Lagrangian-on-Lagrangian Garment Design

Juan Sebastián Montes Maestre

A Thesis

in

The Department

of

Computer Science and Software Engineering

Presented in Partial Fulfillment of the Requirements

For the Degree of Doctor of Philosophy

Concordia University

Montréal, Québec, Canada

September 2020

Concordia University

School of Graduate Studies

This is to certify that the thesis prepared

By: Mr. Juan Sebastián Montes Maestre

Entitled: Lagrangian-on-Lagrangian Garment Design

and submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy (Computer Science)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality .

Signed by the final examining commitee:

Dr. Alex De Visscher	Chair
Dr. Paul Kry	External Examiner
Dr. Nizar Bouguila	External to Program
Dr. Charalambos Poullis	Examiner
Dr. Thomas Fevens	Examiner
Dr. Tiberiu Popa	Supervisor
Dr. Bernhard Thomaszewski	Co-supervisor

Approved

Dr. Lata Narayanan, Graduate Program Director

September 2nd, 2020
Date of Defence

Dr. Mourad Debbabi, Dean
Gina Cody School of Engineering & Computer Science

Abstract

Lagrangian-on-Lagrangian Garment Design

Juan Sebastián Montes Maestre, Ph.D.

Concordia University, 2020

Since the discovery of elastomeric materials, such as spandex or lycra, skintight clothing has revolutionized many different areas of the clothing industry, such as body-shaping clothing, athletic wear, and medical garments, among others. Often, this kind of clothing is designed to fulfill a given purpose, such as providing comfort, mobility, or improving recovery in the case of an athlete, provide support or exert some desired pressure in the case of medical garments, or actively deform the body to acquire some desired shape. Additionally, some designs aim to improve the life of the garment by, for example, minimizing tractions across the seams. While many tight-skin garments are sold in the market for generic body shapes, many of the purposes here mentioned are only achievable through a personalized fitting. To this end, we introduce a novel model, where the cloth is modeled as a membrane, parameterized as a function of the body. The cloth, is then able to slide on the body and deform it while staying always in contact. We call this model Lagrangian-on-Lagrangian. Based on this model, we develop an optimization framework, based on sensitivity analysis, capable of developing sewable patterns such that, when worn by a person, satisfy a given design target. With the framework, we include several design targets such as, body shape, stretch, pressure, sliding under motion, and seam traction. We evaluate our method on a variety of applications, as well as body shapes.

Acknowledgments

To my supervisors Tiberiu Popa and Bernhard Thomaszewski for their guidance. To my parents Arturo Montes and Jacqueline Maestre, and my sister Juliana Montes for their emotional support.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Whether as casual clothing, functional sportswear, or medical compression garments—skintight clothing has many applications, and *fit* is of central importance to all of them. The fit of a garment is determined by its design which, from a technical perspective, consists of two components: (1) a layout that determines the number of patterns and how they connect to each other and (2) the shape of the individual patterns. When fitting a design to a given body shape, the layout is typically kept fix, whereas the pattern shapes are adjusted in order to accommodate different body shapes and sizes. This task of *pattern grading* is a challenging problem, since the designer has to simultaneously consider multiple criteria that relate to the state of the garment once worn.

Although shape is largely determined by the underlying body, there is often substantial room for shape control within the limits of comfort and physics. The shape and location of the seams on the body is another design consideration, important for both aesthetic and functional goals. Apart from these visual criteria, there are several objectives relating to the deformations induced in clothing and body. For example, excessive tensile deformations will affect comfort and may cause fabric and seams to deteriorate prematurely. Compressions, on the other hand, induce wrinkles that are typically perceived as design flaws in tight-fitting clothing. Designing pattern shapes that strike an ideal balance between these criteria requires time and expertise, both of which are important cost factors. With the increased demand for personally fitted clothing for multiple purposes requiring higher levels of precision, such as medical pressure therapy masks, athlete sportswear or diabetic clothing, systems that design skintight clothing automatically could prove beneficial in reducing production costs and improving accuracy.

Designing an automated system for skintight clothing requires combining knowledge from multiple fields. Skintight clothing is seen as a coupled system where the cloth interacts with the body. While some challenges involved in simulating these kinds of coupled systems have been addressed, optimizing for some desired properties in such systems introduces new challenges. For instance, existing coupled simulation frameworks, which being discontinuous, are incompatible with continuous optimization frameworks. Also, some behavior of the cloth such as wrinkling introduces discontinuities of its own.

In this work, we present an automated, optimization-driven fitting approach for skintight clothing. In general terms, we propose a unified simulation model that represents cloth as a two-dimensional elastic membrane embedded in the surface of the deformable body mesh. This *Lagrangian-on-Lagrangian* approach removes the need for detecting and handling collisions between body and cloth. Our approach supports continuous tangential motion (i.e., sliding) of cloth on smooth body meshes during simulation and optimization, allowing us to take advantage of efficient continuous optimization methods. We introduce a set of design objectives that model various design goals related to shape, comfort, and function. In particular, our method allows for minimizing traction forces on the seams, for enforcing lower and upper bounds on deformations to prevent wrinkles and material failure, for controlling pressure forces exerted on the body, and for modelling body shapes and contours.

From a technical perspective, we introduce the following novel contributions:

1. A complex optimization system based on sensitivity analysis that automatically designs cloth patterns based on some user-defined objectives.

2. A coupled cloth-body simulation model based on subdivision surfaces that considers deformation of the body and that can be used with continuous optimization frameworks.

3. A cloth model capable of generating proper physical response to wrinkling while being apt for use with continuous optimization frameworks.

4. A compactness regularizer, capable of maintaining proper pattern shapes during optimization, regardless of the objective target.

5. A method to compute pressure that is accurate and reliable on common general linear triangle meshes.

We demonstrate our method on a set of designs that are representative of different use cases for skintight clothing. We show examples from casual clothing, personalized sportswear, and patient-specific compression garments.

This thesis is divided in 8 chapters where 2, 3 and 4 cover the required background theory, chapter 5 covers the related research work, and chapters 6, 7 and 8 include our contributions.

Chapter 2

Numerical Optimization

In recent years, we have seen a rise in the complexity of the problems requiring solutions in fields across engineering and science. Thanks to modern algorithms, problems such as, finding the trajectory of a rocket, estimating the right amount of product to manufacture or finding the equilibrium of a complex physical system, are not only solvable but allow for the inclusion of complex constraints. All these problems have one thing in common in that they can be framed as the minimization of a mathematical function with respect to some design parameters. We focus solely on continuously differentiable, convex functions. That is to say, given a function $f(\boldsymbol{x})$, where $\boldsymbol{x}$ are its design parameters, the gradient $\frac{df}{d\boldsymbol{x}}$ is continuous.

In general, minimizing a function $f(\boldsymbol{x})$ can be done by finding the roots of its derivative $\frac{df}{d\boldsymbol{x}} = 0$. For a small set of functions, the roots of the derivative can be found analytically, such as quadratic functions. For the vast majority of cases, analytical solutions are impossible or impractical, so we must resort to numerical algorithms that exploit properties of the gradient to find its roots. Problems that do not specify any constraints on its design parameters are known as unconstrained problems. However, very few real-world problems are devoid of any constraints. For example, in a physical system, a common constraint is that the system is at equilibrium at all times, that is to say, the sum of its forces be 0. In this case the constrained optimization problem is established as finding the minimum of $f(\boldsymbol{x})$, subject to $c = [F(\boldsymbol{x}) = 0]$. The constraint c is known as an equality constraint.

In this chapter, we explore some of the most popular algorithms used in unconstrained numerical optimization such as gradient descent, Newton and L-BFGS, and constrained numerical optimization such as merit functions, KKT systems, and sensitivity analysis. The content of this chapter is based on the work of Nocedal and Wright [38].

2.1 Unconstrained Optimization

In this section, we cover the most common algorithms used to solve unconstrained optimization problems: Gradient descent, Newton, and L-BFGS. All three have in common that they exploit low degree polynomial approximations of the function to iteratively find better solutions for the design parameters. Given a current solution $\boldsymbol{x}_i$, an approximation to the function at the point $\boldsymbol{x}_i$ can be found by performing a Taylor series expansion, defined as

$$f = \sum_{n=0}^{\infty} \frac{f^{(n)}(\boldsymbol{x}_i)}{n!}(\boldsymbol{x} - \boldsymbol{x}_i)^n, \tag{1}$$

where n is the degree of the polynomial approximation and $f^{(n)}$ is the n-th derivative of the function. It can be seen then that approximating a linear function requires just the gradient of the function and approximating a quadratic function requires the second derivative. As the degree of the polynomial increases, the approximated function approaches the original. With this information in mind, we move on to the methods.

2.1.1 Gradient Descent

The gradient of a function can be geometrically interpreted as the direction at which a function increases the most, and its magnitude by how much. So, as its name implies, Gradient Descent consists of calculating the gradient at a non-optimal point $\boldsymbol{x}_i$ and taking a step in the direction opposite to it. A function is then minimized by taking consecutive steps until a minimum is reached. If the gradient is greater than 0, given a step $\boldsymbol{h} = -\alpha \frac{df}{d\boldsymbol{x}}(\boldsymbol{x}_i)$, an $\alpha > 0$ is guaranteed to exist such that $f(\boldsymbol{x}_i + \boldsymbol{h}) < f(\boldsymbol{x}_i)$. If we take an infinite number of infinitesimal steps, we are guaranteed to eventually land on a point $\boldsymbol{x}_o$ where $\frac{df}{d\boldsymbol{x}}(\boldsymbol{x}_o) = 0$, thus finding a solution to the problem. However, taking an infinite number of steps is not practical for computers with limited resources, so gradient descent is coupled with strategies to estimate good step lengths that help the method converge faster. One of such strategies is the Line Search method.

Line Search Given a descent step $\boldsymbol{h} = -\alpha \frac{df}{d\boldsymbol{x}}(\boldsymbol{x}_i)$, the objective of the Line Search method is to find an α such that, ideally, the function is minimized the most. This is done by evaluating the function $f(\boldsymbol{x}_i + \boldsymbol{h})$ for different α and choosing the one which shows the most improvement. One can improve the efficacy of the line search method by using information of the function and its gradient at the different sampled points. For instance, given two sampled values of α, α_1 and α_2, we obtain two evaluations for the function $f_1 = f(\boldsymbol{x}_i + \boldsymbol{h}_1)$ and $f_2 = f(\boldsymbol{x}_i + \boldsymbol{h}_2)$ with their respective gradients $\boldsymbol{g}_1$ and $\boldsymbol{g}_2$, we can interpolate a one dimensional function which can approximate the cross-section between the two points along the step. In practice, a cubic function of the form $f(\alpha) \approx p(\alpha) = a_1\alpha^3 + a_2\alpha^2 + a_3\alpha + a_4$ is often used and the coefficients a_1, a_2, a_3, a_4 are found by solving a linear system using the the function and gradient information at the two sampled points. This method is known as inexact Line Search with cubic interpolation, and the next α is found by minimizing the function $p(\alpha)$.

Convergence Despite its simplicity and the use of a good Line Search method to improve the performance of Gradient Descent, it is generally outperformed by most modern smooth convex optimization methods. Gradient Descent is known to converge linearly, making it the slowest of the methods presented here.

2.1.2 Newton

The minimum of a multivariate quadratic function can be found directly by solving a system of linear equations. Suppose a quadratic expression $\frac{1}{2}\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x} + \boldsymbol{x}^T\boldsymbol{b} + c = 0$, where $\boldsymbol{x}$ is a vector of unknowns, $\boldsymbol{A}$ is a matrix of coefficients, $\boldsymbol{b}$ is a vector of coefficients and c is just a scalar. By taking the derivative of that expression, its minimum can be obtained by solving the linear system $\boldsymbol{A}\boldsymbol{x} + \boldsymbol{b} = 0$. Such a system can be solved using a proper computer algorithm known today depending on the properties of $\boldsymbol{A}$ (e.g., Cholesky factorization for symmetric definite matrices).

