDE operator are actually in its kernel. Fixed boundary conditions prevent these modes from being admissible, so they cannot, of course, be kernel components in any global sense. But any rigid-body mode that is altered to satisfy typical conditions at the boundary becomes a near-kernel component, and they can usually be used locally to represent all near-kernel components.

To be more specific, displacements for *linear* elasticity are assumed to be small, thereby simplifying computation of the strain tensor. In particular, the strain is given by $\boldsymbol{\epsilon} = \frac{1}{2}(\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^T)$, where $\boldsymbol{u}$ is the displacement

field. In this case, it is easy to verify that the following vector functions (which represent rotations around each of the three axes) all lead to zero strain :$\boldsymbol{u} = (0, -z, y)$, $\boldsymbol{u} = (z, 0, -x)$, and $\boldsymbol{u} = (-y, x, 0)$. Here, (x, y, z) represents material (undeformed) coordinates. Similarly the following three (constant) vector functions which represent translations along the three axes also leas to zero strain : $\boldsymbol{u} = (1, 0, 0)$, $\boldsymbol{u} = (0, 1, 0)$, and $\boldsymbol{u} = (0, 0, 1)$. Since these rigid-body modes are linear functions, they should be well approximated in the interior of the domain by most finite element discretization methods. Indeed, if the finite element space includes linear functions, then these modes are exactly represented in the discretization except in the elements abutting fixed boundaries. All that is needed in this case is to *interpolate* these rigid-body modes at the nodes. In doing so, we make them admissible and retain their near-kernel nature by forcing their values at the boundary nodes to satisfy any fixed conditions that might be there. For further assurance that they remain near-kernel components, a relaxation sweep may be applied, using them as initial guesses to the solution of the homogeneous equation.

As noted above, this discussion assumes linear elasticity. However, our limited experience suggests that the rigid-body modes that work for those equations may be good candidates for the linearized equations of nonlinear elasticity. We have not examined this issue in any depth, so it may benefit from additional work. One thing to add is that just as relaxation sweeps may improve nearness to the kernel for modes that are altered to satisfy the boundary conditions and lie in the finite element space, so too may relaxation benefit the linear elasticity modes used for the nonlinear case.

To the extent that this approximation proves not to be good enough, then there are adaptive SA methods that attempt to discover the near kernel as part of the setup process, [Brezina et al., 2004; Brandt et al., 2011]. These methods have computationally expensive setup costs, which for this problem would be on the order of three times what it is for SA, not counting the need for an additional 10 cycles. To amortize this cost, convergence from an adaptively found kernel would need to be near optimal. We did experiment with an adaptive approach for this discretization, but, while increased convergence rates were indeed attainable, the extra setup cost made time to solution slower overall.

11.4 Smoothing

We use multigrid as a preconditioner for conjugate gradients (CG) rather than as a stand-alone solver. As a consequence, the multigrid preconditioner

that we use must be symmetric and positive definite. This requirement has multiple implications. To ensure symmetry, the V-cycle that we use must be energy-symmetric and, to ensure positive definiteness, it is critical that the smoother be convergent in energy. (See Appendix A for theory that proves this claim.) While these requirements are not surprising, it is important to take the necessary steps to ensure that they are satisfied.

The basic relaxation scheme that we use is a Chebyshev smoother, which amounts to running a fixed number of Chebyshev iterations at each level based on $\boldsymbol{D}^{-1}\boldsymbol{A}$, [Golub and Varga, 1961]. A nice introduction to this smoother can be found in [Adams et al., 2003]. From a theoretical point of view, it has good properties and, in practice, it also performs the best among available smoothers as shown in [Adams et al., 2003; Baker et al., 2011]. Most importantly, it has the property of being implementable with mat-vec operations and is thus relatively easy to parallelize compared to other smoothers like Gauss-Seidel.

Chebyshev requires an upper bound for the spectral radius of $\boldsymbol{D}^{-1}\boldsymbol{A}$ and an estimate of the smallest part of the spectrum that we wish to attenuate, the same as needed for smoothing the interpolation operator by weighted Jacobi. One approach to obtaining these estimates is to use the power method for computing the largest eigenvalue of a matrix. Unfortunately, this does not generally provide a rigorous bound on the largest eigenvalue, and the convergence rate of the power method is limited by the ratio of the two largest eigenvalues, [Golub and Loan, 1983]. In practice, these two eigenvalues are often very close, so that convergence is very slow, which leads to a trade-off: too loose of an approximation to the spectral radius yields slow Chebyshev smoothing rates, while tighter approximations can be costly.

Another possibility is to use Gerschgorin's theorem to estimate the largest eigenvalue, but this approximation is too loose for our requirements. A potentially better alternative is to use Lanczos's method, [Golub and Loan, 1983]. However, while its convergence rate is better, it is also more expensive per iteration and may require careful numerical treatment to ensure convergence (e. g., periodic re-orthogonalization of the Krylov basis).

In the end, we need an approximation to the spectral radius of $\boldsymbol{D}^{-1}\boldsymbol{A}$, rather than $\boldsymbol{A}$. Since $\boldsymbol{D}^{-1}\boldsymbol{A}$ is not symmetric in the traditional sense we cannot apply the standard Lanczos algorithm. However, $\boldsymbol{D}^{-1}\boldsymbol{A}$ *is* symmetric in the energy inner product defined by $\langle \boldsymbol{u}, \boldsymbol{v} \rangle_{\boldsymbol{A}} = \langle \boldsymbol{u}, \boldsymbol{A}\boldsymbol{v} \rangle$. In fact, for any symmetric positive definite preconditioner $\boldsymbol{M}$, we have

$$\langle \boldsymbol{u}, \boldsymbol{M}^{-1}\boldsymbol{A}\boldsymbol{v} \rangle_{\boldsymbol{A}} = \langle \boldsymbol{u}, \boldsymbol{A}\boldsymbol{M}^{-1}\boldsymbol{A}\boldsymbol{v} \rangle = \langle \boldsymbol{M}^{-1}\boldsymbol{A}\boldsymbol{u}, \boldsymbol{A}\boldsymbol{v} \rangle = \langle \boldsymbol{M}^{-1}\boldsymbol{A}\boldsymbol{u}, \boldsymbol{v} \rangle_{\boldsymbol{A}}.$$

$\boldsymbol{M}^{-1}\boldsymbol{A}$ is also positive definite in energy because

$$\langle \boldsymbol{u}, \boldsymbol{M}^{-1}\boldsymbol{A}\boldsymbol{u} \rangle_{\boldsymbol{A}} = \langle \boldsymbol{u}, \boldsymbol{A}\boldsymbol{M}^{-1}\boldsymbol{A}\boldsymbol{u} \rangle > 0$$

for any $\boldsymbol{u} \neq \boldsymbol{0}$. Its eigenvalues are therefore positive and real, so its spectral radius can be computed by finding the largest λ such that $\boldsymbol{M}^{-1}\boldsymbol{A}\boldsymbol{x} = \lambda \boldsymbol{x}, \boldsymbol{x} \neq \boldsymbol{0}$, which is clearly equivalent to the generalized eigenvalue problem $\boldsymbol{A}\boldsymbol{x} = \lambda \boldsymbol{M}\boldsymbol{x}, \boldsymbol{x} \neq \boldsymbol{0}$. We apply the generalized Lanczos algorithm, [van der Vorst, 1982], to this generalized eigenvalue problem to compute the spectral radius of $\boldsymbol{D}^{-1}\boldsymbol{A}$.

11.5 Prefiltering of equality constraints

A challenge for many of the existing multigrid methods developed for cloth simulation is proper handling of constraints. As shown in [Boxerman and Ascher, 2004], superior performance is achieved when the preconditioner for CG is based not on the full system, but rather on the *constraint null space*, by which we mean the null space of the constraint operator. In the context of the original paper by Baraff and Witkin [1998] the constraints are all equality constraints. The method they propose constructs a pre-filtered matrix restricted to the constraint null space, but it is neither symmetric nor easily treated by our multigrid approach. A reduced set of equations can be constructed based on a null-space basis for the constraints, but this leads to a system where the block size is no longer constant. While this may seem like a minor inconvenience, it leads to either a substantial increase in code complexity and a reduced ability generate highly optimized code or reduced performance due to less coherent memory access patterns. In fact, with a constant block size, we can use BSR (block sparse row) matrices, where the innermost operations are effectively BLAS3 operations with high arithmetic intensity. By comparison, to use standard libraries for matrices with varying block size, we would have to use CSR matrices with comparatively low arithmetic intensity.

To obtain a system with a constant block size, we form a reduced system, but replace all eliminated variables with dummy ones. This retains the block structure while ensuring that our preconditioner operates on the constraint null space. We refer to this method as Pre-filtered Preconditioned CG (PPCG).

To explain PPCG, the "modified" linear system solved by Baraff and Witkin

[1998] can be written in the notation from [Ascher and Boxerman, 2003] as

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & \|\boldsymbol{S}(\boldsymbol{b}-\boldsymbol{A}\boldsymbol{x})\| \\ \text{s.t.} \quad & (\boldsymbol{I}-\boldsymbol{S})\boldsymbol{x} = (\boldsymbol{I}-\boldsymbol{S})\boldsymbol{z}, \end{aligned} \tag{11.1}$$

where: $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ is the matrix for the full system; $\boldsymbol{b}$ is the corresponding right-hand side; $\boldsymbol{S}$, which represents the filtering operation in [Baraff and Witkin, 1998], is the orthogonal projection matrix onto the constraint null space (not to be confused with the SOC matrix in Section 11.2); and $\boldsymbol{z}$ is a vector of the desired values for the constrained variables. It is assumed that $\boldsymbol{A}$ is symmetric positive definite.

Due to the constraint, any feasible point must satisfy

$$\boldsymbol{x} = \boldsymbol{S}\boldsymbol{x} + (\boldsymbol{I}-\boldsymbol{S})\boldsymbol{x} = \boldsymbol{S}\boldsymbol{x} + (\boldsymbol{I}-\boldsymbol{S})\boldsymbol{z}.$$

To incorporate this expression into the objective function, first note that

$$\boldsymbol{S}(\boldsymbol{b}-\boldsymbol{A}\boldsymbol{x}) = \boldsymbol{S}\boldsymbol{b} - \boldsymbol{S}\boldsymbol{A}(\boldsymbol{S}\boldsymbol{x} + (\boldsymbol{I}-\boldsymbol{S})\boldsymbol{z}) = \boldsymbol{S}(\boldsymbol{b}-\boldsymbol{A}\boldsymbol{z}) - \boldsymbol{S}\boldsymbol{A}\boldsymbol{S}(\boldsymbol{x}-\boldsymbol{z}).$$

By introducing $\boldsymbol{c} \equiv \boldsymbol{b}-\boldsymbol{A}\boldsymbol{z}$ and $\boldsymbol{y} \equiv \boldsymbol{x}-\boldsymbol{z}$, we can then rewrite the constrained minimization problem in Eq. (11.1) as

$$\begin{aligned} \min_{\boldsymbol{y}} \quad & \|\boldsymbol{S}\boldsymbol{c} - \boldsymbol{S}\boldsymbol{A}\boldsymbol{S}\boldsymbol{y}\| \\ \text{s.t.} \quad & (\boldsymbol{I}-\boldsymbol{S})\boldsymbol{y} = \boldsymbol{0}. \end{aligned} \tag{11.2}$$

By construction, $\boldsymbol{S}$ is symmetric and, therefore, diagonalizable. Since it is also orthogonal, it follows that the eigenvalues must be either 0 or 1. We can thus compute another orthogonal matrix, $\boldsymbol{Q}$, such that

$$\boldsymbol{S} = \boldsymbol{Q}\begin{pmatrix} \boldsymbol{I}_r & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} \end{pmatrix}\boldsymbol{Q}^T \equiv \boldsymbol{Q}\boldsymbol{J}\boldsymbol{Q}^T, \quad r = \dim(\text{range}(\boldsymbol{S})).$$

If we now partition $\boldsymbol{Q}$ into $\boldsymbol{V} \in \mathbb{R}^{n\times r}$ and $\boldsymbol{W} \in \mathbb{R}^{n\times n-r}$ such that $\boldsymbol{Q} = (\boldsymbol{V}, \boldsymbol{W})$, then $\boldsymbol{V}$ is a basis for the constraint null space while $\boldsymbol{W}$ is a basis for the constrained subspace. From this decomposition, it follows that $\boldsymbol{S} = \boldsymbol{V}\boldsymbol{V}^T$ and $\boldsymbol{I}-\boldsymbol{S} = \boldsymbol{W}\boldsymbol{W}^T$.

Similarly, let

$$\boldsymbol{d} \equiv \begin{pmatrix} \boldsymbol{d}_1 \\ \boldsymbol{d}_2 \end{pmatrix} = \boldsymbol{Q}^T\boldsymbol{c}, \quad \boldsymbol{u} \equiv \begin{pmatrix} \boldsymbol{u}_1 \\ \boldsymbol{u}_2 \end{pmatrix} = \boldsymbol{Q}^T\boldsymbol{y}.$$

Using the last definition, we have $\boldsymbol{y} = \boldsymbol{V}\boldsymbol{u}_1 + \boldsymbol{W}\boldsymbol{u}_2$ and, therefore,

$$\boldsymbol{V}\boldsymbol{V}^T\boldsymbol{y} = \boldsymbol{V}\boldsymbol{u}_1 \quad \text{and} \quad \boldsymbol{W}\boldsymbol{W}^T\boldsymbol{y} = \boldsymbol{W}\boldsymbol{u}_2. \tag{11.3}$$

Combining the above definitions and substituting $\boldsymbol{QJQ}^T$ for $\boldsymbol{S}$, we can rewrite the objective function in Eq. (11.2) as follows :

$$\begin{aligned}\phi &= \|\boldsymbol{Sc} - \boldsymbol{SASy}\| \\ &= \|\boldsymbol{QJQ}^T\boldsymbol{c} - \boldsymbol{QJQ}^T\boldsymbol{AQJQ}^T\boldsymbol{y}\| \\ &= \|\boldsymbol{JQ}^T\boldsymbol{c} - \boldsymbol{JQ}^T\boldsymbol{AQJQ}^T\boldsymbol{y}\| \\ &= \|\boldsymbol{Jd} - \boldsymbol{JQ}^T\boldsymbol{AQJu}\| \\ &= \left\| \begin{pmatrix} \boldsymbol{d}_1 \\ \boldsymbol{0} \end{pmatrix} - \begin{pmatrix} \boldsymbol{V}^T\boldsymbol{AV} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} \end{pmatrix} \begin{pmatrix} \boldsymbol{u}_1 \\ \boldsymbol{u}_2 \end{pmatrix} \right\|.\end{aligned}$$

This system can clearly be reduced by eliminating $\boldsymbol{u}_2$ since any value of $\boldsymbol{u}_2$ produces the same value of ϕ. However, eliminating $\boldsymbol{u}_2$ creates a smaller system, which means that if $\boldsymbol{A}$ is block sparse with fixed block size, then the reduced system will in general be block sparse with different block sizes. To keep the original size of the system we could leave all the zero blocks in place, but the resulting system would then be singular, thereby introducing other problems. On the other hand, we know from Eq. (11.2) that $(\boldsymbol{I} - \boldsymbol{S})\boldsymbol{y} = \boldsymbol{0}$ and, since $(\boldsymbol{I} - \boldsymbol{S})\boldsymbol{y} = \boldsymbol{WW}^T\boldsymbol{y} = \boldsymbol{Wu}_2$, it follows that $\boldsymbol{u}_2 = \boldsymbol{0}$. Minimizing ϕ subject to the desired constraint is therefore equivalent to solving the following linear system :

$$\begin{pmatrix} \boldsymbol{V}^T\boldsymbol{AV} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I} \end{pmatrix} \begin{pmatrix} \boldsymbol{u}_1 \\ \boldsymbol{u}_2 \end{pmatrix} = \begin{pmatrix} \boldsymbol{d}_1 \\ \boldsymbol{0} \end{pmatrix}. \tag{11.4}$$

Rotating this back to our original coordinates yields

$$\boldsymbol{Q} \begin{pmatrix} \boldsymbol{V}^T\boldsymbol{AV} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I} \end{pmatrix} \boldsymbol{Q}^T\boldsymbol{Q} \begin{pmatrix} \boldsymbol{u}_1 \\ \boldsymbol{u}_2 \end{pmatrix} = \boldsymbol{Q} \begin{pmatrix} \boldsymbol{d}_1 \\ \boldsymbol{0} \end{pmatrix}$$

or, equivalently,

$$\left(\boldsymbol{VV}^T\boldsymbol{AVV}^T + \boldsymbol{WW}^T\right)\boldsymbol{y} = \boldsymbol{Vd}_1 = \boldsymbol{VV}^T\boldsymbol{c}.$$

Since $\boldsymbol{S} = \boldsymbol{VV}^T$ and $\boldsymbol{I} - \boldsymbol{S} = \boldsymbol{WW}^T$, we finally arrive at

$$(\boldsymbol{SAS} + \boldsymbol{I} - \boldsymbol{S})\,\boldsymbol{y} = \boldsymbol{Sc}. \tag{11.5}$$

The importance of Eq. (11.5) is that we now have a *symmetric positive definite* (in particular, full rank) system of the same dimensions as the original system, but which correctly projects out all of the constraints. From this system, the solution to Eq. (11.1) is easily recovered as $\boldsymbol{x} = \boldsymbol{y} + \boldsymbol{z}$.