The idea behind Newton is simple. We want to approximate a quadratic function at a current iteration $\boldsymbol{x}_i$ and take a step towards its minimum assuming it is a good approximation of the original function. Using Taylor series expansion, we find that a degree 2 approximation of a function $f(\boldsymbol{x})$ at a point $\boldsymbol{x}_i$ with step $\boldsymbol{h}$ is given by the expression

$$f(\boldsymbol{x}_i + \boldsymbol{h}) \approx f(\boldsymbol{x}_i) + \boldsymbol{h}^T\boldsymbol{g}(\boldsymbol{x}_i) + \frac{1}{2}\boldsymbol{h}^T\boldsymbol{H}(\boldsymbol{x}_i)\boldsymbol{h}, \tag{2}$$

where $\boldsymbol{g}$ is the gradient of the function and $\boldsymbol{H}$ is a symmetric matrix that represents the Hessian (the second derivative) of the function. The optimal step $\boldsymbol{h}$ is then found by solving the linear system

$$\boldsymbol{H}\boldsymbol{h} = -\boldsymbol{g}. \tag{3}$$

If the original function is quadratic on the unknowns, Newton finds its minimum in a single step. However, for general non-linear convex functions, Newton requires multiple steps like Gradient Descent.

Hessian Definiteness Solving Eq. 3 implies the Hessian $\boldsymbol{H}$ is invertible (has full rank / all positive eigenvalues). Unfortunately, cases, where the Hessian is not invertible, can be common. Geometrically, the Hessian may have zeroes in its eigenvalues in regions with zero curvature (e.g., a straight line) or negative in saddle points. To account for these cases, the Newton step is usually complemented by an enrichment which regularizes the Hessian, allowing us to take a step in a descent direction, despite having an ill-defined Hessian. The updated step enrichment is defined as

$$(\boldsymbol{H} + \alpha \boldsymbol{I})\boldsymbol{h} = -\boldsymbol{g}, \tag{4}$$

where $\boldsymbol{I}$ is the identity matrix and α is a scalar representing a damping parameter. This step is known as a damped Newton step. Note that for a large enough α, the step $\boldsymbol{h}$ goes more in the direction of the negative gradient (making it approach a Gradient Descent step), while an α close to 0 means the Hessian is full rank and need not be regularized. In practice, α is large at the start of the optimization, when far from the solution, and it becomes 0 towards the end.

Damped Newton step We start with an initial choice of $\alpha > 0$. We increase α exponentially for every attempt the combined matrix $\boldsymbol{H} + \alpha \boldsymbol{I}$ is not full rank. Otherwise, we take a candidate step h_c. If $f(\boldsymbol{x}_i + \boldsymbol{h_c}) > f(\boldsymbol{x}_i)$, the step is not a descent direction, so we increase α and try again. If $\boldsymbol{h}_c$ is a descent direction, we can estimate how good the step is. Recall the quadratic model we built in Eq. 2. We can define a step quality measure as

$$\delta = \frac{f(\boldsymbol{x}_i + \boldsymbol{h}_c) - f(\boldsymbol{x}_i)}{\boldsymbol{h}_c^T \boldsymbol{g}(\boldsymbol{x}_i) + \frac{1}{2}\boldsymbol{h}_c^T \boldsymbol{H}(\boldsymbol{x}_i)\boldsymbol{h}_c}, \tag{5}$$

where the denominator represents an improvement of the function per the quadratic model and the numerator is the actual improvement. As the ratio gets closer to 1, the function resembles a quadratic one, thus enabling us to adapt α accordingly.

Convergence It is well known that Newton converges quadratically, which is much faster than Gradient Descent (for a function that converges in 10 iterations with Newton, converges in around 100 iterations for Gradient Descent). It does, however, include the additional cost of having to compute the Hessian, which can be expensive depending on the function. For this reason, Newton is usually preferred for functions with a small number of unknowns, or mostly, for functions with sparse Hessians.

2.1.3 BFGS

In the previous two sections, we introduced two numerical optimization methods, Gradient Descent, and Newton. We observe that Gradient Descent is fast to compute but slow to converge and Newton can be fast to converge but requires the computation of a Hessian. However, there is a way to speed up Gradient Descent using concepts of Newton. Known as BroydenâĂŞFletcherâĂŞGoldfarbâĂŞShanno algorithms, this optimizer requires only the gradient while approximating the Hessian after each iteration. For this reason, it is also known as quasi-Newton methods. Each iteration of BFGS is defined as

$$\boldsymbol{B}\boldsymbol{h} = -\boldsymbol{g}, \tag{6}$$

where $\boldsymbol{B}$ is the accumulated approximation of the Hessian. The approximation of the Hessian is based on backward finite differences of the gradient as

$$\boldsymbol{B}(\boldsymbol{x}_{i+1} - \boldsymbol{x}_i) = \boldsymbol{g}(\boldsymbol{x}_{i+1}) - \boldsymbol{g}(\boldsymbol{x}_i). \tag{7}$$

Updating the Hessian We start from an initial approximation $\boldsymbol{B}_0$ which can be the identity, and for each iteration we update the approximation as $\boldsymbol{B}_{i+1} = \boldsymbol{B}_i + \boldsymbol{V}$, where $\boldsymbol{V}$ is a matrix. Other desired properties we want for the Hessian is for it to be positive definite and symmetric. To achieve symmetry, we can have $\boldsymbol{V} = \alpha\boldsymbol{u}\boldsymbol{u}^T + \beta\boldsymbol{v}\boldsymbol{v}^T$, where $\boldsymbol{u}$, $\boldsymbol{v}$, α and

β are defined as

$$\alpha = \frac{1}{\boldsymbol{y}_i^T \boldsymbol{s}_i}, \tag{8}$$

$$\beta = -\frac{1}{\boldsymbol{s}_i^T \boldsymbol{B}_i \boldsymbol{s}}, \tag{9}$$

$$\boldsymbol{u} = \boldsymbol{y}_i, \text{ and} \tag{10}$$

$$\boldsymbol{v} = \boldsymbol{B}_i \boldsymbol{s}_i, \tag{11}$$

where $\boldsymbol{s}_i = \boldsymbol{x}_{i+1} - \boldsymbol{x}_i$ and $\boldsymbol{y}_i = \boldsymbol{g}(\boldsymbol{x}_{i+1}) - \boldsymbol{g}(\boldsymbol{x}_i)$. In order to achieve positive definiteness, each step must satisfy constraints known as the Wolfe conditions.

Wolfe Conditions In general, we may be tempted to accept all steps that improve the function, such that given a step $\boldsymbol{h}$, $f(\boldsymbol{x}_i + \boldsymbol{h}) \leq f(\boldsymbol{x}_i)$. However, to guarantee convergence, a stronger version of this rule is used, the expression $f(\boldsymbol{x}_i + \boldsymbol{h}) < f(\boldsymbol{x}_i) + c_1 \boldsymbol{g}(\boldsymbol{x}_i)^T \boldsymbol{h}$, where c_1 is a parameter of sufficient decrease between 0 and 1. This rule is known as the Armijo condition. A condition is also placed on the curvature $\boldsymbol{h}^T \boldsymbol{g}(\boldsymbol{x}_i + \boldsymbol{h}) \geq c_2 \boldsymbol{h}^T \boldsymbol{g}(\boldsymbol{x}_i)$, where c_2 is a parameter of sufficient decrease for the gradient ($\boldsymbol{h}^T \boldsymbol{g}(\boldsymbol{x}_i)$ being negative), similar to c_1. Together with the Armijo condition, these 2 are called the Wolfe conditions.

Convergence It can be observed that BFGS starts with a Gradient Descent step, slowly transitioning into Newton as the optimization progresses. While the method is not as fast as Newton in terms of the number of iterations required, it is a vast improvement over Gradient Descent, while only requiring the gradient to operate.

2.2 Constrained Optimization

In the previous section, we analyzed methods for optimizing functions $f(\boldsymbol{x})$ without any given constraints. In this section, we explore optimization methods used to find minima of functions subjected to equality constraints of the form

$$\operatorname{argmin}_{\boldsymbol{x}} \; f(\boldsymbol{x}) \text{ s.t. } c(\boldsymbol{x}) = 0, \tag{12}$$

where $c(\boldsymbol{x}) = 0$ is the constraint we wish to satisfy. Here, we explore three methods of enforcing these types of constraints which allow the use of unconstrained optimization methods.

2.2.1 Merit Functions

Merit functions work by adding penalty terms to the original function when a constraint $c(\boldsymbol{x})$ is violated. More specifically, we obtain the minimization problem

$$\operatorname{argmin}_{\boldsymbol{x}} \; r(\boldsymbol{x}) = f(\boldsymbol{x}) + \alpha c(\boldsymbol{x}), \tag{13}$$

where α is a fixed weight representing the strength of the penalty term. While this method may bring down the measure of the violation, it does not guarantee the constraint will be fully enforced. In some cases, this is even desired behavior, such as the case of regularizers, but in some other cases, more complex solutions are needed.

2.2.2 Lagrange Multipliers

To minimize functions where we want the constraints to be fully satisfied, a commonly used method is that of Lagrange multipliers, where the function is enriched with an additional variable λ. It results in the minimization problem

$$\operatorname{argmin}_{\boldsymbol{x}} r(\boldsymbol{x}) = f(\boldsymbol{x}) + \lambda c(\boldsymbol{x}), \tag{14}$$

whose derivatives with respect to $\boldsymbol{x}$ and λ are

$$\frac{\partial f}{\partial \boldsymbol{x}} = g(\boldsymbol{x}) + \lambda \frac{\partial c(\boldsymbol{x})}{\partial \boldsymbol{x}}, \text{ and} \tag{15}$$

$$\frac{\partial f}{\partial \lambda} = c(\boldsymbol{x}), \tag{16}$$

where it is clear that for a $\lambda \neq 0$ that minimizes $r(\boldsymbol{x})$, also satisfies the constraint $c(x) = 0$.

This method may be easily generalized to multiple equality constraints $r(\boldsymbol{x}) = f(\boldsymbol{x}) + \lambda_1 c_1(\boldsymbol{x}) + \lambda_2 c_2(\boldsymbol{x}) \dots \lambda_n c_n(\boldsymbol{x})$ and can be used along with Newton, if the constraints are convex. The Newton step is defined as

$$\begin{pmatrix} \boldsymbol{H}(\boldsymbol{x}) & \frac{\partial \boldsymbol{c}(\boldsymbol{x})}{\partial \boldsymbol{x}} \\ \frac{\partial \boldsymbol{c}(\boldsymbol{x})}{\partial \boldsymbol{x}} & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{x} \\ \lambda \end{pmatrix} = \begin{pmatrix} -\boldsymbol{g}(x) - \frac{\partial \boldsymbol{c}(x)}{\partial \boldsymbol{x}} \\ -\boldsymbol{c}(\boldsymbol{x}) \end{pmatrix}. \tag{17}$$

2.2.3 Sensitivity Analysis

In the previous two methods, we assumed that the constraints may be violated anytime during the optimization. An alternative approach is to explore the space of solutions where the constraint is already satisfied. In this section, we present the sensitivity analysis (Spivak [54]) method, and more specifically, sensitivity analysis by the implicit function theorem (IFT) for equality constraints. IFT states, that for a function $f(\boldsymbol{x}, \boldsymbol{p})$ ($\boldsymbol{x}$ and $\boldsymbol{p}$ both variables) with equality constraints $\boldsymbol{c}(\boldsymbol{x}, \boldsymbol{p}) = \boldsymbol{0}$, there must exist a function $\boldsymbol{x}(\boldsymbol{p})$ where the constraint is satisfied. The Jacobian $\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{p}}$ is known as the sensitivity matrix.