Furthermore, the condition number of the new system is no worse than that of $\boldsymbol{A}$, and may in fact be better. This conclusion is based on the assumption that 1 is in the *field of values* of $\boldsymbol{A}$, which is defined to be the real numbers inclusively between the smallest and largest eigenvalues of $\boldsymbol{A}$. (This

assumption often holds for PDEs; otherwise, we can simply multiply $\boldsymbol{I} - \boldsymbol{S}$ by a scalar that is in $\boldsymbol{A}$'s field of values.) To prove this assertion, first note that our assumption implies that the field of values of $\boldsymbol{V}^T\boldsymbol{A}\boldsymbol{V}$ is the same as that of the matrix in Eq. (11.4), which is in turn the same as that of the matrix in Eq. (11.5) because they are related by a similarity transformation. By Cauchy's interlacing theorem, [Horn and Johnson, 1985], the field of values of $\boldsymbol{V}^T\boldsymbol{A}\boldsymbol{V}$ is a subset of that of $\boldsymbol{A}$, which immediately implies that the condition number of $\boldsymbol{V}^T\boldsymbol{A}\boldsymbol{V}$ is bounded by the condition number of $\boldsymbol{A}$.

For the constraints considered by Baraff and Witkin [1998], $\boldsymbol{S}$ is block diagonal, so the computation of $\boldsymbol{S}\boldsymbol{A}\boldsymbol{S}$ amounts to simple blockwise row and column scaling of $\boldsymbol{A}$, while the addition of $\boldsymbol{I} - \boldsymbol{S}$ only affects the block diagonal. It should be noted that, while the derivation required $\boldsymbol{S}$ to be diagonalized, the final result does not. We refer to the system in Eq. (11.5) as the prefiltered system, to which we apply a standard preconditioned CG algorithm.

The constraints considered by Baraff and Witkin [1998] are limited to (cloth) vertices against (collision) objects. However, the above method easily generalizes to any kind of equality constraint, including the face-vertex and edge-edge constraints used in other collision response algorithms (e.g., [Harmon et al., 2008; Otaduy et al., 2009]).

Finally, it should be noted that none of the derivations in this section depend on multigrid methods: the results can be used with any type of linear solver. However, by using Eq. (11.5) with smoothed aggregation, we have an effective way of coarsening not just the dynamics, but also the constraints.

11.6 Implementation

In the remainder of this chapter, we evaluate our multigrid solver within the context of a production quality cloth solver.

The simulator we use for our experiments is based on a combination of the methods presented by Baraff and Witkin [1998] and Bridson et al. [2002]. In particular, it uses the material model from [Baraff and Witkin, 1998], and the basic contact handling is as described in section 6 of that paper (including approximate handling of friction). Since that does not guarantee that the simulation is free of continuous time collisions at the end of a time step, the linear solve is followed by one or more loops of Bridson's method [Bridson et al., 2002, section 7.4]. If this still fails to resolve all collisions, then the fail-safe in [Harmon et al., 2008] is applied. Only the contributions from Baraff and Witkin's paper actually affect what goes into the linear system that our

solver sees. This includes constraints for cloth/object collisions and repulsion forces for cloth/cloth collisions (the latter being akin to the repulsion forces in section 7.2 of [Bridson et al., 2002] but included into the implicit solve as originally suggested by Baraff and Witkin [1998]). While multigrid methods can be used as stand-alone solvers, we found it more effective in our system to use multigrid as a preconditioner within CG. The only change we made to our system is therefore within the CG method. All other features and limitations remain the same.

Our implementation of SA depends critically on Intel's MKL v11.3 for highly optimized implementations of most of the basic linear algebra operations like sparse matrix-vector and matrix-matrix products as well as QR decompositions. To achieve good performance, many structures are pre-allocated and re-used throughout the simulation, leading to slightly higher memory consumption, but not excessively so. The code is somewhat parallelized, but is generally limited by memory bandwidth rather than CPU resources. Additional improvements are definitely possible.

One of the most expensive operations in any Galerkin multigrid algorithm is forming the coarse-grid operator since it is based on a triple-matrix product. Despite the symmetric nature of this product, it is currently most efficiently implemented using two sparse mat-mat multiply operations. We use MKL for this purpose, while others have explored doing it on a GPU [Dalton et al., 2015]. Regardless of the implementation of the matrix products, care has to be taken when constructing the interpolation matrix in the presence of special nodes. The range of interpolation is the entire fine level and must contain zeros for all special nodes. However, in a naive implementation, if these zero entries in the interpolation matrix are not pruned, then the mat-mat products will introduce structural fill-in that can significantly affect the run time, especially after smoothing the interpolation operator.

11.7 Examples

To evaluate and compare our approach to existing methods, we consider five procedurally generated examples at different resolutions, and a high-resolution production example to show its practical applicability. The five simple examples are shown in Fig. 11.4, while the production example is shown in Fig. 11.1.

The first example is a fully pinned piece of cloth subjected to gravity. The corresponding PDE is fully elliptic with a full Dirichlet boundary, meaning in part that it is positive definite with an order 1 constant. The cloth that

Figure 11.4: *The five examples shown here represent problems of increasing difficulty that we use for benchmarking. They are generated procedurally, with the vertex count ranging from* $1,000$ *to* $1,000,000$*. The ones shown here have* $40,000$ *vertices.*

is pinned on two sides in the second example has a smaller constant and thus provides a test to see how sensitive our methods are to variations in the strength of ellipticity. The third example has a re-entrant corner, which is more difficult yet because it does not possess full ellipticity, which means that the standard approximation properties discussed above do not apply. Standard discretization and solution methods have serious difficulties in the presence of re-entrant corners, so they provide an especially challenging test problem for our methodology. In the fourth example, the cloth falls flat on a flat box with a hole in the middle. This generates many contact constraints and thus illustrates the performance of our PPCG method well. Finally, in the last example, we drop the cloth on the same box, but this time from a vertical configuration. In this way, we observe plenty of buckling and also lots of cloth-cloth collisions. The examples are referred to as *pinned*, *drooping*, *re-entrant*, *dropHorizontal*, and *dropVertical*, respectively. Each of the five simple examples were run with both regular and irregular tessellations.

11.8 Evaluation

All simulations were run on dual socket systems with Intel(R) Xeon(R) E5-2698 v3 @ 2.30GHz processors, each processor with 16 cores and each system configured with 64 GB DDR4 RAM. All simulations were run with a time step of $\Delta t = 2$ ms. The stopping criterion used in the CG method is a small relative residual error : $\|\boldsymbol{r}_i\|_{\boldsymbol{M}^{-1}} / \|\boldsymbol{r}_0\|_{\boldsymbol{M}^{-1}} < \epsilon$, where $\boldsymbol{M}$ is the chosen preconditioner, $\|\cdot\|_{\boldsymbol{M}^{-1}} \equiv \|\boldsymbol{M}^{-\frac{1}{2}} \cdot\|$, $\boldsymbol{r}_i$ is the residual after i iterations, and ϵ is a given tolerance, which we set to 10^{-5}.

An important point to keep in mind is that the relative residual error used to judge convergence gives an unfair advantage to poor preconditioners like the block diagonal ones. This observation comes from first realizing that the relative residual in practice is only as good as how tight it bounds the relative energy norm of the error. Remembering that $\boldsymbol{r} = \boldsymbol{A}\boldsymbol{e}$, we have the

bound

$$\frac{\|\boldsymbol{e}_i\|_{\boldsymbol{A}}}{\|\boldsymbol{e}_0\|_{\boldsymbol{A}}} = \frac{\|\boldsymbol{A}^{-\frac{1}{2}}\boldsymbol{r}_i\|}{\|\boldsymbol{A}^{-\frac{1}{2}}\boldsymbol{r}_0\|} = \frac{\|\boldsymbol{A}^{-\frac{1}{2}}\boldsymbol{M}^{\frac{1}{2}}\boldsymbol{M}^{-\frac{1}{2}}\boldsymbol{r}_i\|}{\|\boldsymbol{A}^{-\frac{1}{2}}\boldsymbol{M}^{\frac{1}{2}}\boldsymbol{M}^{-\frac{1}{2}}\boldsymbol{r}_0\|} \leq C\frac{\|\boldsymbol{r}_i\|_{\boldsymbol{M}^{-1}}}{\|\boldsymbol{r}_0\|_{\boldsymbol{M}^{-1}}},$$

where $C = \sqrt{\frac{\lambda_{\max}}{\lambda_{\min}}}$, with $\lambda_{\max}$ and $\lambda_{\min}$ the respective largest and smallest eigenvalues of $\boldsymbol{M}^{-\frac{1}{2}}\boldsymbol{A}\boldsymbol{M}^{-\frac{1}{2}}$ or, equivalently, of $\boldsymbol{M}^{-1}\boldsymbol{A}$. This bound means that the practical error measure that we use is sharp only up to the square root of spectral condition number of $\boldsymbol{M}^{-1}\boldsymbol{A}$. If $\boldsymbol{M}$ is a reasonable approximation to $\boldsymbol{A}$ as it hopefully is for our multigrid scheme, then this bound is pretty tight. If $\boldsymbol{M}$ is not a very accurate approximation to $\boldsymbol{A}$, as when $\boldsymbol{M}$ is its diagonal part, then we may believe that we have converged well before the true relative energy norm of the error is sufficiently small. Said differently, diagonally preconditioned iteration has the smoothing property that the residuals they produce are very small compared to the actual error. More precisely, as we showed in Chapter 10, relaxation produces an error whose relative energy norm is much larger than its relative residual norm. In other words, the smooth error left after relaxation is hidden in the residual because it consists mostly of low eigenmodes of $\boldsymbol{A}$. SA tends much more to balance the error among the eigenmodes, so a small relative residual for SA is much more reflective of small relative energy error. This should be kept in mind in the consideration of the results we present below.

The size of the cloth in all our examples is 1m $\times$ 1m and the material parameters are the defaults used by our production solver, which approximates cotton. We use $\theta = 0.48$, but have experimented with other values and found the results to be fairly insensitive to the exact choice. If a suboptimal value is chosen, then the convergence rate still tends to be good, but the time to solution suffers from excessive amounts of fill-in on the coarser levels. For smoothing, we use a single sweep of Chebyshev with a second-order polynomial. Higher-order polynomials improve the convergence rate significantly, but at too high a computational price. For the spectral radius estimation, we use 10 iterations of the generalized Lanczos method. Once again, the convergence rate improves if this number is increased, but not in a computationally cost-effective way. For the kernel, we picked six vectors, one constant for each of the unknowns and the three rotations described in section 5.

The benchmark numbers are averages over all the solves required for 10 frames of animation for the pinned, drooping, and re-entrant examples, while we used 15 frames for dropHorizontal and 25 frames for dropVertical. These durations are chosen to capture most of the interesting behavior

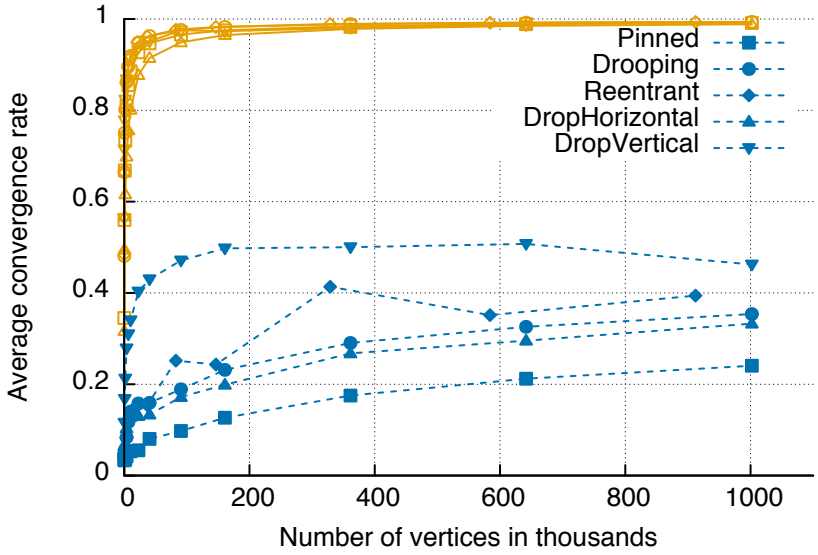


Figure 11.5: *The average convergence rate over the length of the entire animation for each of our examples. The orange curves are for Diag-PCG while the blue curves are for SA+PPCG.*

for each of the examples. In the following, we refer to block diagonally preconditioned conjugate gradients simply as *Diag-PCG*, while we refer to our smoothed aggregation preconditioned conjugate gradient method with prefiltering as *SA+PPCG*.

11.8.1 Convergence rates

The expectation for a multigrid preconditioner is that it should improve the convergence rate of the conjugate gradient method significantly and that the convergence rate should be independent of (or only weakly dependent on) the problem size. To investigate this, define the convergence rate at iteration i by $\rho_i = \|r_i\|_{M^{-1}} / \|r_{i-1}\|_{M^{-1}}$. The average convergence rate, ρ, within a single solve is then the geometric mean of these values. For an entire animation, we refer to the average convergence rate as the simple average over all solves of ρ.

Figure 11.5 shows average convergence rates of SA+PPCG and Diag-PCG for each of our five examples. While Diag-PCG approaches a convergence rate close to 1 very quickly, our method stays bounded away from 1 at significantly lower values. Note that the pinned example converges faster than the drooping example, which in turn converges faster than the re-entrant example. This observation is in agreement with our expectations based on the

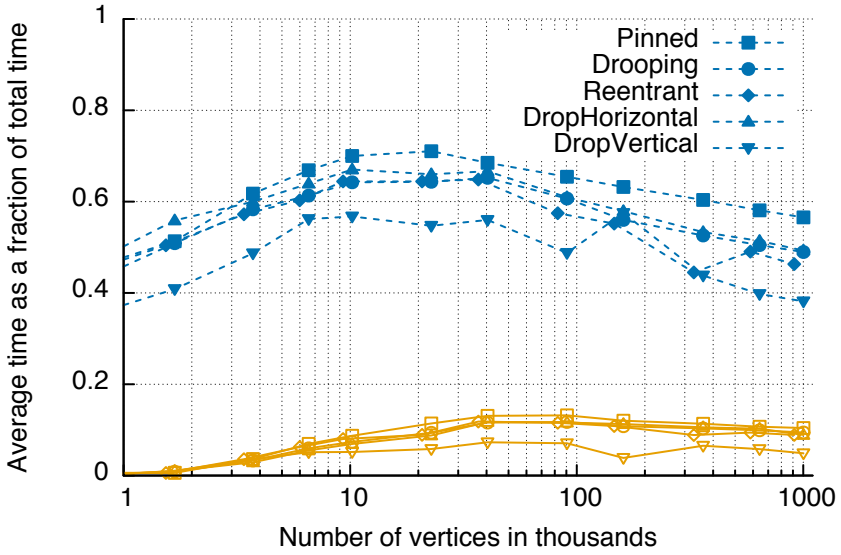


Figure 11.6: *The average time for prefiltering and setup as a percentage of the total solve time. The combined preprocessing time is the sum of the two. Setup is shown in blue while prefiltering is shown in orange. The corresponding total solve time is shown in Fig. 11.7.*

underlying PDEs. Note the irregularity of the curve for the re-entrant corner example, due perhaps to the loss of ellipticity.

11.8.2 Setup time

To achieve the better convergence rates shown above, we need both setup and prefiltering. The computational cost of both of these is illustrated in Fig. 11.6. The setup cost for AMG-type methods is often dominated by the cost of the triple-matrix product when forming the coarse matrices. In our case, this is still a significant cost, but not the dominant one. The computation of the spectral radius estimates is currently expensive and the computation of aggregates remains entirely serial. As such, there is room for improvement of these numbers.

To reduce the setup time, it is possible to only update the preconditioner periodically or adaptively based on the observed convergence rate. We experimented with only performing an update if the convergence rate after four iterations was substantially less than in previous solve. However, we found the cost of restarting the solver to outweigh the benefit. A more sophisticated controller design might make adaptive setup more beneficial, but this remains future work.

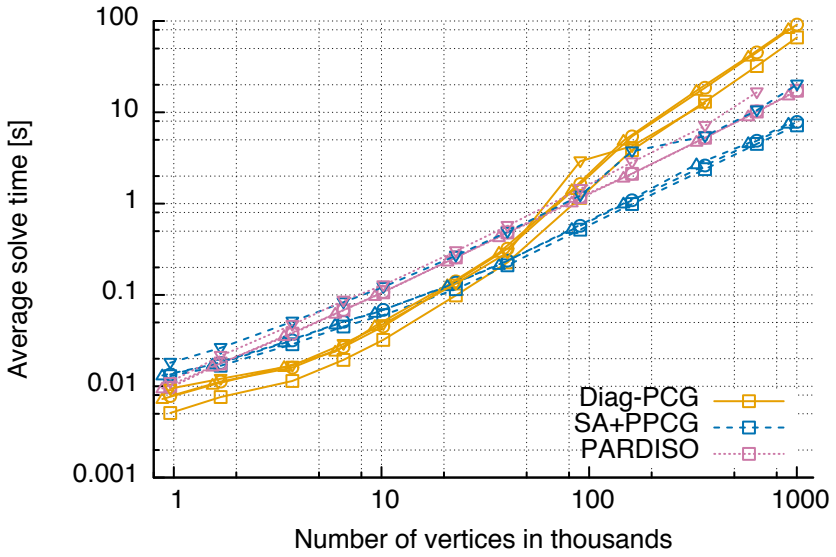


Figure 11.7: *Average time for one linear solve using Diag-PCG (orange), SA+PPCG (blue), and PARDISO (pink). The graphs are shown for four examples : pinned (square markers), drooping (circle markers), re-entrant (diamond markers), and verticalDrop (triangle markers)1.*

11.8.3 Time to solution

Ultimately, the most important metric is usually time to solution. However, making fair comparisons is not entirely straightforward because different solvers have different strengths. As an example, Diag-PCG is generally attractive for systems that are highly diagonally dominant or when the required accuracy is very modest. For cloth simulations, diagonally dominant systems generally occur with small timesteps, soft materials, and/or large masses. At the other end of the spectrum, we compare our results to those of Intel MKL's highly optimized version of PARDISO. As a sparse direct solver, it is attractive if very high accuracy is required or if the problem is very ill-conditioned.

For our comparisons, we use a moderate accuracy (relative error less than 10^{-5}). We note that our results do look better with stiffer materials and/or larger time steps, but do not report on that here.

The results for four of the examples are shown in Fig. 11.7. As can be seen, Diag-PCG is consistently best for small problems, but our method is superior for problems with roughly 25k vertices or more. Somewhat surprisingly, PARDISO shows very good scaling behavior considering that it is a direct solver, but it remains about twice as slow as SA+PPCG. The empirical

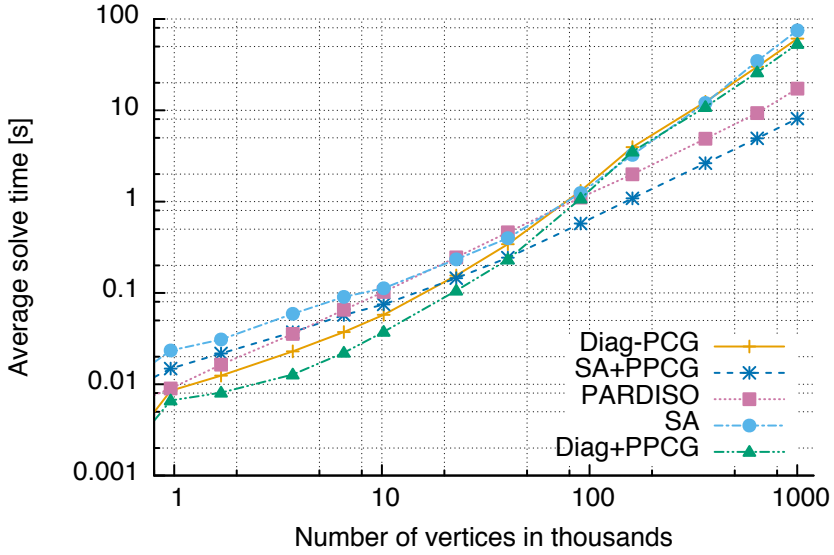


Figure 11.8: *Average time for one linear solve in our horizontal drop example. Note that the fastest time is always obtained using prefiltering.*

complexity of our method can be seen to be close to linear in the size of the problem as expected.

The outlier in the above set of results is the dropVertical example. As already seen in Fig. 11.5, the convergence rate for this problem is worse than for the others, as is the time to solution. The reason for the different behavior in this example is the large number of cloth repulsion springs added to handle cloth-cloth collisions. While our method handles this without any special cases, they do not stem from an underlying and well-behaved PDE, so we lose some optimality. Most likely due to the fact that the near kernel based on rigid body modes is not as good an approximation of the real near kernel as it is in the other examples. However, we still see the same basic scaling behavior as the problem size increases, and the method remains superior to Diag-PCG for large problems.

In Fig. 11.8, we consider the solve time for our horizontal drop example, including now two additional solvers. The first is smoothed aggregation *without* our prefiltering method. The second is Diag-PCG with prefiltering added. For low resolutions, we see that prefiltering further improves the superior performance of Diag-PCG and, for all resolutions, we see that prefiltering is essential for the good performance of SA. In fact, without prefiltering, SA at large resolutions is no better than Diag-PCG and may in fact be worse. However, with prefiltering, SA+PPCG continues to be the best solution for large problems. The reason prefiltering turns out to be so critical for SA is that SA is a very good preconditioner, so a single step without any

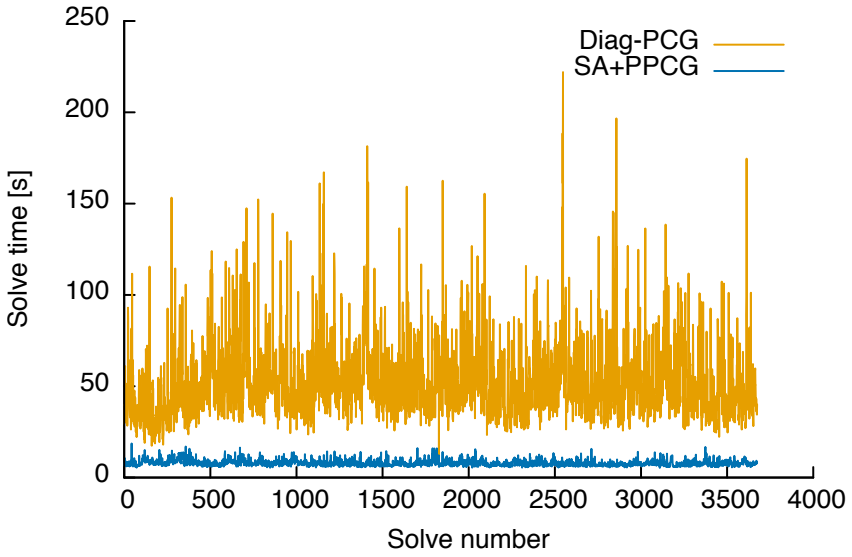


Figure 11.9: *Time for each linear solve in our production example using a walk-cycle animation.*

knowledge about the constraints can bring the solver far from the constraint manifold, and the subsequent projection step is likely to undo much of the progress just made by the solver.

Finally, we present a production example in Fig. 11.9. Overall, we observe an average 8× speedup for a walk-cycle animation and a 6× speedup for a run-cycle. However, as seen from the figure, the variation in the solve time from frame to frame was substantially less with our method than with Diag-PCG.

The speedups for all our examples are summarized in Table 11.1.

11.9 Limitations

One limitation of SA compared to Diag-PCG is its somewhat larger memory overhead. However, the overhead is generally less than 50 percent. SA+PPCG also comes with a higher coding complexity than a simple Diag-PCG, and an unoptimized implementation may often be slower.

The current algorithm is limited by memory bandwidth and, in practice, we saw negligible speedups from parallelization after eight cores. But it remains superior to Diag-PCG and, for our problems, was comparable to the parallel efficiency of MKL's PARDISO. Even so, we believe there is room for improvement, either by exploring techniques such as those presented in [Bell et al., 2012] for forming aggregates in parallel or by algorithmic changes

	Num. vertices	Pinned		Drooping		Re-entrant		DropHorizontal		DropVertical	
Regular tessellation	961	0.39	0.76	0.58	0.72	0.55	0.72	0.57	0.61	0.53	0.61
	1681	0.45	1.06	0.63	0.99	0.60	0.95	0.57	0.75	0.51	0.80
	3721	0.40	1.31	0.50	1.18	0.54	1.17	0.62	0.96	0.35	0.94
	6561	0.43	1.51	0.53	1.35	0.50	1.31	0.65	1.14	0.35	1.04
	10201	0.54	1.78	0.66	1.57	0.72	1.53	0.77	1.37	0.40	1.06
	22801	0.85	2.23	1.03	1.89	1.02	1.86	1.07	1.69	0.50	1.14
	40401	1.07	2.29	1.35	2.03	1.36	2.01	1.40	1.89	0.60	1.18
	90601	2.23	2.27	2.87	2.07	2.67	1.95	2.27	1.92	2.35	1.11
	160801	3.89	2.14	5.01	1.95	4.87	1.92	3.64	1.84	1.14	0.77
	361201	5.48	2.21	7.10	2.01	6.38	1.76	4.70	1.85	2.29	1.32
	641601	7.12	2.26	9.28	2.10	8.84	1.96	6.09	1.89	3.46	1.53
	1002001	9.20	2.40	11.64	2.20	10.89	2.04	7.53	2.13	3.01	1.80
Irregular tessellation	961	0.43	0.87	0.49	0.80	0.51	0.80	0.52	0.64	0.56	0.71
	1681	0.46	1.17	0.52	1.12	0.62	1.04	0.60	0.94	0.54	1.01
	3721	0.43	1.61	0.46	1.45	0.55	1.39	0.56	1.21	0.38	1.06
	6561	0.46	1.86	0.50	1.65	0.56	1.72	0.69	1.44	0.44	1.32
	10201	0.56	2.06	0.61	1.87	0.67	1.93	0.89	1.74	0.53	1.42
	22801	0.81	2.49	0.87	2.11	0.97	2.20	1.13	2.13	0.57	1.43
	40401	1.22	2.61	1.19	2.21	1.43	2.32	1.33	2.23	0.76	1.56
	90601	2.32	2.54	2.51	2.21	2.36	2.33	2.20	2.19	1.29	1.39
	160801	3.78	2.45	4.20	2.18	4.22	2.16	3.26	2.10	2.03	1.39
	361201	5.32	2.49	5.98	2.22	6.13	2.21	4.63	2.13	2.63	1.57
	641601	7.02	2.56	7.84	2.24	7.72	2.19	5.93	2.21	2.39	1.68
	1002001	8.97	2.73	9.82	2.56	9.59	2.47	5.74	2.08	2.81	1.71

Table 11.1: *The speedup factors of SA+PPCG relative to Diag-PCG (first column for each example) and PARDISO (second column for each example). Shaded cells indicate speedups greater than* 1*. Diag-PCG is on average* 11% *faster with the irregular tessellation, while PARDISO on average is* 7% *slower and SA+PPCG is* 8% *faster.*

with better parallel implications. Incremental setup along the lines of what Hecht et al. [2012] did for Cholesky factorizations might be another way to reduce the performance hit of time spent in setup.

While the results show near optimal behavior, they are not perfect. The convergence rates degrade (slowly) as the problem sizes grow, leading to more iterations while cycling, so time to solution is not quite scaling linearly. To attain ideal convergence and subsequently linear scaling, further improvements should be made to the construction of the kernel components to ensure that they match the discretization at all times. Also, a more accurate discretization of the underlying PDEs than that provided by Baraff and Witkin [1998] might offer a better foundation for building multigrid methods. We intend to investigate this in the future.

Finally, we note that in our evaluation here we have only considered linear

problems, but a natural extension is to use the proposed solver in the inner loop of an outer nonlinear iteration that involves a linearization strategy such as Newton's method.

11.10 Summary

This chapter presented a new preconditioner for linear systems formed in implicit cloth simulations by developing an algebraic multigrid hierarchy based on the underlying PDEs and discretization. The SA solver provides a faster solution than existing methods for typical problems with $25,000$ vertices or more. For problems that are stiffer, have smaller mass, or are subjected to larger time steps, the advantages of the method increase and show speedups for smaller problems as well.

To realize the full potential of the SA in cloth simulation and attain near optimal scaling in the presence of collisions, it has to be paired with our new PPCG method. SA+PPCG is attractive because no changes need to be made to existing collision detection and response methods to handle the linear solves.