While the explicit function $\boldsymbol{x}(\boldsymbol{p})$ is often hard to find, the sensitivity matrix can be found numerically from the fact that $\frac{d\boldsymbol{c}}{d\boldsymbol{p}} = \mathbf{0}$, given that $\boldsymbol{c}(\boldsymbol{x}, \boldsymbol{p}) = \mathbf{0}$ is always satisfied. If we take the full derivative of the constraint $\boldsymbol{c}$ with respect to $\boldsymbol{p}$ we have that

$$\frac{d\boldsymbol{c}}{d\boldsymbol{p}} = \frac{\partial \boldsymbol{c}}{\partial \boldsymbol{p}} + \frac{\partial \boldsymbol{c}}{\partial \boldsymbol{x}}\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{p}}, \tag{18}$$

which results in the expression for the sensitivity matrix

$$\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{p}} = -\frac{\partial \boldsymbol{c}}{\partial \boldsymbol{x}}^{-1}\frac{\partial \boldsymbol{c}}{\partial \boldsymbol{p}}. \tag{19}$$

With this in mind, we look at the minimization problem

$$\text{argmin}_{\boldsymbol{x},\boldsymbol{p}}\ f(\boldsymbol{x}, \boldsymbol{p}) \text{ s.t. } \boldsymbol{c}(\boldsymbol{x}, \boldsymbol{p}) = \mathbf{0}. \tag{20}$$

At any iteration, for any given $\boldsymbol{p}_i$ we can find $\boldsymbol{x}_k$ such that $\boldsymbol{c}(\boldsymbol{x}_k, \boldsymbol{p}_i) = \mathbf{0}$. Such a problem can be solved using any unconstrained optimization method mentioned here. Given that changes of $\boldsymbol{x}$ are attached to changes of $\boldsymbol{p}$, we minimize the function f with respect to $\boldsymbol{p}$. The gradient of f with respect to $\boldsymbol{p}$ is therefore defined as

$$\frac{df}{d\boldsymbol{p}} = \frac{\partial f}{\partial \boldsymbol{p}} + \frac{\partial \boldsymbol{x}}{\partial \boldsymbol{p}}^{T}\frac{\partial f}{\partial \boldsymbol{x}}, \tag{21}$$

where the matrix $\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{p}}$ is the one from Eq. 19.

Chapter 3

Finite Element Method

The Finite Element Method is a popular choice for numerically solving systems of partial differential equations, such as elasticity. It generally works by approximating the system into a set of smaller, simpler elements for which the solution is known. In this chapter, we provide an overview on how to handle these elements, and how to use them to simulate elastic objects.

In order to approximate finite elements to a space, we first need to define methods to approximate a set of functions to another continuous functions, and also to partial differential equations. These methods are explored in Sec. 3.1. In Sec. 3.2, we explore how these finite elements can be used to approximate elastic objects and deformations (as in Zienkiewicz et al. [70]).

3.1 Approximation of Functions

In this section we explore how to approximate continuous functions by a set of different functions. Two equivalent ways to do this, known as the Least-Squares approximation, and the Galerkin projection method, are presented in Sec. 3.1.1 (Rencher [45]), and Sec. 3.1.2 respectively. In Sec. 3.1.3 we explore how to approximate a function using finite elements with the method exposed in Sec. 3.1.2. In Sec. 3.1.4, we show how to transform a PDE problem into an energy minimization problem (Farlow [21]), to solve it with finite elements.

3.1.1 Least-Squares method

The least-squares approximation method is commonly used to fit a polynomial function to a set of data points. Given a set of data points $\boldsymbol{p}_1, \boldsymbol{p}_2 ... \boldsymbol{p}_m \in \mathbb{R}^2$, we want to find the coefficients $a_1 ... a_k$ of a polynomial of the form $f(x) = a_1 f_1(x) + a_2 f_2(x) + ... + a_k f_k(x)$ such that the sum of the squared errors between the function and the points is minimized. We define the error function as

$$e = \sum_{i=1}^{m} (f(p_{i_x}) - p_{i_y})^2, \tag{22}$$

where p_{i_x} and p_{i_y} are the x and y coordinates for the i-eth point respectively. To minimize this function, we take its first derivative with respect to the vector of coefficients x and make it equal to 0. Since e is a quadratic function the derivative results in a set of linear functions of the form $Ax = b$ with dimensions $m \times k$,

$$\begin{pmatrix} f_1(p_{1_x}) & \dots & f_k(p_{1_x}) \\ \vdots & \ddots & \vdots \\ f_1(p_{m_x}) & \dots & f_k(p_{m_x}) \end{pmatrix} \begin{pmatrix} a_1 \\ \vdots \\ a_k \end{pmatrix} = \begin{pmatrix} p_{1_y} \\ \vdots \\ p_{m_y} \end{pmatrix}. \tag{23}$$

In practice, $m > k$ most of the time, which results in an over-constrained system. The coefficients can be obtained using the pseudo-inverse $\mathbf{a} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{r}$.

For example, let's say we want to find the function of the form $f(x) = a_1 x^2 + a_2 x$ that best fits the points $\boldsymbol{p} = (0, 0), (0.5, 0.4794), (1, 0.8414)$ which were generated using the

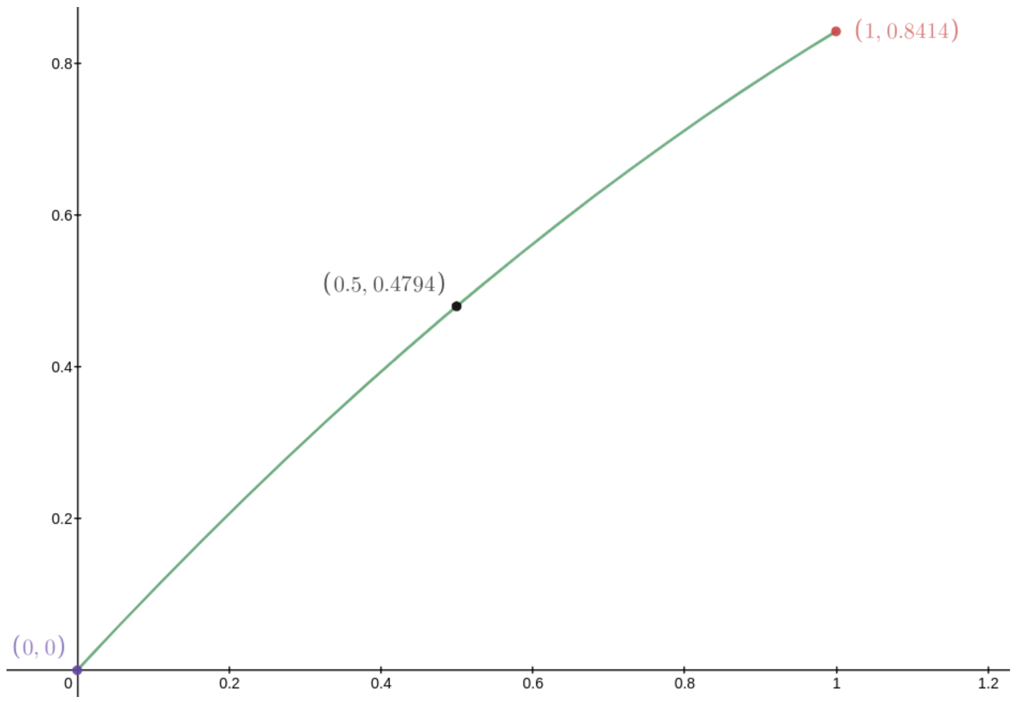


Figure 1: Optimized quadratic function $f(x) = -0.2348x^2 + 1.0762x$ that interpolates the points $(0, 0), (0.5, 0.4794), (1, 0.8414)$.

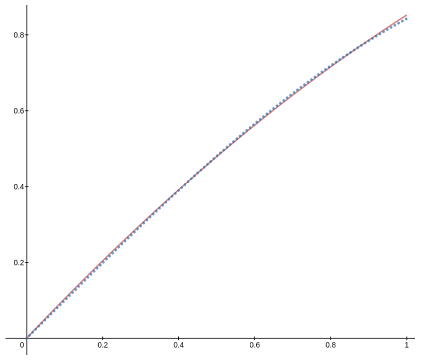
function $sin(x)$. We obtain the system

$$\begin{pmatrix} 0^2 & 0 \\ 0.5^2 & 0.5 \\ 1^2 & 1 \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0.4794 \\ 0.8414 \end{pmatrix}. \tag{24}$$

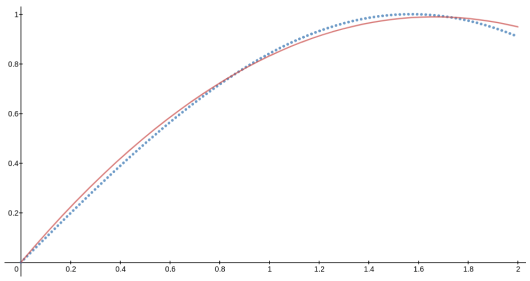
Solving for the coefficients, we get $a_1 = -0.2348$, $a_2 = 1.0762$ resulting in the optimized function $f(x) = -0.2348x^2 + 1.0762x$ (Fig. 1).

The method presented above can be described as a least-squares approximation of discrete functions. However, we are more interested in the approximation of continuous functions. To achieve this, the discrete least-squares formulation is turned into the more general

$$e = \int_{x_0}^{x_1} (f(x) - g(x))^2 dx, \tag{25}$$



(a) Quadratic function approximated to the function $g(x) = sin(x)$ between the interval [0..1].



(b) Quadratic function approximated to the function $g(x) = sin(x)$ between the interval [0..2].

Figure 2: Least-squares continuous approximation within two different intervals.

where we want to find the coefficients $\mathbf{a} = a_1..a_k$ of the function f that best approximates the function g within the intervals x_0 and x_1.

In this case, we take the partial derivatives and set them to 0 giving a linear system of k equations for k unknowns.

For example, let's approximate the function $g(x) = sin(x)$ by the function $f(x) = a_1x^2 + a_2x$ between two different intervals, [0..1] and [0..2].

We obtain two different linear systems for the two intervals

$$\begin{pmatrix} \frac{\partial \int_0^1 (a_1x^2+a_2x-sin(x))^2dx}{\partial a_1} \\ \frac{\partial \int_0^1 (a_1x^2+a_2x-sin(x))^2dx}{\partial a_2} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \text{ and } \begin{pmatrix} \frac{\partial \int_0^2 (a_1x^2+a_2(x)-sin(x))^2dx}{\partial a_1} \\ \frac{\partial \int_0^2 (a_1x^2+a_2(x)-sin(x))^2dx}{\partial a_2} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}. \qquad (26)$$

Solving for the coefficients, we get for the interval $x = [0..1]$, $a_1 = -0.2105, a_2 = 1.06144$ which are almost the same as in the discrete setting with the 3 points. In the case of the second interval $x = [0..2]$ we get the coefficients $a_1 = -0.357262, a_2 = 1.1889$ (Fig: 2).

3.1.2 Galerkin Method

To better understand the Galerkin or projection method for functions, it is helpful to look first at how it works with vectors.

Let's say we want to find the closest point of a point $\boldsymbol{p} \in \mathbb{R}^3$ to the plane spanned by the vectors $\mathbf{b}_1, \mathbf{b}_2 \in \mathbb{R}^3$. One way to solve this problem is to find the shortest distance using the least-squares method. In that sense, we want to find coefficients a_1, a_2 such that $e = \|a_1\mathbf{b}_1 + a_2\mathbf{b}_2 - \boldsymbol{p}\|_2^2$ is minimized. From the previous section, we know we can compute the derivatives and find a solution.

However, it can be easily observed that the shortest distance from $\boldsymbol{p}$ to the plane is the one that goes in the orthogonal direction from the plane. This also means that the direction which is the shortest is orthogonal to both $\mathbf{b}_1$ and $\mathbf{b}_2$. With this information, we can reformulate this problem.

Find the coefficients a_1 and a_2 such that:

$$(\boldsymbol{p} - a_1\mathbf{b}_1 - a_2\mathbf{b}_2) \cdot \mathbf{b}_1 = 0, \tag{27}$$

and

$$(\boldsymbol{p} - a_1\mathbf{b}_1 - a_2\mathbf{b}_2) \cdot \mathbf{b}_2 = 0, \tag{28}$$

where we are asking the projected point on the plane $\boldsymbol{p} - a_1\mathbf{b}_1 - a_2\mathbf{b}_2$ to be orthogonal to both the bases of the plane $\mathbf{b}_1$ and $\mathbf{b}_2$ by making the dot product be 0. This results in a system of 2 equations with 2 unknowns which is linear. This method is known as the Galerkin approximation or the projection method.