CHAPTER 12

Conclusion

Large scale simulation of cloth and hair with contact is a challenging problem for multiple reasons. Not only are these simulations characterized by large strain deformations, but the contacts add substantial complications due to their nonlinear, nonsmooth, and nonconvex nature.

In this thesis we started by establishing an orthotropic material model for cloth which is suitable for large deformations in terms of both bending and stretching. Even though cloth is embedded in 3D, the material space is only two dimensional, so special care has to be taken to identify the correct symmetry groups and invariants for the constitutive model. From the fundamental invariants we have established the basic relationships between material parameters like those in Eq. (4.2) and Eq. (4.4) and noted that they are different in 2D compared to results typically presented for 3D.

The resulting cloth model can be used at little to no extra cost compared to existing models. In fact in Section 4.5.2 we showed that the additional cost of the orthotropic membrane model compared to an isotropic model boils down to the addition of two extra 3×3 matrices per element. Similarly, in Chapter 5 we showed a substantial speedup for the evaluation of the exact Hessians of the bending energy, and we also noted that this exact evaluation is actually faster than some existing approximate approaches.

An important insight presented here is the fact that cloth simulations are inherently ill-conditioned for sufficiently large time steps; even in the absence of any constraints. This is not due to the fact that we are dealing with thin

shells, but simply because the elastic energy and forces are translationally invariant. For objects that are pinned down through boundary conditions this problem goes away, but cloth is typically only held in place through collision constraints. Thus, most cloth problems will suffer from this ill-conditioning.

Other authors have observed that it can be beneficial to use a variational formulation based on the incremental potential for computing the dynamics of a system. We have confirmed that the speedup from this approach can be quite substantial, and we have made the additional observation that it tends to be more robust in face of ill-conditioning.

Proper handling of frictional contact is notoriously difficult. In Chapter 7 we made the essential observation that for thin objects like cloth and hair an additional complication is that it is critical to take the nonlinearity of the materials into account when handling contact. We used that in Chapter 8 to construct a new solver, ADONIS, for stably simulating frictional contact in hair and rod assemblies. By using this solver we showed in Chapter 9 that accommodating the nonlinearity in many cases can be done with almost no overhead. In other cases it requires taking multiple (sometimes even many) Newton iterations. However, doing so allows the simulations to run at time steps that are orders of magnitude larger than what is possible with a simple linear treatment. In the end the cost incurred by the extra nonlinear iterations was therefore marginal compared to the savings realized by taking much larger time steps. More importantly, this allowed us to run simulations with unprecedented complexity, which highlights the importance of capturing the frictional contact phenomena correctly.

As noted at the beginning of the thesis, the extension of these results from hair to cloth is nontrivial. In fact it seems unlikely that one can easily find a way to decouple the problem in the same way it is possible for hair. Since we may therefore be forced to deal with large coupled systems, we have focused on developing a multigrid technique suitable for cloth simulation. In doing so we revisited some of fundamental ideas behind multigrid in Chapter 10. In Chapter 11 we then observed that the equations for elasticity in general leads to vector valued problems with strongly coupled variables. Furthermore we observed that the problems are highly anisotropic with both spatially and temporally varying directions of anisotropy. As a consequence we concluded that neither standard geometric nor algebraic multigrid methods are likely to perform well for this problem. This is important as most existing attempts in graphics at using multigrid for cloth have focused on geometrically inspired methods.

To come up with an efficient multigrid method for cloth problems we applied the smoothed aggregation technique. We then went on to show its

efficacy as a preconditioner for conjugate gradient when applied within a production quality cloth solver. There are many factors that influence the exact speedup factors, but for a representative set of examples we saw up to an order of magnitude speedup compared to a block diagonally preconditioned conjugate gradient method.

Finally we noted, that in order to realize the full potential of the SA in cloth simulations and attain near optimal scaling in the presence of collisions, the preconditioner must be applied to the null space of the constraints. As a last contribution we therefore introduced a new prefiltered preconditioned conjugate gradient (PPCG) method. While the PPCG method is essential for SA it is also beneficial for simple preconditioners and can easily be incorporated into existing solvers without adding the complexity of the smoothed aggregation method.

The design and application of a truly robust method for handling non-convex and ill-conditioned problems remains future work, but the insights gained here constitute an important step in the right direction. As illustrated by our results, this direction allows for more accurate models, and visually more rich imagery by capturing features like clumping in hair due to frictional contact and fine wrinkles in clothing. We therefore hope that this will inspire future work to extend the nonlinearly compliant collision response to cloth and develop multigrid methods capable of solving large QPs with inequality constraints.

APPENDIX A

Multigrid preconditioners

Multigrid methods are often used as preconditioners for the conjugate gradient algorithm. However, to guarantee convergence all such preconditioners must behave as SPD matrices. Here we therefore show that the approximate inverse of $\boldsymbol{A}^h$ associated with a multigrid cycle is SPD under two assumptions. These assumptions are important because the conjugate gradient algorithm not only can fail if they are not fullfilled, but in practice often will.

The first assumption is that the cycling scheme is *symmetric in energy* in the sense that the relaxation scheme used in the descent through the coarser grids and the relaxation scheme used in the ascent back through the finer grids are stationary linear iterative methods with error propagation matrices that are energy adjoints of each other. We denote the respective error propagation matrices by $(\boldsymbol{G}^h)^*$ and $\boldsymbol{G}^h$. The second assumption is that relaxation by itself is *convergent in energy*. i.e., that $\|\boldsymbol{G}^h\|_{\boldsymbol{A}^h} < 1$.

Energy symmetry can easily be shown to hold whenever $\boldsymbol{G}^h$ can be written as a polynomial in $\boldsymbol{N}^h = \boldsymbol{M}^h\boldsymbol{A}^h$, where $\boldsymbol{M}^h$ is any symmetric grid h matrix. To see this, let ν be a positive integer and note that the energy adjoint of

$(\boldsymbol{N}^h)^\nu$ is

$$\begin{aligned}
\left((\boldsymbol{N}^h)^\nu\right)^* &= (\boldsymbol{A}^h)^{-1}\left((\boldsymbol{N}^h)^\nu\right)^T \boldsymbol{A}^h \\
&= (\boldsymbol{A}^h)^{-1}\left((\boldsymbol{M}^h\boldsymbol{A}^h)^\nu\right)^T \boldsymbol{A}^h \\
&= (\boldsymbol{A}^h)^{-1}\left((\boldsymbol{M}^h\boldsymbol{A}^h)^T\right)^\nu \boldsymbol{A}^h \\
&= (\boldsymbol{A}^h)^{-1}(\boldsymbol{A}^h\boldsymbol{M}^h)^\nu \boldsymbol{A}^h \\
&= (\boldsymbol{A}^h)^{-1}\boldsymbol{A}^h\boldsymbol{M}^h(\boldsymbol{A}^h\boldsymbol{M}^h)^{\nu-1}\boldsymbol{A}^h \\
&= \boldsymbol{M}^h(\boldsymbol{A}^h\boldsymbol{M}^h)^{\nu-1}\boldsymbol{A}^h \\
&= (\boldsymbol{M}^h\boldsymbol{A}^h)^\nu \\
&= (\boldsymbol{N}^h)^\nu.
\end{aligned}$$

It follows that many well-known relaxation schemes are energy symmetric, including Richardson with $\boldsymbol{M} = \omega\boldsymbol{I}$, damped Jacobi with $\boldsymbol{M} = \omega\boldsymbol{D}^{-1}$, and Chebyshev with any symmetric $\boldsymbol{M}$. To ensure convergence, the step size parameter, ω, for Richardson must be in $(0, 2/\|\boldsymbol{A}^h\|)$, while the damping factor, ω, for Jacobi must be in $(0, 2/\|(\boldsymbol{D}^h)^{-1}\boldsymbol{A}^h\|)$. The parameters for Chebyshev iterations must be chosen based on bounds for the largest and smallest eigenvalues of $\boldsymbol{M}^h\boldsymbol{A}^h$.

The theory also applies to Gauss-Seidel provided the orderings used in the descent and ascent phases are the reverses of each other. The well-known energy-convergence property of Gauss-Seidel can be easily seen by noting that the k^{th} step within an iteration minimizes the energy functional (and, hence, the energy norm of the error) in the direction of the k^{th} coordinate.

In the proof below, we are assuming a $V(1,1)$-cycle, with the coarsest-grid equations treated either by an exact solver or one iteration of the energy-symmetric smoother (i. e., the dual iteration associated with error propagation matrix $\boldsymbol{G}^h(\boldsymbol{G}^h)^*$). However, the extension of these results to a general $V(\nu,\nu)$-cycle can be made simply by replacing $\boldsymbol{G}^h$ by $(\boldsymbol{G}^h)^\nu$. It should also be clear that the theory applies to W-cycles as well as any other cycling scheme that visits the levels in a symmetric way. By this we mean that the sequence of levels that defines the cycle's schedule is the same if the order is reversed. The result excludes full multigrid cycles, although conjugate gradients can be used for the inner V-cycles used on each level of the full multigrid cycle. In any case, full multigrid is generally not meant as an iterative solver, but rather a direct one that aims to obtain discretization-error accuracy in one pass.

Theorem 2. *(Preconditioning)* An energy-symmetric multigrid cycle based on an energy-convergent relaxation scheme is a stationary linear iterative

method with error propagation matrix of the form $\boldsymbol{MV}^h = \boldsymbol{I} - \boldsymbol{B}^h\boldsymbol{A}^h$, where $\boldsymbol{B}^h$ is SPD.

Proof. The proof uses induction on the number of V-cycle levels. We begin by assuming that there is only one level, so that level h is both the finest and the coarsest grid. For the case that the coarsest grid is treated by an exact solver, then $\boldsymbol{MV}^h = \boldsymbol{0}$ and, thus, $\boldsymbol{B}^h = (\boldsymbol{A}^h)^{-1}$, which is SPD. The case that the coarsest grid is treated by the energy-symmetric smoother requires more effort. First note that $(\boldsymbol{G}^h)^*(\boldsymbol{A}^h)^{-1} = (\boldsymbol{A}^h)^{-1}(\boldsymbol{G}^h)^T$. Solving for $\boldsymbol{B}^h$ in $\boldsymbol{MV}^h = \boldsymbol{I} - \boldsymbol{B}^h\boldsymbol{A}^h$ thus yields

$$\boldsymbol{B}^h = (\boldsymbol{A}^h)^{-1} - \boldsymbol{G}^h(\boldsymbol{G}^h)^*(\boldsymbol{A}^h)^{-1} = (\boldsymbol{A}^h)^{-1} - \boldsymbol{G}^h(\boldsymbol{A}^h)^{-1}(\boldsymbol{G}^h)^T.$$

This clearly establishes symmetry, but it is also establishes positive definiteness. To see this, first note that

$$\langle \boldsymbol{v}^h, \boldsymbol{v}^h \rangle = \langle \boldsymbol{A}^h(\boldsymbol{A}^h)^{-1}\boldsymbol{v}^h, \boldsymbol{v}^h \rangle \leq \|\boldsymbol{A}^h\| \langle (\boldsymbol{A}^h)^{-1}\boldsymbol{v}^h, \boldsymbol{v}^h \rangle$$

for any vector $\boldsymbol{v}^h$. Note also that $\|\boldsymbol{C}\boldsymbol{C}^T\| = \|\boldsymbol{C}^T\|^2 = \|\boldsymbol{C}\|^2$ for any square matrix $\boldsymbol{C}$. Note finally that

$$\|\boldsymbol{G}^h\|_{\boldsymbol{A}^h} = \max_{\boldsymbol{v}^h \neq \boldsymbol{0}} \frac{\|\boldsymbol{G}^h\boldsymbol{v}^h\|_{\boldsymbol{A}^h}}{\|\boldsymbol{v}^h\|_{\boldsymbol{A}^h}} = \max_{\boldsymbol{v}^h \neq \boldsymbol{0}} \frac{\|(\boldsymbol{A}^h)^{\frac{1}{2}}\boldsymbol{G}^h(\boldsymbol{A}^h)^{-\frac{1}{2}}(\boldsymbol{A}^h)^{\frac{1}{2}}\boldsymbol{v}^h\|}{\|(\boldsymbol{A}^h)^{\frac{1}{2}}\boldsymbol{v}^h\|} = \|\boldsymbol{G}_A^h\|,$$

where $\boldsymbol{G}_A^h \equiv (\boldsymbol{A}^h)^{\frac{1}{2}}\boldsymbol{G}^h(\boldsymbol{A}^h)^{-\frac{1}{2}}$. These observations lead to the following bound:

$$\begin{aligned}
\left\langle \left((\boldsymbol{A}^h)^{-1} - \boldsymbol{G}^h(\boldsymbol{A}^h)^{-1}(\boldsymbol{G}^h)^T \right) \boldsymbol{u}^h, \boldsymbol{u}^h \right\rangle = \\
\left\langle (\boldsymbol{A}^h)^{-\frac{1}{2}}(\boldsymbol{I} - (\boldsymbol{A}^h)^{\frac{1}{2}}\boldsymbol{G}^h(\boldsymbol{A}^h)^{-1}(\boldsymbol{G}^h)^T(\boldsymbol{A}^h)^{\frac{1}{2}})(\boldsymbol{A}^h)^{-\frac{1}{2}}\boldsymbol{u}^h, \boldsymbol{u}^h \right\rangle = \\
\left\langle \left(\boldsymbol{I} - (\boldsymbol{A}^h)^{\frac{1}{2}}\boldsymbol{G}^h(\boldsymbol{A}^h)^{-1}(\boldsymbol{G}^h)^T(\boldsymbol{A}^h)^{\frac{1}{2}} \right) (\boldsymbol{A}^h)^{-\frac{1}{2}}\boldsymbol{u}^h, (\boldsymbol{A}^h)^{-\frac{1}{2}}\boldsymbol{u}^h \right\rangle \geq \\
(1 - \|\boldsymbol{G}_A^h(\boldsymbol{G}_A^h)^T\|) \langle (\boldsymbol{A}^h)^{-\frac{1}{2}}\boldsymbol{u}^h, (\boldsymbol{A}^h)^{-\frac{1}{2}}\boldsymbol{u}^h \rangle \geq \gamma \langle \boldsymbol{u}^h, \boldsymbol{u}^h \rangle
\end{aligned}$$

for any vector $\boldsymbol{u}^h$, where

$$\gamma \equiv \frac{1 - \|\boldsymbol{G}_A^h\|^2}{\|\boldsymbol{A}^h\|} > 0.$$

This establishes the lemma for the case that the V-cycle consists of one level. We continue the induction proof to the case of two or more V-cycle levels. Our aim is to show that $\boldsymbol{B}^h$ is SPD when $\boldsymbol{B}^{2h} = (\boldsymbol{A}^{2h})^{-1} - \boldsymbol{MV}^{2h}(\boldsymbol{A}^{2h})^{-1}$ is. We first note that the proof of Lemma 4.1 in McCormick [1984] develops a

recursive expression for a $V(1,1)$-cycle error propagation matrix that, when modified for our purposes, becomes

$$\boldsymbol{MV}^h = \boldsymbol{G}^h \left(\boldsymbol{I}_{2h}^h \boldsymbol{MV}^{2h} (\boldsymbol{A}^{2h})^{-1} \boldsymbol{I}_h^{2h} \boldsymbol{A}^h + \boldsymbol{T}^h \right) (\boldsymbol{G}^h)^*.$$