For example, let's try to find the projection of the point $\boldsymbol{p} = (1, 1, 1)$ to the plane spanned by the vectors $\mathbf{b}_1 = (0, 1, 1), \mathbf{b}_2 = (1, -1, 0)$. We set the systems of equations and solve for

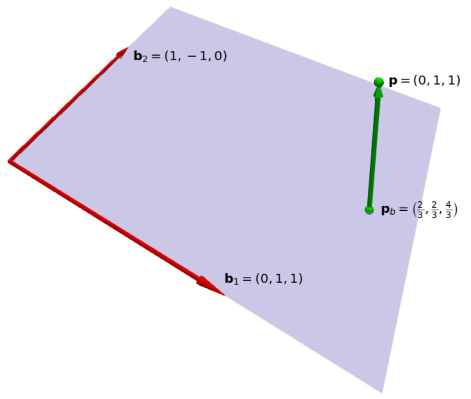

$\mathbf{b}_2 = (1, -1, 0)$
$\mathbf{p} = (0, 1, 1)$
$\mathbf{p}_b = (\frac{2}{3}, \frac{2}{3}, \frac{4}{3})$
$\mathbf{b}_1 = (0, 1, 1)$

Figure 3: Projection of the point $\boldsymbol{p} = (1, 1, 1)$ onto the plane spanned by the vectors $\mathbf{b}_1 = (0, 1, 1)$ and $\mathbf{b}_2 = (1, -1, 0)$.

a_1 and a_2 as described by

$$\begin{pmatrix} 1 - 0a_1 - 1a_2 \\ 1 - 1a_1 - (-1)a_2 \\ 1 - 1a_1 - 0a_2 \end{pmatrix} \cdot \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix} = 0, \tag{29}$$

and

$$\begin{pmatrix} 1 - 0a_1 - 1a_2 \\ 1 - 1a_1 - (-1)a_2 \\ 1 - 1a_1 - 0a_2 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ -1 \\ 0 \end{pmatrix} = 0. \tag{30}$$

This gives the linear system of equations

$$\begin{pmatrix} -2 & 1 \\ 1 & -2 \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \end{pmatrix} = \begin{pmatrix} -2 \\ 0 \end{pmatrix}, \tag{31}$$

which can be easily solved to give the coefficients $a_1 = \frac{4}{3}, a_2 = \frac{2}{3}$ and the optimal projected point $\boldsymbol{p}_b = \frac{4}{3}(0, 1, 1) + \frac{2}{3}(1, -1, 0) = (\frac{2}{3}, \frac{2}{3}, \frac{4}{3})$ (Fig. 3).

There is a couple of things to say about this example. First, we can think of the coefficients a_1, a_2 as a parameterization in the space of the basis vectors.

Second, notice that the basis vectors are not orthonormal among them. Even though we can find a valid solution, it is often desired to have orthonormality among the bases for numerical stability. For example, suppose we have bases $\mathbf{b}_1$ and $\mathbf{b}_2$ that define the same plane but are very close to each other. If we desire to define a plane point using the coefficients a_1, a_2 it can be observed that small perturbations of the point in 3D can result in big changes for the coefficients.

In our above example, we can achieve orthonormality by simply normalizing the basis vectors $\hat{\mathbf{b}_1} = \frac{\mathbf{b}_1}{\|\mathbf{b}_1\|}, \hat{\mathbf{b}_2} = \frac{\mathbf{b}_2}{\|\mathbf{b}_2\|}$ and replacing the second one with a vector lying on the same plane but orthogonal to $\mathbf{b}_1$ by using the cross-product $\hat{\mathbf{b}_3} = \hat{\mathbf{b}_1} \times \hat{\mathbf{b}_2} \times \hat{\mathbf{b}_1}$, giving the new set of basis vectors $(\hat{\mathbf{b}_1}, \hat{\mathbf{b}_3})$.

Galerkin approximation for functions

From the vector case, we can see that we need an equivalent version of the dot product for the case of functions. This is called the inner product and is defined as

$$\int_a^b f(x)g(x)dx, \tag{32}$$

where a, b represent the boundary of the domain, and $f(x)$ and $g(x)$ are the functions. Then, if we have a function $f(x)$ and a set of basis functions $(g_1(x), g_2(x), ..., g_n(x))$ we can define the Galerkin approximation as the system of equations

$$\int_a^b (f(x) - \sum_i^n a_i g_i(x))(g_1(x))dx = 0, \tag{33}$$

$$\int_a^b (f(x) - \sum_i^n a_i g_i(x))(g_2(x))dx = 0, \tag{34}$$

$$.., \tag{35}$$

$$\int_a^b (f(x) - \sum_i^n a_i g_i(x))(g_n(x))dx = 0, \tag{36}$$

where we have the approximated function $\sum_i^n a_i g_i(x)$ and the residual $(f(x) - \sum_i^n a_i g_i(x))$ to be orthogonal to every basis function $g_i(x)$. This is the same as projecting the function $f(x)$ to the space spanned by the functions $g(1)..g(n)$.

As an example, we can approximate the function $f(x) = sin(x)$ again with the functions $g_1(x) = x^2$ and $g_2(x) = x$ between the interval [0..2] as in the least-squares example. The approximated function then would be $g(x) = a_1x^2 + a_2x$ and the residual $r(x) = sin(x) - a_1x^2 - a_2x$. Projecting using the Galerkin method gives the equations

$$\int_0^2 (sin(x) - a_1x^2 - a_2x)x^2dx = 0, \text{ and} \tag{37}$$

$$\int_0^2 (sin(x) - a_1x^2 - a_2x)xdx = 0. \tag{38}$$

Rearranged, this gives us the linear system

$$\begin{pmatrix} \int_0^2 x^4dx & \int_0^2 x^3dx \\ \int_0^2 x^3dx & \int_0^2 x^2dx \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \end{pmatrix} = \begin{pmatrix} \int_0^2 x^2sin(x)dx \\ \int_0^2 xsin(x)dx. \end{pmatrix} \tag{39}$$

Solving this system gives the coefficients $a_1 = -0.357262, a_2 = 1.1889$, which is the same result as the least-squares method. The least-squares method and the Galerkin approximation are equivalent, however, it can be seen that the second is easier to compute.

As in the vector approximation, to obtain stable approximations of functions it is preferable to use orthogonal bases. In this case, x and x^2 are not orthogonal in the domain 0..2.

3.1.3 Lagrange Polynomials and Finite Elements

Up until now, we have considered approximations of functions by a single continuous function. While this is fine for simple functions, real-world problems are often defined by more complex functions. One may be tempted to use higher-order polynomials to solve these kinds of problems but it usually results in unwanted oscillations. In practice, it is often preferred to use polynomials with degrees as low as possible and achieve complexity differently.

Lagrange interpolating polynomials are polynomials of degree n which interpolates a given set of $n + 1$ points. It is defined as

$$L(x) = \sum_i N_i(x) y_i, \tag{40}$$

where the group of N_i are known as Lagrangian bases defined as

$$N_i(x) = \prod_{0 \leq j \leq i, i \neq j} \frac{x - x_j}{x_i - x_j}, \tag{41}$$

where each pair of (x_i, y_i) represents a point the polynomial interpolates.

Linear Lagrange polynomials

The simplest Lagrange polynomial is the linear interpolation function $L(x) = (1-N)y_0 + N y_1$ which interpolates two points $(x_0, y_0), (x_1, y_1)$ with a linear basis function $N = \frac{x-x_0}{x_1-x_0}$.

For example, finding the a line that goes through the points $(0, 0), (1, 2)$ can be found

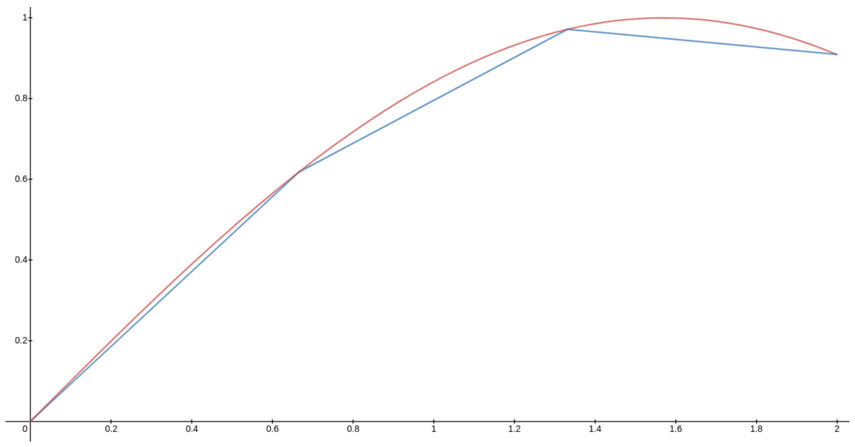


Figure 4: Approximating the sine function with three linear Lagrange polynomials.

by replacing the corresponding values in $L(x)$ which gives us the expression $L(x) = 0(1 - \frac{x-0}{1-0}) + 2\frac{x-0}{1-0}$. Simplified, we obtain the expected trivial function $L(x) = 2x$.

On its own, a single Lagrange polynomial is as powerful as a standard polynomial, but the ability to control the interpolating points opens the possibility of stitching multiple polynomials together in a piece-wise manner to approximate more complex functions. This segmentation of a function into multiple low degree polynomials is known as finite elements.

For instance, consider the example $f(x) = sin(x)$ again within the range $0..2$. Let's say we want to approximate that function using three equally spaced piece-wise linear functions, thus obtaining the samples $x_0 = 0, x_1 = \frac{2}{3}, x_2 = \frac{4}{3}, x_3 = 2$. With these samples, we can generate the interpolating points $p = \{(x_0, f(x_0)), (x_1, f(x_1)), (x_2, f(x_2)), (x_3, f(x_3))\}$ and fit each segment with the polynomials

$$L_1(x) = \left(1 - \frac{x - x_0}{x_1 - x_0}\right) f(x_0) + \frac{x - x_0}{x_1 - x_0} f(x_1), \tag{42}$$

$$L_2(x) = \left(1 - \frac{x - x_1}{x_2 - x_1}\right) f(x_1) + \frac{x - x_1}{x_2 - x_1} f(x_2), \text{ and} \tag{43}$$

$$L_3(x) = \left(1 - \frac{x - x_2}{x_3 - x_2}\right) f(x_2) + \frac{x - x_2}{x_3 - x_2} f(x_3). \tag{44}$$

Alternatively, the function can also be approximated using the Galerkin method. Let's say we divide the function into two equally spaced segments L_1 which ranges from 0 to 1 and L_2 that ranges from 1 to 2. We obtain the expressions

$$N_0 = \left(1 - \frac{x - x_0}{x_1 - x_0}\right), \tag{45}$$

$$N_1 = \frac{x - x_0}{x_1 - x_0}, \tag{46}$$

$$N_2 = \left(1 - \frac{x - x_1}{x_2 - x_1}\right), \tag{47}$$

$$N_3 = \frac{x - x_1}{x_2 - x_1}, \tag{48}$$

$$L_1(x) = N_0 y_0 + N_1 y_1, \text{ and} \tag{49}$$

$$L_2(x) = N_2 y_1 + N_3 y_2, \tag{50}$$

as seen in Fig. 4.