Remembering again that $(\boldsymbol{G}^h)^*(\boldsymbol{A}^h)^{-1} = (\boldsymbol{A}^h)^{-1}(\boldsymbol{G}^h)^T$, we thus have that

$$\begin{aligned}
\boldsymbol{B}^h &= (\boldsymbol{A}^h)^{-1} - \boldsymbol{G}^h \left(\boldsymbol{I}_{2h}^h \boldsymbol{MV}^{2h} (\boldsymbol{A}^{2h})^{-1} \boldsymbol{I}_h^{2h} \boldsymbol{A}^h + \boldsymbol{T}^h \right) (\boldsymbol{G}^h)^* (\boldsymbol{A}^h)^{-1} \\
&= (\boldsymbol{A}^h)^{-1} - \boldsymbol{G}^h \left(\boldsymbol{I}_{2h}^h \boldsymbol{MV}^{2h} (\boldsymbol{A}^{2h})^{-1} \boldsymbol{I}_h^{2h} + \boldsymbol{T}^h (\boldsymbol{A}^h)^{-1} \right) \boldsymbol{G}^{h\,T} \\
&= (\boldsymbol{A}^h)^{-1} - \boldsymbol{G}^h \left(\boldsymbol{I}_{2h}^h \left((\boldsymbol{A}^{2h})^{-1} - \boldsymbol{B}^{2h} \right) \boldsymbol{I}_h^{2h} + \boldsymbol{T}^h (\boldsymbol{A}^h)^{-1} \right) (\boldsymbol{G}^h)^T \\
&= \boldsymbol{G}^h \boldsymbol{I}_{2h}^h \boldsymbol{B}^{2h} \boldsymbol{I}_h^{2h} (\boldsymbol{G}^h)^T + (\boldsymbol{A}^h)^{-1} - \boldsymbol{G}^h \left(\boldsymbol{I}_{2h}^h (\boldsymbol{A}^{2h})^{-1} \boldsymbol{I}_h^{2h} + \boldsymbol{T}^h (\boldsymbol{A}^h)^{-1} \right) (\boldsymbol{G}^h)^T.
\end{aligned}$$

Since $\boldsymbol{I}_h^{2h} = (\boldsymbol{I}_{2h}^h)^T$, then the first term is clearly symmetric. But it is also nonnegative definite because

$$\langle \boldsymbol{G}^h \boldsymbol{I}_{2h}^h \boldsymbol{B}^{2h} \boldsymbol{I}_h^{2h} (\boldsymbol{G}^h)^T \boldsymbol{u}^h, \boldsymbol{u}^h \rangle = \langle \boldsymbol{B}^{2h} \boldsymbol{I}_h^{2h} (\boldsymbol{G}^h)^T \boldsymbol{u}^h, \boldsymbol{I}_h^{2h} (\boldsymbol{G}^h)^T \boldsymbol{u}^h \rangle \geq 0$$

for any vector $\boldsymbol{u}^h$. The remaining terms are what is obtained by setting $\boldsymbol{B}^{2h} = \boldsymbol{0}$, that is, when no coarsening is used, so they must be SPD. To be more precise, first remember that $\boldsymbol{T}^h = \boldsymbol{I} - \boldsymbol{I}_{2h}^h (\boldsymbol{A}^{2h})^{-1} \boldsymbol{I}_h^{2h} \boldsymbol{A}^h$, so

$$\begin{aligned}
&\boldsymbol{I}_{2h}^h (\boldsymbol{A}^{2h})^{-1} \boldsymbol{I}_h^{2h} + \boldsymbol{T}^h (\boldsymbol{A}^h)^{-1} = \\
&\qquad \left(\boldsymbol{I}_{2h}^h (\boldsymbol{A}^{2h})^{-1} \boldsymbol{I}_h^{2h} \boldsymbol{A}^h + \boldsymbol{I} - \boldsymbol{I}_{2h}^h (\boldsymbol{A}^{2h})^{-1} \boldsymbol{I}_h^{2h} \boldsymbol{A}^h \right) (\boldsymbol{A}^h)^{-1} = (\boldsymbol{A}^h)^{-1}.
\end{aligned}$$

We thus have that

$$(\boldsymbol{A}^h)^{-1} - \boldsymbol{G}^h \left(\boldsymbol{I}_{2h}^h (\boldsymbol{A}^{2h})^{-1} \boldsymbol{I}_h^{2h} + \boldsymbol{T}^h (\boldsymbol{A}^h)^{-1} \right) (\boldsymbol{G}^h)^T = (\boldsymbol{A}^h)^{-1} - \boldsymbol{G}^h (\boldsymbol{A}^h)^{-1} (\boldsymbol{G}^h)^T.$$

This is the same operator that we obtained above for the energy-symmetric smoother applied to the grid h equations, so it too is symmetric and we can conclude similarly that

$$\left\langle \left((\boldsymbol{A}^h)^{-1} - \boldsymbol{G}^h (\boldsymbol{A}^h)^{-1} (\boldsymbol{G}^h)^T \right) \boldsymbol{u}^h, \boldsymbol{u}^h \right\rangle \geq \gamma \langle \boldsymbol{u}^h, \boldsymbol{u}^h \rangle > 0.$$

We have thus shown that $\boldsymbol{B}^h$ is the sum of symmetric nonnegative and SPD terms, so it must itself be SPD, thus proving the lemma. □

References

[Adams et al., 2003] Mark Adams, Marian Brezina, Jonathan Hu, and Ray Tuminaro. **Parallel Multigrid Smoothing: Polynomial Versus Gauss–Seidel**. *J. Comput. Phys.*, 188(2):593–610, July 2003. ISSN 0021-9991. doi:10.1016/S0021-9991(03)00194-3.

[Ainsley et al., 2012] Samantha Ainsley, Etienne Vouga, Eitan Grinspun, and Rasmus Tamstorf. **Speculative parallel asynchronous contact mechanics**. *ACM Trans. Graph.*, 31(6):151:1–151:8, November 2012. ISSN 0730-0301. doi:10.1145/2366145.2366170.

[Alart and Curnier, 1991] P. Alart and A. Curnier. **A mixed formulation for frictional contact problems prone to Newton like solution methods**. *Computer Methods in Applied Mechanics and Engineering*, 92(3):353–375, November 1991. ISSN 0045-7825. doi:10.1016/0045-7825(91)90022-X.

[Aliaga et al., 2015] Carlos Aliaga, Carol O'Sullivan, Diego Gutierrez, and Rasmus Tamstorf. **Sackcloth or Silk ? The Impact of Appearance vs Dynamics on the Perception of Animated Cloth**. In *ACM Symposium on Applied Perception (SAP '15)*, September 2015. doi:10.1145/2804408.2804412.

[Allard et al., 2010] Jérémie Allard, François Faure, Hadrien Courtecuisse, Florent Falipou, Christian Duriez, and Paul G. Kry. **Volume Contact Constraints at Arbitrary Resolution**. *ACM Trans. Graph.*, 29(4):82:1–82:10, July 2010. ISSN 0730-0301. doi:10.1145/1778765.1778819.

[An et al., 2007] Heng-Bin An, Ze-Yao Mo, and Xing-Ping Liu. **A choice of forcing terms in inexact Newton method**. *Journal of Computa-*

tional and Applied Mathematics, 200(1):47–60, 2007. ISSN 0377-0427. doi:10.1016/j.cam.2005.12.030.

[Anitescu and Tasora, 2010] Mihai Anitescu and Alessandro Tasora. **An iterative approach for cone complementarity problems for nonsmooth dynamics**. *Computational Optimization and Applications*, 47:207–235, 2010. ISSN 0926-6003. doi:10.1007/s10589-008-9223-4.

[Antman, 2005] Stuart S. Antman. **Nonlinear Problems of Elasticity**. Applied Mathematical Sciences. Springer New York, 2005. ISBN 978-0-387-20880-0. doi:10.1007/0-387-27649-1.

[Ascher and Boxerman, 2003] Uri M. Ascher and Eddy Boxerman. **On the modified conjugate gradient method in cloth simulation**. *The Visual Computer*, 19:526–531, 2003. ISSN 0178-2789. doi:10.1007/s00371-003-0220-4.

[Audoly and Pomeau, 2010] Basile Audoly and Yves Pomeau. **Elasticity and Geometry: From hair curls to the nonlinear response of shells**. Oxford University Press, 2010. ISBN 978-0198506256.

[Başar et al., 2000] Y. Başar, M. Itskov, and A. Eckstein. **Composite laminates: nonlinear interlaminar stress analysis by multi-layer shell elements**. *Computer Methods in Applied Mechanics and Engineering*, 185(2–4): 367 – 397, 2000. ISSN 0045-7825. doi:http://dx.doi.org/10.1016/S0045-7825(99)00267-4.

[Bader and Schnabel, 2007] Brett W. Bader and Robert B. Schnabel. **On the Performance of Tensor Methods for Solving Ill-Conditioned Problems**. *SIAM Journal on Scientific Computing*, 29(6):2329–2351, 2007. doi:10.1137/040607745.

[Baker et al., 2011] Allison H. Baker, Robert D. Falgout, Tzanio V. Kolev, and Ulrike Meier Yang. **Multigrid Smoothers for Ultraparallel Computing**. *SIAM Journal on Scientific Computing*, 33(5):2864–2887, 2011. doi:10.1137/100798806.

[Ball, 1976] John M. Ball. **Convexity conditions and existence theorems in nonlinear elasticity**. *Archive for Rational Mechanics and Analysis*, 63:337–403, 1976. ISSN 0003-9527. doi:10.1007/BF00279992.

[Baraff, 1989] David Baraff. **Analytical Methods for Dynamic Simulation of Non-penetrating Rigid Bodies**. In *Proceedings of the 16th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '89, pages 223–232, New York, NY, USA, 1989. ACM. ISBN 0-89791-312-4. doi:10.1145/74333.74356.

[Baraff, 1996] David Baraff. **Linear-time Dynamics Using Lagrange Multipliers**. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '96, pages 137–146, New York, NY, USA, 1996. ACM. ISBN 0-89791-746-4. doi:10.1145/237170.237226.

[Baraff and Witkin, 1998] David Baraff and Andrew Witkin. **Large steps in cloth simulation**. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '98, pages 43–54, New York, NY, USA, 1998. ACM. ISBN 0-89791-999-8. doi:10.1145/280814.280821.

[Batty et al., 2012] Christopher Batty, Andres Uribe, Basile Audoly, and Eitan Grinspun. **Discrete viscous sheets**. *ACM Trans. Graph.*, 31(4):1131–1137, July 2012. ISSN 0730-0301. doi:10.1145/2185520.2185609.

[Bell et al., 2012] Nathan Bell, S. Dalton, and Luke Olson. **Exposing Fine-Grained Parallelism in Algebraic Multigrid Methods**. *SIAM Journal on Scientific Computing*, 34(4):C123–C152, 2012. doi:10.1137/110838844.

[Bergou et al., 2006] Miklós Bergou, Max Wardetzky, David Harmon, Denis Zorin, and Eitan Grinspun. **A Quadratic Bending Model for Inextensible Surfaces**. In *SGP '06: Proceedings of the fourth Eurographics Symposium on Geometry Processing*, pages 227–230. Eurographics Association, 2006. ISBN 30905673-36-3. doi:10.2312/SGP/SGP06/227-230.

[Bergou et al., 2008] Miklós Bergou, Max Wardetzky, Stephen Robinson, Basile Audoly, and Eitan Grinspun. **Discrete Elastic Rods**. *ACM Trans. Graph.*, 27(3):63:1–63:12, August 2008. ISSN 0730-0301. doi:10.1145/1360612.1360662.

[Bergou et al., 2010] Miklós Bergou, Basile Audoly, Etienne Vouga, Max Wardetzky, and Eitan Grinspun. **Discrete viscous threads**. *ACM Trans. Graph.*, 29:116:1–116:10, July 2010. ISSN 0730-0301. doi:10.1145/1778765.1778853.

[Bernoulli, 1789] Jacques Bernoulli. **Essai théoretique sur les vibrations des plaques élastique rectangulaires et libres**. In *Nova acta Academiae scientiarum imperialis Petropolitanae*, volume 5, pages 197–219. Academiae scientiarum, 1789. URL `https://play.google.com/store/books/details/Nova_acta_Academiae_scientiarum_imperialis_Petropo?id=UVs-AAAAcAAJ`.

[Bertails-Descoubes et al., 2011] Florence Bertails-Descoubes, Florent Cadoux, Gilles Daviet, and Vincent Acary. **A Nonsmooth Newton Solver for Capturing Exact Coulomb Friction in Fiber Assemblies**.

ACM Trans. Graph., 30(1):6:1–6:14, February 2011. ISSN 0730-0301. doi:10.1145/1899404.1899410.

[Betsch and Siebert, 2009] Peter Betsch and Ralf Siebert. **Rigid body dynamics in terms of quaternions: Hamiltonian formulation and conserving numerical integration**. *International Journal for Numerical Methods in Engineering*, 79(4):444–473, 2009. ISSN 1097-0207. doi:10.1002/nme.2586.

[Betten, 1982] Josef Betten. **Integrity basis for a second-order and a fourth-order tensor**. *International Journal on Mathematics and Mathematical Sciences*, 5(1):87–96, 1982. doi:10.1155/S0161171282000088.

[Bobenko and Suris, 1999] A. I. Bobenko and Yu. B. Suris. **Discrete Time Lagrangian Mechanics on Lie Groups, with an Application to the Lagrange Top**. *Communications in Mathematical Physics*, 204:147–188, 1999. ISSN 0010-3616. doi:10.1007/s002200050642.

[Böhlke and Bertram, 2002] Thomas Böhlke and Albrecht Bertram. **On the Ellipticity of Finite Isotropic Linear Elastic Laws**. Preprint. Otto von Guericke Universität Magdeburg, Fak. für Maschinenbau, 2002. URL `http://www.uni-magdeburg.de/ifme/l-festigkeit/pdf/1/boehlke-on-the-ellip.pdf`.

[Bonet and Burton, 1998] J. Bonet and A. J. Burton. **A simple orthotropic, transversely isotropic hyperelastic constitutive equation for large strain computations**. *Computational Methods in Applied Mechanical Engineering*, 162(1–4):151–164, 1998. doi:10.1016/S0045-7825(97)00339-3.

[Bonnefon and Daviet, 2011] Olivier Bonnefon and Gilles Daviet. **Quartic formulation of Coulomb 3D frictional contact**. Technical Report RT-0400, INRIA, January 2011. URL `https://hal.inria.fr/inria-00553859/PDF/RT-0400.pdf`.

[Bouaricha, 1997] Ali Bouaricha. **Tensor Methods for Large, Sparse Unconstrained Optimization**. *SIAM Journal on Optimization*, 7(3):732–756, 1997. doi:10.1137/S1052623494267723.

[Boxerman and Ascher, 2004] Eddy Boxerman and Uri Ascher. **Decomposing Cloth**. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '04, pages 153–161, Aire-la-Ville, Switzerland, 2004. Eurographics Association. ISBN 3-905673-14-2. doi:10.1145/1028523.1028543.

[Boyd and Vandenberghe, 2004] Stephen Boyd and Lieven Vandenberghe. **Convex optimization**. Cambridge University Press, 2004. ISBN 978-0-521-83378-3. URL `http://stanford.edu/~boyd/cvxbook/`.