Notice we no longer interpolate through the function points $f(x_0), f(x_1), f(x_2)$. Instead, we are going use those weights y_0, y_1, y_2 as unknowns to find a fit that best approximates the function in a least-squares sense. It should also be noted that the weight y_1 shares two basis functions N_1, N_2. If we project using the Galerkin method on the bases N_0, N_1, N_2, N_3, we obtain in matrix form the expression

$$\begin{pmatrix} \int_0^1 N_0 N_0 & \int_0^1 N_0 N_1 & 0 \\ \int_0^1 N_1 N_0 & \int_0^1 N_1 N_1 + \int_1^2 N_2 N_2 & \int_1^2 N_2 N_3 \\ 0 & \int_0^1 N_3 N_2 & \int_1^2 N_3 N_3 \end{pmatrix} \begin{pmatrix} y_0 \\ y_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} \int_0^1 N_0 sin(x) \\ \int_0^1 N_1 sin(x) + \int_1^2 N_2 sin(x) \\ \int_1^2 N_3 sin(x) \end{pmatrix}, \tag{51}$$

which can be easily solved for the unknowns giving the result exposed in Fig. 5.

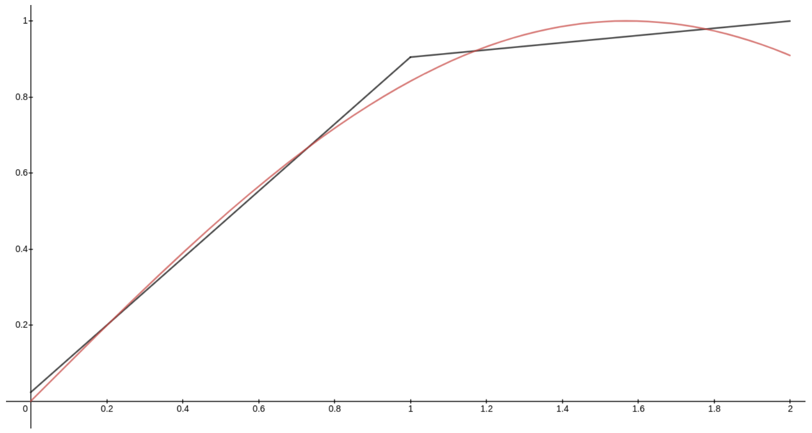


Figure 5: Approximating the sine function with two linear Lagrange polynomials using the Galerkin method.

Lagrange Polynomials in higher dimensions

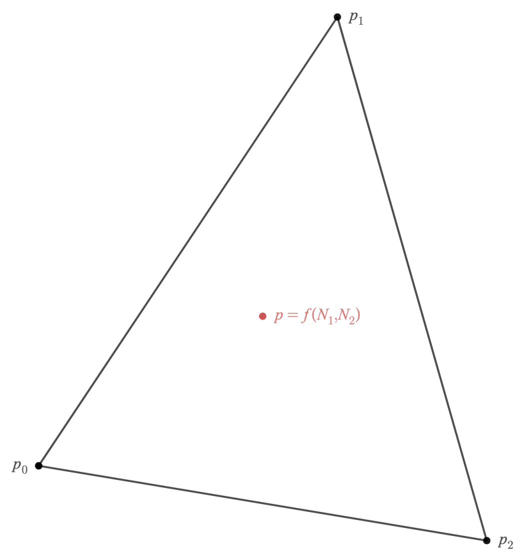
Hardly any real-world problem is 1-dimensional. Problems that concern us are mostly 3-dimensional and sometimes 2-dimensional. 1-dimensional definitions while not useful in practice, often help clarify concepts.

Fortunately, Lagrange polynomials are easily extensible to support multiple dimensions. Consider the line segment we have defined with a single function $x = \sum_i N_i x_i$. If we add in an additional function $y = \sum_i N_i y_i$, where y is orthogonal to x with the same basis function, suddenly we have a 2-dimensional line segment with the same bases. If we add yet another function $z = \sum_i N_i z_i$, z orthogonal to both x and y, we have a 3-dimensional line expressed as a parametric vector

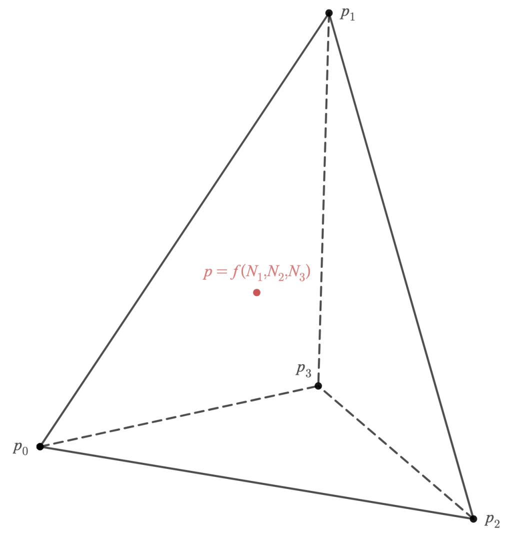
$$\boldsymbol{p}_{3D}(N) = \begin{pmatrix} (1-N)x_0 + Nx_1 \\ (1-N)y_0 + Ny_1 \\ (1-N)z_0 + Nz_1 \end{pmatrix}, \tag{52}$$

with $0 \leq N \leq 1$.

Representing lines in 2D and 3D, while useful, is still not sufficient to completely represent objects geometrically in higher dimensions as they only allow us to interpolate



(a) Linear Triangle.



(b) Linear Tetrahedron.

Figure 6: Visualization of a linear triangle and a linear tetrahedron.

length. A complete definition would allow us to interpolate areas and volumes in 2D and 3D respectively.

The linear triangle

To interpolate areas we can rely on the linear triangle (Fig. 6a), which is the equivalent of Lagrange polynomials in 2D. Where lines had one basis function and interpolated two points, a triangle has two and interpolates three points. A linear triangle is defined parametrically as

$$\begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} = \begin{bmatrix} p_{0x} & p_{1x} & p_{2x} \\ p_{0y} & p_{1y} & p_{2y} \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} N_1 \\ N_2 \\ 1 - N_1 - N_2 \end{bmatrix}, \tag{53}$$

where N_1 and N_2 are independent variables such that $0 \leq N_1, N_2 \leq 1$ and $N_1 + N_2 = 1$. If the triangle is well formed (points are not co-linear and distinct) the matrix is invertible and there is a 1-to-1 relationship between N and p.

The interpolation variables are also commonly expressed as barycentric coordinates and setting them to $N_1 = \frac{1}{2}, N_2 = \frac{1}{2}$ results in the barycenter of the triangle. The determinant of the matrix is twice the area of the triangle.

Notice that the third row is full of ones. This is to ensure that the barycentric coordinates add up to one and allows the triangle to be represented in homogeneous coordinates. If we translate the triangle to the origin, we get the simplified expression

$$\begin{bmatrix} X \\ Y \end{bmatrix} = \begin{bmatrix} p_{1x} - p_{0x} & p_{2x} - p_{0x} \\ p_{1y} - p_{0y} & p_{2y} - p_{0y} \end{bmatrix} \begin{bmatrix} N_1 \\ N_2 \end{bmatrix}, \tag{54}$$

which gives the same mapping.

It is also possible to extend the triangle basis functions to 3D as done with the line. It is enough just to add the additional linear function at the last row of the system as

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} p_{1x} - p_{0x} & p_{2x} - p_{0x} \\ p_{1y} - p_{0y} & p_{2y} - p_{0y} \\ p_{1z} - p_{0z} & p_{2z} - p_{0z} \end{bmatrix} \begin{bmatrix} N_1 \\ N_2 \end{bmatrix}. \tag{55}$$

Notice that while we can map any barycentric coordinate to a unique point in 3D, the inverse is not true. With this system, any point lying on a line perpendicular to the plane of the triangle gets mapped to the same set of barycentric coordinates.

The linear tetrahedron

To interpolate volumes, we may use the linear tetrahedron (Fig. 6b), which consists of three parametric variables and interpolates four points linearly in space.

The basis functions of the tetrahedron are expressed in non-homogeneous coordinates as

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} p_{1x} - p_{0x} & p_{2x} - p_{0x} & p_{3x} - p_{0x} \\ p_{1y} - p_{0y} & p_{2y} - p_{0y} & p_{3y} - p_{0y} \\ p_{1z} - p_{0z} & p_{2z} - p_{0z} & p_{3z} - p_{0z} \end{bmatrix} \begin{bmatrix} N_1 \\ N_2 \\ N_3 \end{bmatrix}, \tag{56}$$

where N_1, N_2, N_3 are the barycentric coordinates of the tetrahedron and $N_1 + N_2 + N_3 = 1$. The matrix is invertible as long as the four points are not co-planar.

The barycenter of the tetrahedron is obtained by setting the barycentric coordinates to $N_1 = \frac{1}{3}, N_2 = \frac{1}{3}, N_3 = \frac{1}{3}$. Its volume is one twelfth of the determinant of the matrix.

3.1.4 Solving PDEs

Partial differential equations are expressions that combine functions with multiple variables and their derivatives. The order of a PDE is determined by the degree of the highest derivative. For example $\nabla u = 0$ (where $\nabla = \frac{\partial}{\partial x}$), is a first-order differential equation while $\nabla \cdot \nabla u + \nabla u = 0$ is a second-order differential equation. Solving a PDE means finding an expression of the form $u(x)$, where x is the variable.

PDEs with a single variable are usually referred to as ordinary differential equations (ODE) and are sometimes easier to solve than PDEs.

Examples of ODEs include the Laplace equation $\Delta u = \nabla^2 u = 0$ and the Poisson equation $\Delta u = \nabla^2 u = b$. These equations are often used to model complex physical phenomena such as the behavior of fluids or elasticity in solids.

An ODE becomes a PDE once additional independent variables are added to the system. For example, the Laplace equation becomes a PDE if additional spatial variables are added. Thus, the PDE $\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0$ represents the Laplacian equation for two dimensions. To facilitate notation of PDEs, derivatives of u are often represented as subscripts, so the previous equation becomes $u_{xx} + u_{yy} = 0$.

Let's say we want to solve the ODE $u_{xx} = 0$. As it can be observed, any linear equation of the form $u(x) = ax + b$ satisfies the constraints. However, if we want to further restrain the space of solutions, we can do it through the enforcement of boundary conditions. These conditions may be applied to the function itself $u(a) = b$, in which case it is called a Dirichlet boundary condition or to its derivative $u'(a) = b$, in which case it is called a Neumann boundary condition. There exist more, but we will focus on these two.

Suppose now we want to find now the solution to the ODE $u_{xx} = 0$ subject to the Neumann boundary condition $u'(0) = 2$ and the Dirichlet boundary condition $u(0) = 1$. If the expression is integrated once, we obtain the expression $u_x = c_1$, where c_1 is a constant. To enforce the first boundary condition, it suffices to set $c_1 = 2$. Integrating once again, we obatin the expression $u(x) = 2x + c_2$. Setting $c_2 = 1$ satisfies the second boundary condition, which leaves $u(x) = 2x + 1$ as the unique solution to the ODE.

While the previous differential equation was easy to solve, it is hardly the case for most of the ODEs and even less for PDEs where analytical solutions may not even be possible. This is where finite element approximation becomes particularly useful.

Weak form

Consider the ODE $u_{xx} = -sin(x)$, with $x = [0..2]$ and with Dirichlet boundary conditions $u(0) = sin(0)$, $u(2) = sin(2)$, which has the obvious solution $u(x) = sin(x)$, but whose solution we want to find using linear finite elements.

The first obstacle to finding a solution comes right away since linear finite elements' second derivative is always 0, which means we can not solve the ODE in this form.

A common technique consists in transforming the high order differential equation into a lower one by multiplying the expression by so called "test functions" ($v(x)$) and integrating in the domain, the ODE becomes $\int_\Omega v(x)u_{xx} = -\int_\Omega v(x)sin(x)$. Integrating by parts gives

$$v(x)u_x|\Omega = \int_\Omega v_x u_x + \int_\Omega v(x)u_{xx}, \tag{57}$$

which rearranged reads

$$\int_\Omega v(x)u_{xx} = v(x)u_x|_\Omega - \int_\Omega v_x u_x. \tag{58}$$

In the resulting expression $v(x)u_x|_\Omega - \int_\Omega v_x u_x = -\int_\Omega v(x)sin(x)$ we have successfully transformed a second-order ODE into a first-order ODE. This expression is known as the weak form, as the initial requirement for a second-order ODE (strong form) has been weakened to a first-order ODE.

It can also be observed that the expression looks familiar to a Galerkin projection, which means that we have transformed a constrained problem into an energy minimization problem. It should also be noted that both forms, strong and weak are equivalent and the weak form provides exact solutions for specific functions.

Now, that the problem only requires first derivatives, we can define our approximation of $u(x)$ using finite elements. Thus, $u(x) \approx \sum_i N_i(x)\hat{u}_i$. While many choices for $v(x)$ exist, the most common one is simply to use the Lagrangian basis functions. Notice also, that under this approximation $u_x \approx \frac{\partial N}{\partial x}\hat{\boldsymbol{u}}$ since $\hat{\boldsymbol{u}}$ is just some set of weights independent of x. In the literature, $\boldsymbol{B} = \frac{\partial N}{\partial x}$ is used to simplify notation. Finally, the expression $v(x)u_x|\Omega$ vanishes and replacing the values in the ODE, we are left with the simplified expression in matrix form

$$-\int_\Omega \boldsymbol{B}^T\boldsymbol{B}\hat{\boldsymbol{u}} = -\int_\Omega \boldsymbol{N}sin(x), \tag{59}$$

where we can solve for the weights $\hat{u}$, now considered nodes. In this specific case, since we are using linear elements, the left matrix of this equation is constant on x, and the solution

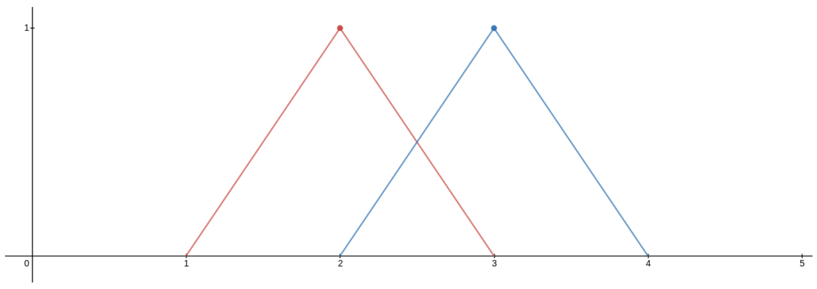


Figure 7: Linear lagrangian basis functions for nodes 2 and 3. The red lines correspond to the basis functions where it is 1 for node 2 and the blue lines where the functions are 1 for node 3.

is found just by solving a system of the type $Ax = b$. For non-linear elements, an iterative procedure can be used, such as Newton.

Boundary constraints can be enforced either by using an Augmented Lagrangian model or a combination of hard constraints and soft constraints.

Vanishing boundary integral

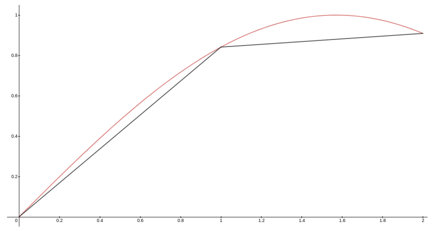
To understand how the expression $v(x)u_x|_\Omega$ vanishes, it is necessary to look at how the Lagrange basis functions work. If we consider the value of the Lagrangian bases at the nodes, we can observe that two basis functions meet at the value of 1 for a node, taking the value of 0 for all the other nodes. This behavior is illustrated in Fig. 7.

For example, the basis functions $N_{21} = \frac{x-p_1}{p_2-p_1}$ and $N_{22} = 1-\frac{x-p_2}{p_3-p_2}$ are the two contributors to node 2. If we sum their contributions we get

$$v(x)u_x|_\Omega \approx u_x\hat{v}\frac{x-p_1}{p_2-p_1}\bigg|_{p_1}^{p_2} + u_x\hat{v}\left(1-\frac{x-p_2}{p_3-p_2}\right)\bigg|_{p_2}^{p_3}, \tag{60}$$

where after replacing the integral limits we get the simplified expression

$$v(x)u_x|_\Omega \approx u_x\hat{v} - u_x\hat{v} = 0. \tag{61}$$


(a) Approximation of the ODE $\Delta x = -sin(x)$ with 2 elements compared to the analytical solution $sin(x)$.

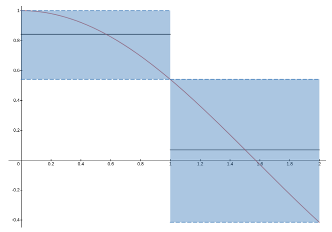
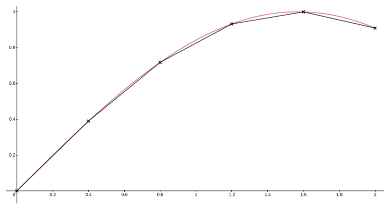

(b) Comparison of the derivative of each element to the analytical derivative $cos(x)$.

Figure 8: Finite element solution with 2 elements.


(a) Approximation of the ODE $\Delta u = -sin(x)$ with 5 elements compared to the analytical solution $sin(x)$.

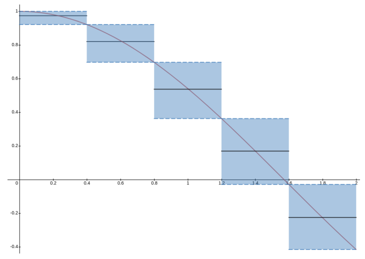

(b) Comparison of the derivative of each element to the analytical derivative $cos(x)$.

Figure 9: Finite element solution with 5 elements.

Numerical examples

In this section, we offer two numerical examples for the ODE with 2 and 5 elements.

For 2 elements we obtain the system

$$\begin{pmatrix} 1 & 0 & 0 \\ -1 & 2 & -1 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} y_0 \\ y_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0.841471 \\ 0.909297 \end{pmatrix}, \tag{62}$$

where the first row and last row of the matrix are set to identity, with the first row and last row of the right side set to $sin(0)$ and $sin(2)$ respectively to enforce the boundary conditions. Essentially, we are looking for the missing value of y_1. The result is illustrated in Fig. 8.

If we approximate the ODE with 5 elements, we obtain

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ -2.5 & 5 & -2.5 & 0 & 0 & 0 \\ 0 & -2.5 & 5 & -2.5 & 0 & 0 \\ 0 & 0 & -2.5 & 5 & -2.5 & 0 \\ 0 & 0 & 0 & -2.5 & 5 & -2.5 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \end{pmatrix} = \begin{pmatrix} 0 \\ 0.389418 \\ 0.717356 \\ 0.932039 \\ 0.999574 \\ 0.909297 \end{pmatrix}, \tag{63}$$

where again the first and last rows are constrained to enforce the boundary constraints. By looking at this example, we can see a pattern starts to emerge. Each row of the non-constrained nodes has exactly 3 elements, in fact, it is a tridiagonal matrix. If we increase the number of elements, this pattern would still be maintained.

A better way of enforcing the boundary constraints is to eliminate the rows and columns of the fixed nodes and modifying the right side accordingly. In this way, we can keep a symmetric PSD matrix on the left side and use a sparse solver such as Cholesky for a large number of elements.

The solution to this system is illustrated in Fig. 9.

Notice how increasing the number of elements, increases the accuracy of the solution.

3.2 Elasticity

Elasticity is the ability of an object to keep its original shape under external pressure. The most basic mathematical definition of elasticity is given by Hooke's Law, which establishes a linear relationship between the deformation of a body and the forces acting on it. It is mostly known for describing the behavior of springs with the expression

$$F = -k\Delta x, \tag{64}$$

where F is the force acting on the spring, Δx is the deformation suffered by the spring and k is a stiffness constant. See that for $\Delta x = 0$ the force vanishes as there is no deformation. We can say then that the force on a spring depends on a deformed configuration x and some undeformed configuration X unique to a spring such that $F = -k(x - X)$.

The stiffness constant depends on both the length and the material of the spring. For instance, if we take two springs of different lengths but same material, and extend each one by 10%, we should expect to have the same resulting force for both springs. However, this does not hold if both springs have the same stiffness constant. A better way of representing the stiffness of the spring is by replacing the constant with the expression $k = \frac{E}{l_0}$, where E is known as the Young modulus, which represents the true stiffness of the material and l_0 is the initial length of the spring. We get the expression

$$F = \frac{E}{l_0}(x - X). \tag{65}$$

There is still a couple of issues with that last expression. First, it assumes the spring does not have a mass and the forces only affect the endpoint of the spring. It is very common to find basic physics examples where a mass is attached to the end of the spring and the student is asked to find the deformation of the spring under gravity, disregarding the mass of the spring. However, it is in our interest to study the deformation of the spring on its entirety as a continuous mechanical body. For that purpose, the field of continuum mechanics exists.

Continuum mechanics

To find the deformation of the massless spring with a mass of attached to its end, we can just compute using $mg = \frac{E}{l_0}\Delta x$, where m is the mass of the attached object and g is a constant representing gravity. Now, if we change the problem and say we want to find the deformation of a spring with total mass equal to that of the attached object and let it hang by itself under gravity. We need a different expression, which is given in the form of a known

ODE

$$-E\Delta u = -E\frac{\partial^2 u}{\partial X^2} = b, \tag{66}$$

where E is the Young modulus (keeping in mind we are still in 1D), u is a displacement field ($x - X$ given that x and X are continuous quantities) and b is a force field. So, in continuum mechanics, Hooke's law of elasticity takes the form of a Poisson equation multiplied by a constant.

Now, if we want to find the solution to our problem, we have gravity as the force field and we get the expression

$$E\Delta u = \rho g. \tag{67}$$

Notice that since we are dealing with infinitesimal values, instead of considering the total mass of the spring, we consider the density at a point of the spring ρ.

Let's now compare the result of a massless spring with an attached mass of $m = 1$, with that of a heavy spring (Essen and Nordmark [19]) with a total mass of $m = 1$ (uniform density), both deformed under gravity and with $E = 1$ and length $l_0 = 1$.

For the first case, we get simply $\Delta x = g$, meaning the total deformation is equal to the gravity. In the second case, solving the ODE gives $u(X) = \rho g l_0 X - \frac{\rho g X^2}{2}|_0^{l_0} = \frac{g}{2}$ (enforcing the boundary condition $\frac{du}{dX}(l_0) = 0$), which is half as much as the spring with the mass attached.

Strain

The second issue is how deformation is measured. At first glance, $x - X$ may seem a logical choice, but consider what would happen if we want to translate the deformed spring to a different point in space. Applying the current formula would count that translation to be part of the deformation. This is of course unwanted behavior. A better choice consists on expressing the deformed configuration as a function of the undeformed configuration $x(X)$ and let the deformation be $\epsilon = \frac{\partial x}{\partial X} - 1$, or in terms of the displacements as $\epsilon = \frac{\partial u}{\partial X}$. This newly introduced value ϵ is known as strain, which quantifies the deformation of a body. The quantity $\frac{\partial x}{\partial X}$ is known as the deformation gradient.

If we move to higher dimensions, we find new kinds of deformations that are not the result of elongation nor compression. For example, in 2D we may change the shape of an object in a way that its area does not change. Such deformations are known as shear strain. A material that only experiences shear deformation without affecting its area or volume is said to be incompressible.

In addition to shear deformations, it can be observed that elongation can happen in different directions. For this reason strain in higher dimensions is expressed as a tensor rather than a value.