[Bramble et al., 1996] James H Bramble, Richard E Ewing, Joseph E Pasciak, and Jian Shen. **The analysis of multigrid algorithms for cell centered finite difference methods**. *Advances in Computational Mathematics*, 5(1): 15–29, 1996. ISSN 1019-7168. doi:10.1007/BF02124733.

[Brandt et al., 1982] A. Brandt, S. McCormick, and J. Ruge. **Algebraic multigrid (AMG) for automatic multigrid solution with application to geodetic computations**. Technical report, Institute for Computational Studies, Colorado State University, POB 1852, Fort Collins, Colorado, 1982.

[Brandt et al., 1985] A. Brandt, S. F. McCormick, and J. W. Ruge. **Algebraic multigrid (AMG) for sparse matrix equations**. In David John Evans, editor, *Sparsity and its Applications*, pages 257–284. Cambridge University Press, 1985. ISBN 0-521-26272-0.

[Brandt et al., 2011] A. Brandt, J. Brannick, K. Kahl, and I. Livshits. **Bootstrap AMG**. *SIAM J. Sci. Comput.*, 33(2):612–632, March 2011. ISSN 1064-8275. doi:10.1137/090752973.

[Brandt, 1977] Achi Brandt. **Multi-level adaptive solutions to boundary-value problems**. *Mathematics of Computation*, 31(138):333–390, 1977. doi:10.1090/S0025-5718-1977-0431719-X.

[Brandt and Livne, 2011] Achi Brandt and Oren E. Livne. **Multigrid Techniques: 1984 Guide with Applications to Fluid Dynamics, Revised Edition**. Classics in Applied Mathematics. Society for Industrial and Applied Mathematics, 2011. ISBN 978-1-61197-074-6. doi:10.1137/1.9781611970753.

[Brannick et al., 2007] James Brannick, Marian Brezina, David Keyes, Oren Livne, Irene Livshits, Scott MacLachlan, Tom Manteuffel, Steve McCormick, John Ruge, and Ludmil Zikatanov. **Adaptive Smoothed Aggregation in Lattice QCD**. In Olof B. Widlund and David E. Keyes, editors, *Domain Decomposition Methods in Science and Engineering XVI*, volume 55 of *Lecture Notes in Computational Science and Engineering*, pages 505–512. Springer Berlin Heidelberg, 2007. ISBN 978-3-540-34468-1. doi:10.1007/978-3-540-34469-8_63.

[Brezina et al., 2004] M. Brezina, R. Falgout, S. MacLachlan, T. Manteuffel, S. McCormick, and J. Ruge. **Adaptive Smoothed Aggregation (αSA)**. *SIAM Journal on Scientific Computing*, 25(6):1896–1920, 2004. doi:10.1137/S1064827502418598.

[Bridson et al., 2002] Robert Bridson, Ronald Fedkiw, and John Anderson. **Robust Treatment of Collisions, Contact and Friction for Cloth Ani-**

mation. *ACM Trans. Graph.*, 21(3):594–603, July 2002. ISSN 0730-0301. doi:10.1145/566654.566623.

[Bridson et al., 2003] Robert Bridson, Sebastian Marino, and Ronald Fedkiw. **Simulation of clothing with folds and wrinkles**. In *SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 28–36. Eurographics Association, 2003. ISBN 1-58113-659-5.

[Briggs et al., 2000] W. Briggs, V. Henson, and S. McCormick. **A Multigrid Tutorial, Second Edition**. Society for Industrial and Applied Mathematics, 2000. doi:10.1137/1.9780898719505.

[Cartis et al., 2011a] Coralia Cartis, Nicholas I. M. Gould, and Philippe L. Toint. **Adaptive cubic regularisation methods for unconstrained optimization. Part I: motivation, convergence and numerical results**. *Mathematical Programming*, 127(2):245–295, 2011a. ISSN 0025-5610. doi:10.1007/s10107-009-0286-5.

[Cartis et al., 2011b] Coralia Cartis, Nicholas I. M. Gould, and Philippe L. Toint. **Adaptive cubic regularisation methods for unconstrained optimization. Part II: worst-case function- and derivative-evaluation complexity**. *Mathematical Programming*, 130(2):295–319, 2011b. ISSN 0025-5610. doi:10.1007/s10107-009-0337-y.

[Chladni, 1787] Ernst Florens Friedrich Chladni. **Entdeckungen über die Theorie des Klanges**. Weidmanns Erben und Reich, Leipzig, 1787. doi:10.3931/e-rara-4235.

[Ciarlet, 1980] Philippe G. Ciarlet. **A justification of the von Kármán equations**. *Archive for Rational Mechanics and Analysis*, 73(4):349–389, 1980. ISSN 0003-9527. doi:10.1007/BF00247674. URL `http://dx.doi.org/10.1007/BF00247674`.

[Ciarlet, 2000] Philippe G. Ciarlet, editor. **Mathematical Elasticity Volume III: Theory of Shells**, volume 29 of *Studies in Mathematics and Its Applications*. 2000. ISBN 978-0-444-82891-0. URL `http://www.sciencedirect.com/science/bookseries/01682024/29`.

[Cirio et al., 2014] Gabriel Cirio, Jorge Lopez-Moreno, David Miraut, and Miguel A. Otaduy. **Yarn-level Simulation of Woven Cloth**. *ACM Trans. Graph.*, 33(6):207:1–207:11, November 2014. ISSN 0730-0301. doi:10.1145/2661229.2661279.

[Dalton et al., 2015] Steven Dalton, Luke Olsen, and Nathan Bell. **Optimizing Sparse Matrix-Matrix Multiplication for the GPU**. *ACM Transactions on Mathematical Software*, 41(4), 2015.

[Daviet et al., 2011] Gilles Daviet, Florence Bertails-Descoubes, and Laurence Boissieux. **A Hybrid Iterative Solver for Robustly Capturing Coulomb Friction in Hair Dynamics**. *ACM Trans. Graph.*, 30(6):139:1–139:12, December 2011. ISSN 0730-0301. doi:10.1145/2070781.2024173.

[Delassus, 1917] Étienne Delassus. **Mémoire sur la théorie des liasons finies unilatérales**. *Annales scientifiques de l'É.N.S., 3^e série*, 34:95–179, 1917. URL http://www.numdam.org/item?id=ASENS_1917_3_34__95_0.

[Delingette, 2008] Hervé Delingette. **Triangular Springs for Modeling Nonlinear Membranes**. *IEEE Transactions on Visualization and Computer Graphics*, 14(2):329–341, March/April 2008. doi:10.1109/TVCG.2007.70431.

[Dembo et al., 1982] Ron S. Dembo, Stanley C. Eisenstat, and Trond Steihaug. **Inexact Newton Methods**. *SIAM Journal on Numerical Analysis*, 19(2): 400–408, 1982. doi:10.1137/0719025.

[Dennis and Turner, 1987] J.E. Dennis, Jr and Kathryn Turner. **Generalized conjugate directions**. *Linear Algebra and its Applications*, 88–89(0):187–209, 1987. ISSN 0024-3795. doi:10.1016/0024-3795(87)90109-1.

[Drucker, 1957] D. C. Drucker. **A definition of stable inelastic matieral**. Technical Report 2, Division of Engeering, Brown University, September 1957. URL http://www.dtic.mil/dtic/tr/fulltext/u2/143756.pdf.

[Duriez et al., 2004] Christian Duriez, Claude Andriot, and Abderrahmane Kheddar. **Signorini's contact model for deformable objects in haptic simulations**. In *Intelligent Robots and Systems (IROS 2004). Proceedings 2004 IEEE/RSJ International Conference on*, volume 4, pages 3232–3237, Sep 2004. doi:10.1109/IROS.2004.1389915.

[Duriez et al., 2006] Christian Duriez, Frederic Dubois, Abderrahmane Kheddar, and Claude Andriot. **Realistic Haptic Rendering of Interacting Deformable Objects in Virtual Environments**. *IEEE Transactions on Visualization and Computer Graphics*, 12(1):36–47, January 2006. ISSN 1077-2626. doi:10.1109/TVCG.2006.13.

[English and Bridson, 2008] Elliot English and Robert Bridson. **Animating Developable Surfaces Using Nonconforming Elements**. *ACM Trans. Graph.*, 27(3):66:1–66:5, August 2008. ISSN 0730-0301. doi:10.1145/1360612.1360665.

[Ericson, 2004] Christer Ericson. **Real-Time Collision Detection**. The Morgan Kaufmann Series in Interactive 3D Technology. Morgan Kaufmann, December 2004. ISBN 1558607323.

[Etzmuß et al., 2003] O. Etzmuß, M. Keckeisen, and W. Straßer. **A fast finite element solution for cloth modelling**. In *Computer Graphics and Applications, 2003. Proceedings. 11th Pacific Conference on*, pages 244–251, Oct 2003. doi:10.1109/PCCGA.2003.1238266.

[Euler, 1766a] Leonhard Euler. In *Novi commentarii Academiae scientiarum imperialis Petropolitanae*, volume 10, pages 261–281. Academiae scientiarum, 1766a. URL `http://eulerarchive.maa.org/pages/E303.html`.

[Euler, 1766b] Leonhard Euler. In *Novi commentarii Academiae scientiarum imperialis Petropolitanae*, volume 10, pages 243–260. Academiae scientiarum, 1766b. URL `http://eulerarchive.maa.org/pages/E302.html`.

[Euler, 1960] Leonhard Euler. **The rational mechanics of flexible or elastic bodies 1638 - 1788**. Opera mechanica et astronomica. Birkhäuser Basel, 1960. ISBN 978-3-7643-1441-5.

[Faure et al., 2008] François Faure, Sébastien Barbier, Jérémie Allard, and Florent Falipou. **Image-based Collision Detection and Response Between Arbitrary Volume Objects**. In *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '08, pages 155–162, Aire-la-Ville, Switzerland, Switzerland, 2008. Eurographics Association. ISBN 978-3-905674-10-1. URL `http://dl.acm.org/citation.cfm?id=1632592.1632615`.

[Fish et al., 1996] J. Fish, L. Pan, V Belsky, and S. Gomaa. **Unstructured multigrid methods for shells**. *International Journal for Numerical Methods in Engineering*, 39(7):1181–1197, 1996. ISSN 1097-0207. doi:10.1002/(SICI)1097-0207(19960415)39:7<1181::AID-NME899>3.0.CO;2-Y.

[Fröhlich and Botsch, 2011] Stefan Fröhlich and Mario Botsch. **Example-Driven Deformations Based on Discrete Shells**. *Computer Graphics Forum*, 30(8):2246–2257, 2011. ISSN 1467-8659. doi:10.1111/j.1467-8659.2011.01974.x.

[Garg et al., 2007] Akash Garg, Eitan Grinspun, Max Wardetzky, and Denis Zorin. **Cubic shells**. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '07, pages 91–98. Eurographics Association, 2007. ISBN 978-1-59593-624-0.

[Gast and Schroeder, 2014] Theodore F. Gast and Craig Schroeder. **Optimization Integrator for Large Time Steps**. In Vladlen Koltun and Eftychios Sifakis, editors, *Eurographics/ ACM SIGGRAPH Symposium on Computer Animation*. The Eurographics Association, 2014. ISBN 978-3-905674-61-3. doi:10.2312/sca.20141120.

[Gee, 2004] Michael Gee. **Effiziente Lösungsstrategien in der nichtlinearen Schalenmechanik**. PhD thesis, Institut für Baustatik, Universität Stuttgart, 2004.

[Gee et al., 2005] Michael Gee, Ekkehard Ramm, and Wolfgang A. Wall. **Parallel multilevel solution of nonlinear shell structures**. *Computer Methods in Applied Mechanics and Engineering*, 194(21—24):2513–2533, 2005. ISSN 0045-7825. doi:10.1016/j.cma.2004.07.043. Computational Methods for Shells.

[Gee and Tuminaro, 2006] Michael W. Gee and Ray S. Tuminaro. **Nonlinear Algebraic Multigrid for Constrained Solid Mechanics Problems Using Trilinos**. Technical Report SAND2006-2256, Sandia National Laboratories, April 2006.

[Gingold et al., 2004] Yotam Gingold, Adrian Secord, Jefferson Y. Han, Eitan Grinspun, and Denis Zorin. **A Discrete Model for Inelastic Deformation of Thin Shells**. Technical report, Courant Institute of Mathematical Sciences, New York University, Aug 2004. URL `http://www.cs.columbia.edu/cg/pdfs/1162939464-fracture.pdf`.

[Goldenthal et al., 2007] Rony Goldenthal, David Harmon, Raanan Fattal, Michel Bercovier, and Eitan Grinspun. **Efficient Simulation of Inextensible Cloth**. *ACM Trans. Graph.*, 26(3), July 2007. ISSN 0730-0301. doi:10.1145/1276377.1276438.

[Golub and Loan, 1983] Gene H. Golub and Charles F. Van Loan. **Matrix Computations**. The Johns Hopkins University Press, 3rd edition, 1983.

[Golub and Varga, 1961] Gene H. Golub and Richard S. Varga. **Chebyshev semi-iterative methods, successive overrelaxation iterative methods, and second order richardson iterative methods**. *Numerische Mathematik*, 3(1):157–168, 1961. ISSN 0029-599X. doi:10.1007/BF01386014.

[Gould et al., 2001] Nicholas I. M. Gould, Mary E. Hribar, and Jorge Nocedal. **On the Solution of Equality Constrained Quadratic Programming Problems Arising in Optimization**. *SIAM Journal on Scientific Computing*, 23(4):1376–1395, 2001. doi:10.1137/S1064827598345667.

[Gould et al., 2012] N.I.M. Gould, M. Porcelli, and P.L. Toint. **Updating the regularization parameter in the adaptive cubic regularization algorithm**. *Computational Optimization and Applications*, 53(1):1–22, 2012. ISSN 0926-6003. doi:10.1007/s10589-011-9446-7.

[Goyal et al., 1991] Suresh Goyal, Andy Ruina, and Jim Papadopoulos. **Planar sliding with dry friction Part 1. Limit surface and moment function**. *Wear*, 143(2):307–330, 1991.

[Griewank and Osborne, 1983] A. Griewank and M. R. Osborne. **Analysis of Newton's Method at Irregular Singularities**. *SIAM Journal on Numerical Analysis*, 20(4):747–773, 1983. doi:10.1137/0720050.

[Grinspun et al., 2003] Eitan Grinspun, Anil N. Hirani, Mathieu Desbrun, and Peter Schröder. **Discrete shells**. In *SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 62–67. Eurographics Association, 2003. ISBN 1-58113-659-5.

[Hager and Zhang, 2005] William W. Hager and Hongchao Zhang. **A New Conjugate Gradient Method with Guaranteed Descent and an Efficient Line Search**. *SIAM Journal on Optimization*, 16(1):170–192, 2005. doi:10.1137/030601880.

[Hager and Zhang, 2006] William W. Hager and Hongchao Zhang. **Algorithm 851: CG_DESCENT, a Conjugate Gradient Method with Guaranteed Descent**. *ACM Trans. Math. Softw.*, 32(1):113–137, March 2006. ISSN 0098-3500. doi:10.1145/1132973.1132979.

[Hairer and Wanner, 2004] Ernst Hairer and Gerhard Wanner. **Solving Ordinary Differential Equations II: Stiff and Differential-Algebraic Problems**. Springer, second edition, 2004. ISBN 978-3-642-05220-0. doi:10.1007/978-3-642-05221-7.