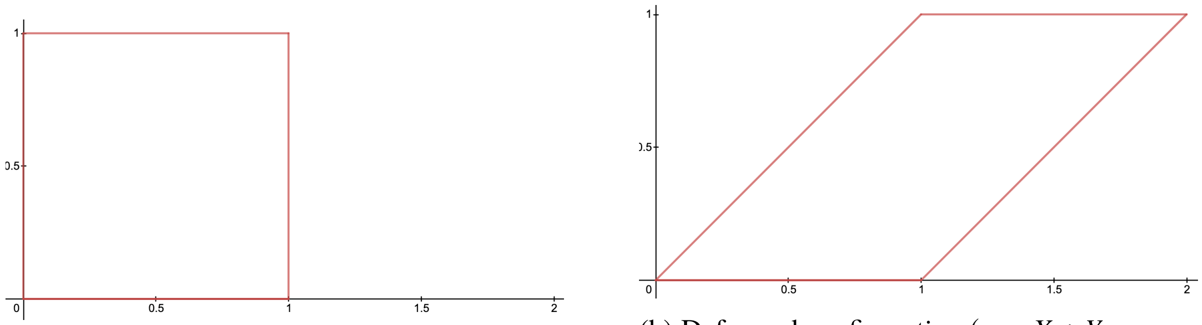
In 2D the strain tensor can be expressed in two ways, as a 2×2 matrix or as a 3-dimensional vector

$$\boldsymbol{\epsilon} = \begin{bmatrix} \epsilon_{xx} & \epsilon_{xy} \\ \epsilon_{yx} & \epsilon_{yy} \end{bmatrix}, \text{ or } \boldsymbol{\epsilon} = \begin{bmatrix} \epsilon_{xx} \\ \epsilon_{yy} \\ 2\epsilon_{xy} \end{bmatrix}, \tag{68}$$

where ϵ_{xx} and ϵ_{yy} are the directional strains, and ϵ_{xy} and ϵ_{yx} are the shear strains.

Observe that the tensor assumes that $\epsilon_{xy} = \epsilon_{yx}$ making the tensor symmetric. This is a requirement to guarantee force equilibrium.

It can also be observed that having the strain be $\boldsymbol{\epsilon} = \frac{\partial \boldsymbol{u}}{\partial \boldsymbol{X}}$ does not satisfy the symmetry requirement. In order to have a symmetric tensor, we may use Cauchy's strain tensor,


(a) Undeformed configuration (X, Y).

(b) Deformed configuration $(x = X + Y, y = Y)$.

Figure 10: A 2-dimensional square experiencing simple shear deformation.

defined as

$$\boldsymbol{\epsilon} = \frac{1}{2}\left(\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{X}} + \frac{\partial \boldsymbol{u}}{\partial \boldsymbol{X}}^T\right), \tag{69}$$

which is a linear truncation of the quadratic strain tensor whose properties will be exposed in section 3.2.2.

For example consider the case of a square under simple shear deformation $(x = X+Y, y = Y)$ as illustrated in Fig. 10. If we compute the strain as $\boldsymbol{\epsilon} = \frac{\partial \boldsymbol{u}}{\partial \boldsymbol{X}} = \frac{\partial \boldsymbol{x}}{\partial \boldsymbol{X}} - \boldsymbol{I}_2$, where $\boldsymbol{I}_2$ is the 2×2 identity matrix, we obtain the tensor

$$\boldsymbol{\epsilon} = \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix}, \tag{70}$$

which properly captures the shear deformation but is not symmetric. Using the linear Cauchy's strain tensor we get

$$\boldsymbol{\epsilon} = \begin{bmatrix} 0 & \frac{1}{2} \\ \frac{1}{2} & 0 \end{bmatrix}, \tag{71}$$

which captures shear deformation and is symmetric.

In 3D, the tensor takes the form of a 3×3 matrix or a 6-dimensional vector

$$\boldsymbol{\epsilon} = \begin{bmatrix} \epsilon_{xx} & \epsilon_{xy} & \epsilon_{xz} \\ \epsilon_{yx} & \epsilon_{yy} & \epsilon_{yz} \\ \epsilon_{zx} & \epsilon_{zy} & \epsilon_{zz} \end{bmatrix}, \text{ or } \boldsymbol{\epsilon} = \begin{bmatrix} \epsilon_{xx} \\ \epsilon_{yy} \\ \epsilon_{zz} \\ 2\epsilon_{xy} \\ 2\epsilon_{xz} \\ 2\epsilon_{yz} \end{bmatrix}, \tag{72}$$

where we have the directional strains ϵ_{xx}, ϵ_{yy}, ϵ_{zz} and the shear strains ϵ_{xy}, ϵ_{xz}, ϵ_{yz}. The tensor is also symmetric in this case as it is in 2D and therefore $\epsilon_{xy} = \epsilon_{yx}, \epsilon_{xz} = \epsilon_{zx}, \epsilon_{yz} = \epsilon_{zy}$.

Stress

While strain quantifies the deformation, stress is the force generated by such deformation and is represented by the Greek letter σ.

We saw that Hooke's Law establishes a linear relationship between stress and strain where the rate of change is the Young modulus. Under the new terminology, Hooke's Law is defined as

$$\sigma = E\epsilon, \tag{73}$$

where E is the Young modulus.

Moving to 2D, we have to introduce the concept of anisotropy. A material may experience forces of different magnitudes for the same stretch applied in different directions. Consider for example woven fabric; The material is stiffer in the direction the fabric was woven.

Isotropic materials define the simplest form of anisotropy. They encompass all the materials that react with the same force for the same stretch regardless of the direction. So, for example, if we have the two directional stresses $\sigma_{xx} = E_x \epsilon_{xx}$ and $\sigma_{yy} = E_y \epsilon_{yy}$, it is clear that for an isotropic material $E_x = E_y$.

Another behavior to be considered is the effect of stretching in one direction has in the other directions. Many materials, when stretched tend to compensate by compressing in the other directions.

This behavior is captured mathematically by the Poisson's ratio ν, which takes values between -1 and $\frac{1}{2}$. A negative value means that if a material is compressed in one direction, it will also compress in the other directions; this is common in certain types of foams. If it takes a positive value, tensile forces are experienced in the other directions for a given compressive force in one direction. Finally, there is the special case of $\frac{1}{2}$ which means the material is completely incompressible. If the material is compressed in one direction it will stretch in the other directions in a way that the object conserves its volume.

In vector form, Hooke's Law for isotropic materials in $2D$ is expressed as

$$\boldsymbol{\sigma} = \frac{E}{1-\nu^2}\begin{bmatrix} 1 & -\nu & 0 \\ -\nu & 1 & 0 \\ 0 & 0 & \frac{1-\nu}{2} \end{bmatrix}\begin{bmatrix} \epsilon_{xx} \\ \epsilon_{yy} \\ 2\epsilon_{xy} \end{bmatrix}. \tag{74}$$

As is the case with strain, the stress tensor is also a symmetric matrix.

In 3D, Hooke's Law for isotropic materials is given by

$$\boldsymbol{\sigma} = \frac{E}{(1+\nu)(1-2\nu)}\begin{bmatrix} 1-\nu & \nu & \nu & 0 & 0 & 0 \\ \nu & 1-\nu & \nu & 0 & 0 & 0 \\ \nu & \nu & 1-\nu & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{2}-\nu & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{2}-\nu & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{2}-\nu \end{bmatrix}\begin{bmatrix} \epsilon_{xx} \\ \epsilon_{yy} \\ \epsilon_{zz} \\ 2\epsilon_{xy} \\ 2\epsilon_{xz} \\ 2\epsilon_{yz} \end{bmatrix}. \tag{75}$$

For orthotropic and anisotropic cases a corresponding 3×3 material matrix for 2D and a 6×6 material for 3D can be formulated with their corresponding material parameters given the level of anisotropy. For simplicity, we only consider isotropic materials for Hooke's Law.

3.2.1 Linear Elasticity Weak Form

In section 3.1.4 we saw how to obtain the weak form for the Poisson PDE. Given that the continuous version of Hooke's Law in 1D is a Poisson PDE multiplied by a constant E, its weak force derivation is straightforward and expressed for a single segment by

$$\int_\Omega \boldsymbol{B}^T E \boldsymbol{B}\hat{\boldsymbol{u}} = \int_\Omega \boldsymbol{N} b, \tag{76}$$

where $\hat{\boldsymbol{u}}$ are the weights representing the discrete node displacements, $\boldsymbol{N}$ are the shape operators interpolating the displacements ($u(X) \approx \sum_i N_i(X)\hat{u}_i$), $\boldsymbol{B}$ is the matrix of derivatives of the shape functions with respect to the undeformed configuration X, E is the Young modulus and b are the continuous forces.

To interpolate the displacements in 2D, we can use the linear triangle. Remember that a linear triangle has 3 interpolation nodes. We need to formulate our strain in a way that is compatible with our material matrix $\boldsymbol{E}$ of size 3×3.

We interpolate the displacement through nodes using the expressions $u(X) \approx N_1\hat{u}_{1x} + N_2\hat{u}_{2x} + (1-N_1-N_2)\hat{u}_{3x}$ and $u(Y) \approx N_1\hat{u}_{1y} + N_2\hat{u}_{2y} + (1-N_1-N_2)\hat{u}_{3y}$ where $\boldsymbol{u}_1$, $\boldsymbol{u}_2$ and $\boldsymbol{u}_3$ are the displacements experienced by nodes 1, 2 and 3 respectively. Using Cauchy's linear strain tensor $\left[\boldsymbol{\epsilon} = \frac{1}{2}\left(\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{X}} + \frac{\partial \boldsymbol{u}}{\partial \boldsymbol{X}}^T\right)\right]$, we obtain

$$\epsilon_{xx} = \frac{\partial u_x}{\partial X} = \begin{bmatrix} \frac{\partial N_1}{\partial X} & \frac{\partial N_2}{\partial X} & -\frac{\partial N_1}{\partial X} - \frac{\partial N_2}{\partial X} \end{bmatrix} \begin{bmatrix} \hat{u}_{1x} \\ \hat{u}_{2x} \\ \hat{u}_{3x} \end{bmatrix}, \tag{77}$$

$$\epsilon_{yy} = \frac{\partial u_y}{\partial Y} = \begin{bmatrix} \frac{\partial N_1}{\partial Y} & \frac{\partial N_2}{\partial Y} & -\frac{\partial N_1}{\partial Y} - \frac{\partial N_2}{\partial Y} \end{bmatrix} \begin{bmatrix} \hat{u}_{1y} \\ \hat{u}_{2y} \\ \hat{u}_{3y} \end{bmatrix}, \tag{78}$$

and

$$\begin{aligned} \epsilon_{xy} = \frac{1}{2}\left(\frac{\partial u_x}{\partial Y} + \frac{\partial u_y}{\partial X}\right) = \frac{1}{2} &\begin{bmatrix} \frac{\partial N_1}{\partial Y} & \frac{\partial N_2}{\partial Y} & -\frac{\partial N_1}{\partial Y} - \frac{\partial N_2}{\partial Y} \end{bmatrix} \begin{bmatrix} \hat{u}_{1x} \\ \hat{u}_{2x} \\ \hat{u}_{3x} \end{bmatrix} + \\ &\begin{bmatrix} \frac{\partial N_1}{\partial X} & \frac{\partial N_2}{\partial X} & -\frac{\partial N_1}{\partial X} - \frac{\partial N_2}{\partial X} \end{bmatrix} \begin{bmatrix} \hat{u}_{1y} \\ \hat{u}_{2y} \\ \hat{u}_{3y} \end{bmatrix} \end{aligned} \tag{79}$$

We can rearrange these values as a single system as

$$\boldsymbol{\epsilon} = \boldsymbol{B}\hat{\boldsymbol{u}} = \begin{bmatrix} \frac{\partial N_1}{\partial X} & 0 & \frac{\partial N_2}{\partial X} & 0 & -\frac{\partial N_1}{\partial X} - \frac{\partial N_2}{\partial X} & 0 \\ 0 & \frac{\partial N_1}{\partial Y} & 0 & \frac{\partial N_2}{\partial Y} & 0 & -\frac{\partial N_1}{\partial Y} - \frac{\partial N_2}{\partial Y} \\ \frac{\partial N_1}{\partial Y} & \frac{\partial N_1}{\partial X} & \frac{\partial N_2}{\partial Y} & \frac{\partial N_2}{\partial X} & -\frac{\partial N_1}{\partial Y} - \frac{\partial N_2}{\partial Y} & -\frac{\partial N_1}{\partial X} - \frac{\partial N_2}{\partial X} \end{bmatrix} \begin{bmatrix} \hat{u}_{1x} \\ \hat{u}_{1y} \\ \hat{u}_{2x} \\ \hat{u}_{2y} \\ \hat{u}_{3x} \\ \hat{u}_{3y} \end{bmatrix}. \tag{80}$$

With which we can get the stress vector using the material matrix $\boldsymbol{E}$.

$$\boldsymbol{\sigma} = \boldsymbol{E}\boldsymbol{\epsilon} = \boldsymbol{E}\boldsymbol{B}\hat{\boldsymbol{u}} \tag{81}$$

The right part of the equation ($\int_\Omega \boldsymbol{N}b$) is computed as usual. Remember that the shape functions are linear on X and Y, which means that the strain is constant over the element.