[Hairer et al., 2004] Ernst Hairer, Gerhard Wanner, and Syvert P. Nørsett. **Solving Ordinary Differential Equations I: Nonstiff Problems**. Springer, second edition, 2004. ISBN 978-3-540-56670-0. doi:10.1007/978-3-540-78862-1.

[Hairer et al., 2006] Ernst Hairer, Gerhard Wanner, and Christian Lubich. **Geometric Numerical Integration : Structure-Preserving Algorithms for Ordinary Differential Equations**. Springer, 2006. ISBN 978-3-540-30663-4. doi:10.1007/3-540-30666-8.

[Harmon, 2010] David Harmon. **Robust, Efficient, and Accurate Contact Algorithms**. PhD thesis, Columbia University, 2010.

[Harmon et al., 2008] David Harmon, Etienne Vouga, Rasmus Tamstorf, and Eitan Grinspun. **Robust Treatment of Simultaneous Collisions**. *ACM Trans. Graph.*, 27(3):23:1–23:4, August 2008. ISSN 0730-0301. doi:10.1145/1360612.1360622.

[Harmon et al., 2009] David Harmon, Etienne Vouga, Breannan Smith, Rasmus Tamstorf, and Eitan Grinspun. **Asynchronous contact mechanics**. In *ACM SIGGRAPH 2009 papers*, SIGGRAPH '09, pages 87:1–87:12, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-726-4. doi:10.1145/1576246.1531393.

[Harmon et al., 2012] David Harmon, Etienne Vouga, Breannan Smith, Rasmus Tamstorf, and Eitan Grinspun. **Asynchronous contact mechanics**. *Commun. ACM*, 55(4):102–109, April 2012. ISSN 0001-0782. doi:10.1145/2133806.2133828.

[Hauth et al., 2003] Michael Hauth, Olaf Etzmuß, and Wolfgang Straßer. **Analysis of numerical methods for the simulation of deformable models**. *The Visual Computer*, 19:581–600, 2003. ISSN 0178-2789. doi:10.1007/s00371-003-0206-2.

[He and Zheng, 1996] Q.-C. He and Quanshui Zheng. **On the Symmetries of 2D Elastic and Hyperelastic Tensors**. *Journal of Elasticity*, 43:203–225, 1996. doi:10.1007/BF00042501.

[Hecht et al., 2012] Florian Hecht, Yeon Jin Lee, Jonathan R. Shewchuk, and James F. O'Brien. **Updated Sparse Cholesky Factors for Corotational Elastodynamics**. *ACM Transactions on Graphics*, 31(5):123:1–13, October 2012. doi:10.1145/2231816.2231821.

[Heidelberger et al., 2003] Bruno Heidelberger, Matthias Teschner, and Markus Gross. **Real-time volumetric intersections of deforming objects**. In Thomas Ertl, editor, *Proc. of Vision, Modeling, and Visualization VMV'03*, page 461—468. Aka GmbH, November 2003. ISBN 3-89838-048-3.

[Hill, 1957] R. Hill. **On uniqueness and stability in the theory of finite elastic strain**. *Journal of the Mechanics and Physics of Solids*, 5(4):229–241, 1957. doi:10.1016/0022-5096(57)90016-9.

[Hill, 1958] R. Hill. **A general theory of uniqueness and stability in elastic-plastic solids**. *Journal of the Mechanics and Physics of Solids*, 6(3): 236–249, 1958. ISSN 0022-5096. doi:http://dx.doi.org/10.1016/0022-5096(58)90029-2.

[Hoffmann, 2000] Tim Hoffmann. **Discrete Curves and Surfaces**. PhD thesis, Fachbereich 3 Mathematik der Technischen Universität Berlin, January 2000.

[Horn and Johnson, 1985] Roger A. Horn and Charles R. Johnson. **Matrix Analysis**. Cambridge University Press, 1985. ISBN 9780511810817. doi:10.1017/CBO9780511810817. Cambridge Books Online.

[Itskov, 2001] Mikhail Itskov. **A generalized orthotropic hyperelastic material model with application to incompressible shells**. *International Journal for Numerical Methods in Engineering*, 50(8):1777–1799, 2001. ISSN 1097-0207. doi:10.1002/nme.86.

[Itskov and Aksel, 2004] Mikhail Itskov and Nuri Aksel. **A class of orthotropic and transversely isotropic hyperelastic constitutive models based on a polyconvex strain energy function**. *International Journal of Solids and Structures*, 41(14):3833–3848, 2004. ISSN 0020-7683. doi:10.1016/j.ijsolstr.2004.02.027.

[Jean and Moreau, 1992] M. Jean and J. J Moreau. **Unilaterality and dry friction in the dynamics of rigid body collections**. In *Proceedings of Contact Mechanics International Symposium*, volume 1, pages 31–48, October 1992.

[Jeon et al., 2013] Inyong Jeon, Kwang-Jin Choi, Tae-Yong Kim, Bong-Ouk Choi, and Hyeong-Seok Ko. **Constrainable Multigrid for Cloth**. *Computer Graphics Forum*, 32(7):31–39, 2013. ISSN 1467-8659. doi:10.1111/cgf.12209.

[Kaldor et al., 2008] Jonathan M. Kaldor, Doug L. James, and Steve Marschner. **Simulating Knitted Cloth at the Yarn Level**. *ACM Trans. Graph.*, 27:65:1–65:9, August 2008. ISSN 0730-0301. doi:10.1145/1360612.1360664.

[Kaufman, 2009] Danny Kaufman. **Coupled principles for computational frictional contact mechanics**. PhD thesis, Rutgers, The State University of New Jersey, May 2009. doi:10.7282/T3VD6ZKG.

[Kaufman et al., 2014] Danny Kaufman, Rasmus Tamstorf, Breannan Smith, Jean-Marie Aubry, and Eitan Grinspun. **Adaptive Nonlinearity for Collisions in Complex Rod Assemblies**. *ACM Trans. Graph*, 33(4), July 2014. doi:10.1145/2601097.2601100.

[Kaufman et al., 2008] Danny M. Kaufman, Shinjiro Sueda, Doug L. James, and Dinesh K. Pai. **Staggered Projections for Frictional Contact in Multibody Systems**. *ACM Trans. Graph.*, 27(5):164:1–164:11, December 2008. ISSN 0730-0301. doi:10.1145/1409060.1409117.

[Kaufmann et al., 2009] Peter Kaufmann, Sebastian Martin, Mario Botsch, and Markus Gross. **Implementation of Discontinuous Galerkin**

Kirchhoff-Love Shells. Technical Report 622, Institute of Visual Computing, ETH Zürich, 2009. URL http://graphics.ethz.ch/~smartin/data/publication_Kau09c.pdf.

[Kharevych et al., 2006] L. Kharevych, Weiwei Yang, Y. Tong, E. Kanso, J. E. Marsden, P. Schröder, and M. Desbrun. **Geometric, variational integrators for computer animation**. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA '06, pages 43–51. Eurographics Association, 2006. ISBN 3-905673-34-7. doi:10.2312/SCA/SCA06/043-051.

[Kikuchi and Oden, 1988] N. Kikuchi and J. T. Oden. **Contact Problems in Elasticity: A Study of Variational Inequalities and Finite Element Methods**, volume 8. Society for Industrial and Applied Mathematics, 1988. doi:10.1137/1.9781611970845.

[King, 2006] Michael J. King. **A Continuum Constitutive Model for the Mechanical Behavior of Woven Fabrics Including Slip and Failure**. PhD thesis, MIT, June 2006. URL http://hdl.handle.net/1721.1/36192.

[Koiter, 1966] Warner T. Koiter. **On the nonlinear theory of thin elastic shells**. *Proceedings of the Koninklijke Nederlandse Academie van Wetenschappen. Series B: Physical Sciences*, B69:1–54, 1966. URL http://www.dwc.knaw.nl.

[Krishnan et al., 2013] Dilip Krishnan, Raanan Fattal, and Richard Szeliski. **Efficient Preconditioning of Laplacian Matrices for Computer Graphics**. *ACM Trans. Graph.*, 32(4):142:1–142:15, July 2013. ISSN 0730-0301. doi:10.1145/2461912.2461992.

[Lee et al., 2010] Yongjoon Lee, Sung-eui Yoon, Seungwoo Oh, Duksu Kim, and Sunghee Choi. **Multi-Resolution Cloth Simulation**. *Computer Graphics Forum*, 29(7):2225–2232, 2010. ISSN 1467-8659. doi:10.1111/j.1467-8659.2010.01811.x.

[Leine and Aeberhard, 2007] R. I. Leine and U. Aeberhard. **The Euler-Maupertuis principle of least action as variational inequality**. *PAMM*, 7(1):4010019–4010020, 2007. ISSN 1617-7061. doi:10.1002/pamm.200700666.

[Liu, 2005] I-Shih Liu. **Further remarks on euclidean objectivity and the principle of material frame-indifference**. *Continuum Mechanics and Thermodynamics*, 17(2):125–133, 2005. ISSN 0935-1175. doi:10.1007/s00161-004-0191-3.

[Marsden and Hughes, 1983] Jerrold E. Marsden and Thomas J. R. Hughes. **Mathematical foundations of elasticity**. Dover Publications, Inc., 1983. ISBN 0-486-67865-2.

[Mazhar et al., 2015] Hammad Mazhar, Toby Heyn, Dan Negrut, and Alessandro Tasora. **Using Nesterov's Method to Accelerate Multibody Dynamics with Friction and Contact**. *ACM Trans. Graph.*, 34(3):32:1–32:14, May 2015. ISSN 0730-0301. doi:10.1145/2735627.

[McAdams et al., 2011] Aleka McAdams, Yongning Zhu, Andrew Selle, Mark Empey, Rasmus Tamstorf, Joseph Teran, and Eftychios Sifakis. **Efficient Elasticity for Character Skinning with Contact and Collisions**. *ACM Trans. Graph.*, 30(4):37:1–37:12, July 2011. ISSN 0730-0301. doi:10.1145/2010324.1964932.

[McCormick, 1984] Stephen McCormick. **Multigrid Methods for Variational Problems: Further Results**. *SIAM Journal on Numerical Analysis*, 21(2): 255–263, 1984. doi:10.1137/0721018.

[Meurant, 2006] Gérard Meurant. **The Lanczos and Conjugate Gradient Algorithms: From Theory to Finite Precision Computations**. Society for Industrial and Applied Mathematics, 2006. ISBN 978-0-89871-616-0. doi:10.1137/1.9780898718140.

[Miguel et al., 2013] Eder Miguel, Rasmus Tamstorf, Derek Bradley, Sara C. Schvartzman, Bernhard Thomaszewski, Bernd Bickel, Wojciech Matusik, Steve Marschner, and Miguel A. Otaduy. **Modeling and Estimation of Internal Friction in Cloth**. *ACM Trans. Graph.*, 32(6):212:1–212:10, November 2013. ISSN 0730-0301. doi:10.1145/2508363.2508389.

[Míka and Vaněk, 1992] Stanislav Míka and Petr Vaněk. **Acceleration of convergence of a two-level algebraic algorithm by aggregation in smoothing process**. *Applications of Mathematics*, 37(5):343–356, 1992. ISSN 0862-7940. URL `https://eudml.org/doc/15720`.

[Mirtich and Canny, 1995] Brian Mirtich and John Canny. **Impulse-based Simulation of Rigid Bodies**. In *Proceedings of the 1995 Symposium on Interactive 3D Graphics*, I3D '95, pages 181–ff., New York, NY, USA, 1995. ACM. ISBN 0-89791-736-7. doi:10.1145/199404.199436.

[Moakher, 2008] Maher Moakher. **Fourth-order cartesian tensors: old and new facts, notions and applications**. *Quarterly Journal of Mechanics & Applied Mathematics*, 61(2):181–203, 2008. doi:10.1093/qjmam/hbm027.

[Morandi, 2007] Patrick J. Morandi. **Symmetry Groups : The Classification of Wallpaper Patterns**. Technical report, Department of Mathematical

Sciences, New Mexico State University, Las Cruces, 2007. URL http://sierra.nmsu.edu/morandi/notes/Wallpaper.pdf.

[Moré and Thuente, 1994] Jorge J. Moré and David J. Thuente. **Line Search Algorithms with Guaranteed Sufficient Decrease**. *ACM Trans. Math. Softw.*, 20(3):286–307, September 1994. ISSN 0098-3500. doi:10.1145/192115.192132.

[Moreau, 1988] J. J. Moreau. **Unilateral contact and dry friction in finite freedom dynamics**. In J.J. Moreau and P.D. Panagiotopoulos, editors, *Nonsmooth Mechanics and Applications*, volume 302 of *International Centre for Mechanical Sciences*, pages 1–82. Springer Vienna, 1988. ISBN 978-3-211-82066-7. doi:10.1007/978-3-7091-2624-0_1.

[Moreau, 1970] Jean Jacques Moreau. **Convexité et frottement**. Technical Report 32, Université de Montréal. Département d'informatique, March 1970.

[Moreau, 2011] Jean Jacques Moreau. **On Unilateral Constraints, Friction and Plasticity**. In Gianfranco Capriz and Guido Stampacchia, editors, *New Variational Techniques in Mathematical Physics*, volume 63 of *C.I.M.E. Summer Schools*, pages 171–322. Springer Berlin Heidelberg, 2011. ISBN 978-3-642-10958-4. doi:10.1007/978-3-642-10960-7_7.

[Moreau, 1999] J.J. Moreau. **Some Basics of Unilateral Dynamics**. In F. Pfeiffer and Ch. Glocker, editors, *IUTAM Symposium on Unilateral Multibody Contacts*, volume 72 of *Solid Mechanics and its Applications*, pages 1–14. Springer Netherlands, 1999. ISBN 978-94-010-5853-7. doi:10.1007/978-94-011-4275-5_1.

[Morrey, 1952] Charles B. Morrey. **Quasi-convexity and the lower semicontinuity of multiple integrals**. *Pacific Journal of Mathematics*, 2(1):25–53, 1952. URL http://projecteuclid.org/euclid.pjm/1103051941.

[Narain et al., 2012] Rahul Narain, Armin Samii, and James F. O'Brien. **Adaptive Anisotropic Remeshing for Cloth Simulation**. *ACM Trans. Graph.*, 31(6):152:1–152:10, November 2012. ISSN 0730-0301. doi:10.1145/2366145.2366171.

[Nash and Sofer, 1990] Stephen G. Nash and Ariela Sofer. **Assessing a search direction within a truncated-newton method**. *Operations Research Letters*, 9(4):219–221, 1990. ISSN 0167-6377. doi:10.1016/0167-6377(90)90065-D.

[Nesterov and Polyak, 2006] Yurii Nesterov and B.T. Polyak. **Cubic regularization of Newton method and its global performance**. *Mathematical*

Programming, 108(1):177–205, 2006. ISSN 0025-5610. doi:10.1007/s10107-006-0706-8.

[Neumann, 1885] Franz Ernst Neumann. **Vorlesungen über die Theorie der Elastizität der festen Körper und des Lichtäthers**. B. G. Teubner-Verlag, Leipzig, 1885.

[Ni et al., 2015] Xiang Ni, Laxmikant Kale, and Rasmus Tamstorf. **Scalable Asynchronous Contact Mechanics using Charm++**. In *IEEE 29th International Parallel and Distributed Processing Symposium (IPDPS 2015)*, May 2015. doi:10.1109/IPDPS.2015.45.

[Nocedal and Wright, 2006] Jorge Nocedal and Stephen J. Wright. **Numerical Optimization**. Springer New York, 2006. ISBN 978-0-387-30303-1. doi:10.1007/978-0-387-40065-5.

[Noll, 1954] Walter Noll. **On the Continuity of the Solid and Fluid States**. PhD thesis, Indiana University, 1954.

[Noll, 2005] Walter Noll. **A Frame-Free Formulation of Elasticity**, 2005. URL `http://www.math.cmu.edu/~wn0g/FFFE.pdf`.

[Ogden, 1984] R. W. Ogden. **Non-linear elastic deformations**. Wiley & Sons, 1984.

[Oh et al., 2008] SeungWoo Oh, Junyong Noh, and Kwangyun Wohn. **A physically faithful multigrid method for fast cloth simulation**. *Computer Animation and Virtual Worlds*, 19(3-4):479–492, 2008. ISSN 1546-427X. doi:10.1002/cav.255.

[Otaduy et al., 2009] Miguel A. Otaduy, Rasmus Tamstorf, Denis Steinemann, and Markus Gross. **Implicit Contact Handling for Deformable Objects**. *Computer Graphics Forum*, 28(2):559–568, 2009. doi:10.1111/j.1467-8659.2009.01396.x.

[Press et al., 2007] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. **Numerical Recipes: The Art of Scientific Computing**. Cambridge University Press, third edition, 2007. ISBN 0521880688. URL `http://www.nr.com`.

[Provot, 1997] Xavier Provot. **Collision and self-collision handling in cloth model dedicated to design garments**. In Daniel Thalmann and Michiel van de Panne, editors, *Computer Animation and Simulation '97*, Eurographics, pages 177–189. Springer Vienna, 1997. ISBN 978-3-211-83048-2. doi:10.1007/978-3-7091-6874-5_13.

[Radovitzky and Ortiz, 1999] R. Radovitzky and M. Ortiz. **Error estimation and adaptive meshing in strongly nonlinear dynamic problems**. *Computer Methods in Applied Mechanics and Engineering*, 172(1-4):203–240, 1999. ISSN 0045-7825. doi:10.1016/S0045-7825(98)00230-8.

[Rivlin, 1992] Ronald S. Rivlin. **Stability of an elastic material**. In P.E. Ricci, editor, *Problemi Attuali dell'Analisi e della Fisica Matematica*, pages 201–206, Rome, Taormina, Italy, 15-17 October, 1992 1992. Dipartimento di Matematica, Universita di Roma "La Sapienza".

[Rivlin, 1997] Ronald S. Rivlin. **Stability of an elastic material**. In Grigori Isaakovich Barenblatt and Daniel D. Joseph, editors, *Collected Papers of R.S. Rivlin*, pages 883–908. Springer New York, 1997. ISBN 978-1-4612-7530-5. doi:10.1007/978-1-4612-2416-7_60.

[Robbins, 2012] Clarence R. Robbins. **Chemical and Physical Behavior of Human Hair**. Springer, fifth edition, 2012.

[Sadeghi et al., 2010] Iman Sadeghi, Heather Pritchett, Henrik Wann Jensen, and Rasmus Tamstorf. **An Artist Friendly Hair Shading System**. *ACM Trans. Graph.*, 29(3):56:1–10, 2010. doi:10.1145/1833349.1778793.

[Schnabel and Chow, 1991] Robert B. Schnabel and Ta-Tung Chow. **Tensor Methods for Unconstrained Optimization Using Second Derivatives**. *SIAM Journal on Optimization*, 1(3):293–315, 1991. doi:10.1137/0801020.

[Schröder, 2010] Jörg Schröder. **Anisotropic polyconvex energies**. In Jörg Schröder and Patrizio Neff, editors, *Poly-, Quasi- and Rank-One Convexity in Applied Mechanics*, volume CISM Courses and Lectures, Vol. 516. Springer, 2010. doi:10.1007/978-3-7091-0174-2.

[Shariff, 1995] M.H.B.M. Shariff. **A constrained conjugate gradient method and the solution of linear equations**. *Computers & Mathematics with Applications*, 30(11):25–37, 1995. ISSN 0898-1221. doi:10.1016/0898-1221(95)00161-Q.

[Smith et al., 2012] Breannan Smith, Danny M. Kaufman, Etienne Vouga, Rasmus Tamstorf, and Eitan Grinspun. **Reflections on simultaneous impact**. *ACM Trans. Graph.*, 31(4):106:1–106:12, July 2012. ISSN 0730-0301. doi:10.1145/2185520.2185602.

[Spillmann and Teschner, 2007] J. Spillmann and M. Teschner. **CoRdE: Cosserat Rod Elements for the Dynamic Simulation of One-dimensional Elastic Objects**. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '07, pages

63–72, Aire-la-Ville, Switzerland, Switzerland, 2007. Eurographics Association. ISBN 978-1-59593-624-0. URL http://dl.acm.org/citation.cfm?id=1272690.1272700.

[Steigmann, 2012] David J. Steigmann. **A well-posed finite-strain model for thin elastic sheets with bending stiffness**. *Mathematics and Mechanics of Solids*, 18(1):103–112, 2012. ISSN 1081-2865. doi:10.1177/1081286512441105.

[Steigmann, 2013] David J. Steigmann. **Koiter's Shell Theory from the Perspective of Three-dimensional Nonlinear Elasticity**. *Journal of Elasticity*, 111(1):91–107, 2013. ISSN 0374-3535. doi:10.1007/s10659-012-9393-2.

[Sterck et al., 2008] H. De Sterck, Thomas A. Manteuffel, Stephen F. McCormick, Quoc Nguyen, and John Ruge. **Multilevel Adaptive Aggregation for Markov Chains, with Application to Web Ranking**. *SIAM Journal on Scientific Computing*, 30(5):2235–2262, 2008. doi:10.1137/070685142.

[Stewart, 2000] David E. Stewart. **Rigid-Body Dynamics with Friction and Impact**. *SIAM Review*, 42(1):3–39, 2000. doi:10.1137/S0036144599360110.

[Stewart, 2001] David E. Stewart. **Finite-dimensional contact mechanics**. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 359(1789):2467–2482, 2001. ISSN 1364-503X. doi:10.1098/rsta.2001.0904.

[Stewart, 2011] David E. Stewart. **Dynamics with Inequalities: Impacts and Hard Constraints**. Society for Industrial and Applied Mathematics, 2011. doi:10.1137/1.9781611970715.

[Tamstorf, 2013] Rasmus Tamstorf. **Derivation of discrete bending forces and their gradients**. Technical report, Walt Disney Animation Studios, Sep 2013. URL http://www.disneyanimation.com/technology/publications/56.

[Tamstorf and Grinspun, 2013] Rasmus Tamstorf and Eitan Grinspun. **Discrete bending forces and their Jacobians**. *Graphical Models*, 75(6):362–370, November 2013. doi:10.1016/j.gmod.2013.07.001.

[Tamstorf et al., 2015] Rasmus Tamstorf, Toby Jones, and Stephen F. McCormick. **Smoothed Aggregation Multigrid for Cloth Simulation**. *ACM Trans. Graph.*, 34(6), November 2015. ISSN 0730-0301. doi:10.1145/2816795.2818081.

[Terzopoulos et al., 1987] Demetri Terzopoulos, John Platt, Alan Barr, and Kurt Fleischer. **Elastically Deformable Models**. In *Proceedings of the 14th*

Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '87, pages 205–214, New York, NY, USA, 1987. ACM. ISBN 0-89791-227-6. doi:10.1145/37401.37427.

[Teschner et al., 2005] M. Teschner, S. Kimmerle, B. Heidelberger, G. Zachmann, L. Raghupathi, A. Fuhrmann, M.-P. Cani, F. Faure, N. Magnenat-Thalmann, W. Strasser, and P. Volino. **Collision Detection for Deformable Objects**. *Computer Graphics Forum*, 24(1):61–81, 2005. ISSN 1467-8659. doi:10.1111/j.1467-8659.2005.00829.x.

[Thomaszewski et al., 2006] Bernhard Thomaszewski, Markus Wacker, and Wolfgang Straßer. **A Consistent Bending Model for Cloth Simulation with Corotational Subdivision Finite Elements**. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '06, pages 107–116, Aire-la-Ville, Switzerland, Switzerland, 2006. Eurographics Association. ISBN 3-905673-34-7. URL `http://dl.acm.org/citation.cfm?id=1218064.1218079`.

[Timoshenko and Woinowsky-Krieger, 1959] Stephen P. Timoshenko and S. Woinowsky-Krieger. **Theory of plates and shells**. McGraw-Hill, 2nd edition, 1959. ISBN 0-07-085820-9.

[Trinkle et al., 1997] J. C. Trinkle, J.-S. Pang, S. Sudarsky, and G. Lo. **On Dynamic Multi-Rigid-Body Contact Problems with Coulomb Friction**. *ZAMM - Journal of Applied Mathematics and Mechanics / Zeitschrift für Angewandte Mathematik und Mechanik*, 77(4):267–279, 1997. ISSN 1521-4001. doi:10.1002/zamm.19970770411.

[Trottenberg et al., 2000] Ulrich Trottenberg, Cornelius W Oosterlee, and Anton Schuller. **Multigrid**. Academic press, 2000.

[Truesdell and Noll, 1965] Clifford Truesdell and Walter Noll. **The Non-Linear Field Theories of Mechanics**. Springer-Verlag, 1965.

[Turner et al., 1956] M. J. Turner, R. W. Clough, H. C. Martin, and L. P. Topp. **Stiffness and deflection analysis of complex structures**. *J. Aeronautical Society*, 23(9):805–824, 1956.

[van Baar et al., 2011] Jeroen van Baar, Steven Poulakos, Wojciech Jarosz, Derek Nowrouzezahrai, Rasmus Tamstorf, and Markus Gross. **Perceptually-Based Compensation of Light Pollution in Display Systems**. In *Symposium on Applied Perception in Graphics and Visualization*, 2011.

[van der Vorst, 1982] H. A. van der Vorst. **A Generalized Lanczos Scheme**. *Mathematics of Computation*, 39(160):pp. 559–561, Oct 1982. ISSN 00255718. doi:10.2307/2007333.

[Vaněk, 1992] Petr Vaněk. **Acceleration of convergence of a two-level algorithm by smoothing transfer operator**. *Applications of Mathematics*, 37: 265–274, 1992. URL `http://dml.cz/dmlcz/104515`.

[Vassilevski, 2008] Panayot S. Vassilevski. **Multilevel Block Factorization Preconditioners, Matrix-based Analysis and Algorithms for Solving Finite Element Equations**. Springer, 2008. doi:10.1137/1.9780898719505.

[Volino et al., 2009] Pascal Volino, Nadia Magnenat-Thalmann, and Francois Faure. **A simple approach to nonlinear tensile stiffness for accurate cloth simulation**. *ACM Trans. Graph.*, 28:105:1–105:16, September 2009. ISSN 0730-0301. doi:10.1145/1559755.1559762.

[Vouga et al., 2011] Etienne Vouga, David Harmon, Rasmus Tamstorf, and Eitan Grinspun. **Asynchronous variational contact mechanics**. *Computer Methods in Applied Mechanics and Engineering*, 200(25–28):2181–2194, 2011. doi:10.1016/j.cma.2011.03.010.

[Wardetzky et al., 2007] Max Wardetzky, Miklós Bergou, David Harmon, Denis Zorin, and Eitan Grinspun. **Discrete Quadratic Curvature Energies**. *Computer Aided Geometric Design*, 24(8-9):499–518, Nov 2007.

[Wardetzky et al., 2008] Max Wardetzky, Miklós Bergou, Akash Garg, David Harmon, Denis Zorin, and Eitan Grinspun. **Discrete Differential Geometry: An Applied Introduction**. In *SIGGRAPH Asia '08: ACM SIGGRAPH Asia 2008 Courses*, New York, NY, USA, 2008. ACM.

[Williams, 2010] Robert W. Williams. **Measuring and modeling the anisotropic, nonlinear and hysteretic behavior of woven fabrics**. PhD thesis, University of Iowa, December 2010. URL `http://ir.uiowa.edu/etd/907`.

[Wineman and Pipkin, 1964] A.S. Wineman and A.C. Pipkin. **Material symmetry restrictions on constitutive equations**. *Archive for Rational Mechanics and Analysis*, 17(3):184–214, 1964. ISSN 0003-9527. doi:10.1007/BF00282437.

[Witkin and Baraff, 1997] Andrew Witkin and David Baraff. **Physically Based Modeling: Principles and Practice**. In *SIGGRAPH '97: ACM SIGGRAPH 1997 Courses*. ACM, 1997.

[Wu et al., 2003] Z. Wu, C.K. Au, and Matthew Yuen. **Mechanical properties of fabric materials for draping simulation**. *Interna-*

tional Journal of Clothing Science and Technology, 15(1):56–68, 2003. doi:10.1108/09556220310461169.

[Yan, 1999] Baisheng Yan. **Existence and Regularity Theory for Nonlinear Elliptic Systems and Multiple Integrals in the Calculus of Variations**. Course notes, Math 994-01, Spring '99, 1999. URL `http://users.math.msu.edu/users/yan/full-notes.pdf`.

[Zhang and Fu, 2000] Y.T. Zhang and Y.B. Fu. **A micromechanical model of woven fabric and its application to the analysis of buckling under uniaxial tension: Part 1: The micromechanical model**. *International Journal of Engineering Science*, 38(17):1895–1906, 2000. ISSN 0020-7225. doi:10.1016/S0020-7225(00)00012-4.

[Zhang and Fu, 2001] Y.T. Zhang and Y.B. Fu. **A micro-mechanical model of woven fabric and its application to the analysis of buckling under uniaxial tension. Part 2: buckling analysis**. *International Journal of Engineering Science*, 39(1):1–13, 2001. ISSN 0020-7225. doi:10.1016/S0020-7225(00)00013-6.

[Zheng and James, 2011] Changxi Zheng and Doug L. James. **Toward High-Quality Modal Contact Sound**. *ACM Trans. Graph.*, 30(4):38:1–38:12, July 2011.

[Zheng, 1993a] Q.-S. Zheng. **On transversely isotropic, orthotropic and relative isotropic functions of symmetric tensors, skew-symmetric tensors and vectors. Part I: Two dimensional orthotropic and relative isotropic functions and three dimensional relative isotropic functions**. *International Journal of Engineering Science*, 31(10):1399 – 1409, 1993a. ISSN 0020-7225. doi:10.1016/0020-7225(93)90005-F.

[Zheng, 1993b] Quanshui Zheng. **Two-dimensional Tensor Function Representation for All Kinds of Material Symmetry**. *Proceedings: Mathematical and Physical Sciences*, 443(1917):127–138, Oct 1993b.

[Zheng, 1994] Quanshui Zheng. **Theory of Representations for Tensor Functions—A Unified Invariant Approach to Constitutive Equations**. *Applied Mechanics Reviews*, 47(11):545–587, 1994. doi:10.1115/1.3111066.

[Zheng and Boehler, 1994] Quanshui Zheng and J. P. Boehler. **The description, classification, and reality of material and physical symmetries**. *Acta Mechanica*, 102(1–4):73–89, 1994. doi:10.1007/BF01178519.

[Zhou et al., 2009] Wenwen Zhou, Joshua D. Griffin, and Ioannis G. Akrotirianakis. **A globally convergent modified conjugate-gradient line-search**

algorithm with inertia controlling. Technical Report 2009-01, SAS Institute Inc., September 2009. URL http://www.optimization-online.org/DB_HTML/2009/09/2406.html.