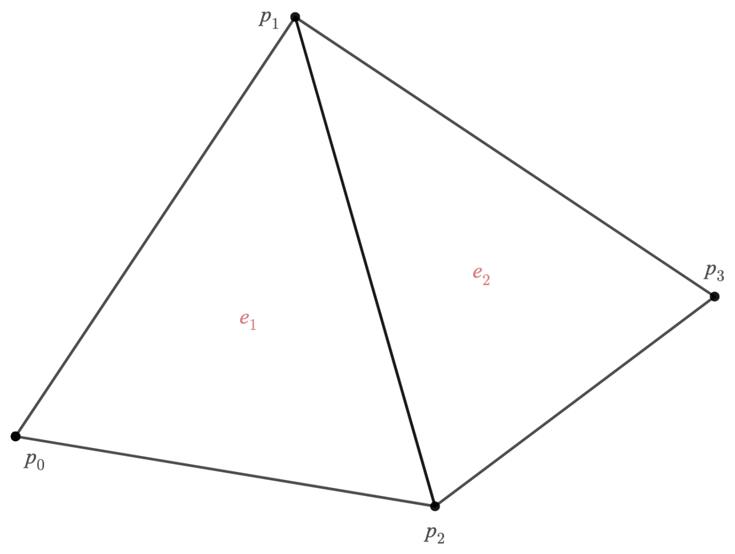


Figure 11: Mesh with two linear triangles sharing one edge.

With this information, and knowing that $\int_\Omega 1 = A$, where A is the area of the undeformed element, we have

$$A\boldsymbol{B}^T\boldsymbol{E}\boldsymbol{B}\hat{\boldsymbol{u}} = \int_\Omega \boldsymbol{N} b. \tag{82}$$

The resulting 6×6 matrix $\boldsymbol{B}^T\boldsymbol{E}\boldsymbol{B}$ is also known as the stiffness matrix (similar definition to the stiffness coefficient k for springs) and is represented by the symbol $\boldsymbol{K}$.

A single node may belong to multiple elements since they share edges. Every contribution of every individual element is added to a global stiffness matrix for every corresponding shared node.

For example, consider the mesh with two elements in Fig. 11. It contains a total of 4 nodes with the elements e_1, e_2 sharing a common edge $\boldsymbol{p}_1, \boldsymbol{p}_2$. Consider the two stiffness matrices K_1 and K_2 for element q and 2 respectively. The matrix K_1 has the nodes $\boldsymbol{p}_0, \boldsymbol{p}_1, \boldsymbol{p}_2$ and K_2 the nodes $\boldsymbol{p}_2, \boldsymbol{p}_3, \boldsymbol{p}_1$.

If we were to divide each stiffness matrix in 9 blocks of 2×2 for each pair of nodes they represent, we would get

$$\boldsymbol{K}_e\hat{\boldsymbol{u}}_e = \begin{bmatrix} \boldsymbol{K}_{p_0,p_0} & \boldsymbol{K}_{p_0,p_1} & \boldsymbol{K}_{p_0,p_2} \\ \boldsymbol{K}_{p_1,p_0} & \boldsymbol{K}_{p_1,p_1} & \boldsymbol{K}_{p_1,p_2} \\ \boldsymbol{K}_{p_2,p_0} & \boldsymbol{K}_{p_2,p_1} & \boldsymbol{K}_{p_2,p_2} \end{bmatrix} \begin{bmatrix} \hat{\boldsymbol{u}}_{p_1} \\ \hat{\boldsymbol{u}}_{p_2} \\ \hat{\boldsymbol{u}}_{p_3} \end{bmatrix}. \tag{83}$$

And adding all contributions, we get a global stiffness matrix of $8x8$ (2 times the number of nodes) with the structure

$$\boldsymbol{K}\hat{\boldsymbol{u}} = \begin{bmatrix} \boldsymbol{K}_{e_1,p_0,p_0} & \boldsymbol{K}_{e_1,p_0,p_1} & \boldsymbol{K}_{e_1,p_0,p_2} & 0 \\ \boldsymbol{K}_{e_1,p_1,p_0} & \boldsymbol{K}_{e_1,p_1,p_1} + \boldsymbol{K}_{e_2,p_1,p_1} & \boldsymbol{K}_{e_1,p_1,p_2} + \boldsymbol{K}_{e_2,p_1,p_2} & \boldsymbol{K}_{e_2,p_1,p_3} \\ \boldsymbol{K}_{e_1,p_2,p_0} & \boldsymbol{K}_{e_1,p_2,p_1} + \boldsymbol{K}_{e_2,p_2,p_1} & \boldsymbol{K}_{e_1,p_2,p_2} + \boldsymbol{K}_{e_2,p_2,p_2} & \boldsymbol{K}_{e_2,p_2,p_3} \\ 0 & \boldsymbol{K}_{e_2,p_3,p_1} & \boldsymbol{K}_{e_2,p_3,p_2} & \boldsymbol{K}_{e_2,p_3,p_3} \end{bmatrix} \begin{bmatrix} \hat{\boldsymbol{u}}_{p_0} \\ \hat{\boldsymbol{u}}_{p_1} \\ \hat{\boldsymbol{u}}_{p_2} \\ \hat{\boldsymbol{u}}_{p_3} \end{bmatrix}. \tag{84}$$

Notice how the blocks that reference only $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$ have each two contributions, while a block containing $\boldsymbol{p}_0$ and p_3 has only one contribution or 0 if they are mutually related, since element-wise there is no relationship between $\boldsymbol{p}_0$ and $\boldsymbol{p}_3$.

This structure can be expanded to a mesh with multiple elements. If the mesh is highly regular it can be seen that each node is shared on average by 6 elements. This means the stiffness matrix is sparse as each row, regardless of size, has on average contributions on 6 blocks of 2×2.

Moving on to 3D, we have linear interpolation between the 4 nodes of a tetrahedron. This gives a stiffness matrix of 12×12 (4 nodes x 3 dimensions). With the same process we used to obtain the strain for the linear triangle, we can get the strain vector for a linear tetrahedron which is defined as

$$\boldsymbol{B\hat{u}} = \begin{bmatrix} a_1 & 0 & 0 & a_2 & 0 & 0 & a_3 & 0 & 0 & a_4 & 0 & 0 \\ 0 & b_1 & 0 & 0 & b_2 & 0 & 0 & b_3 & 0 & 0 & b_4 & 0 \\ 0 & 0 & c_1 & 0 & 0 & c_2 & 0 & 0 & c_3 & 0 & 0 & c_4 \\ b_1 & a_1 & 0 & b_2 & a_2 & 0 & b_3 & a_3 & 0 & b_4 & a_4 & 0 \\ 0 & c_1 & b_1 & 0 & c_2 & b_2 & 0 & c_3 & b_3 & 0 & c_4 & b_4 \\ c_1 & 0 & a_1 & c_2 & 0 & a_2 & c_3 & 0 & a_3 & c_4 & 0 & a_4 \end{bmatrix} \begin{bmatrix} \hat{u}_{1x} \\ \hat{u}_{1y} \\ \hat{u}_{1z} \\ \hat{u}_{2x} \\ \hat{u}_{2y} \\ \hat{u}_{2z} \\ \hat{u}_{3x} \\ \hat{u}_{3y} \\ \hat{u}_{3z} \\ \hat{u}_{4x} \\ \hat{u}_{4y} \\ \hat{u}_{4z} \end{bmatrix}, \tag{85}$$

where

$$a_1 = \frac{\partial N_1}{\partial X}, a_2 = \frac{\partial N_2}{\partial X}, a_3 = \frac{\partial N_3}{\partial X}, a_4 = -\frac{\partial N_1}{\partial X} - \frac{\partial N_2}{\partial X} - \frac{\partial N_3}{\partial X}, \tag{86}$$

$$b_1 = \frac{\partial N_1}{\partial Y}, b_2 = \frac{\partial N_2}{\partial Y}, b_3 = \frac{\partial N_3}{\partial Y}, b_4 = -\frac{\partial N_1}{\partial Y} - \frac{\partial N_2}{\partial Y} - \frac{\partial N_3}{\partial Y}, \tag{87}$$

and

$$c_1 = \frac{\partial N_1}{\partial Z}, c_2 = \frac{\partial N_2}{\partial Z}, c_3 = \frac{\partial N_3}{\partial Z}, c_4 = -\frac{\partial N_1}{\partial Z} - \frac{\partial N_2}{\partial Z} - \frac{\partial N_3}{\partial Z}. \tag{88}$$

As was the case with the linear triangle, the left side of the equation is also constant on the domain for the linear tetrahedron, only in this case we are not interpolating area but volume. Our expression for the linear tetrahedron then becomes:

$$V\boldsymbol{B}^T\boldsymbol{E}\boldsymbol{B}\hat{\boldsymbol{u}} = \int_\Omega \boldsymbol{N}b \tag{89}$$

where V is the volume of the tetrahedron in the undeformed configuration. The resulting expression is of the form $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ where $\boldsymbol{A} = \boldsymbol{K}$, where the matrix is symmetric. If appropriate boundary conditions are enforced, the matrix is also non-singular. Since in this particular case $\boldsymbol{K}$ does not depend on the weights $\hat{\boldsymbol{u}}$, it can be solved in a single step.

In the case of shape functions of higher-order or non-linear stress-strain relationships, the system can be solved iteratively using Newton-Rhapson or similar.

3.2.2 Strain Measures

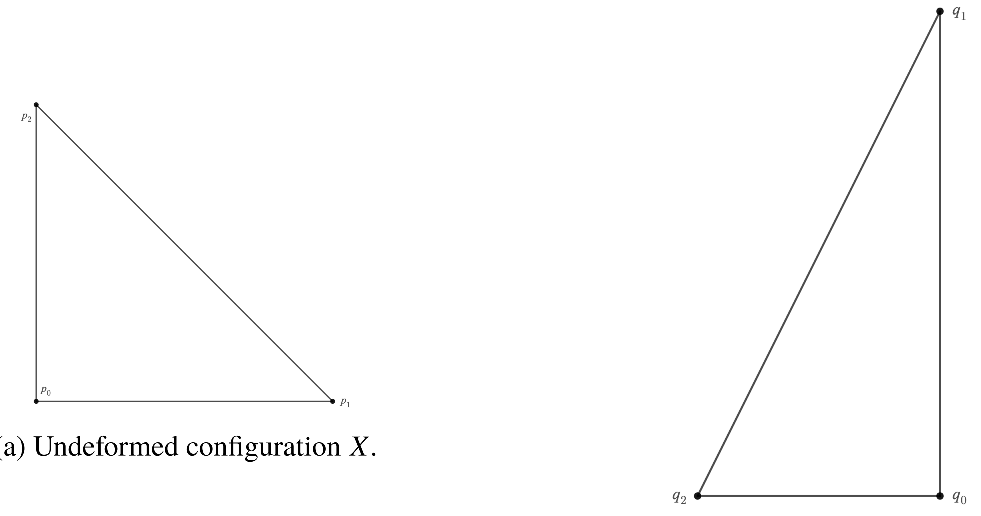
In section 3.2.1, we focused on Cauchy's linear strain tensor to obtain a weak formulation of the elastic PDE. This strain tensor also receives the name of infinitesimal strain tensor for the reason that it is only accurate for very small displacements.

Consider the relationship between the undeformed configuration of an object X and its deformed configuration x. If we use linear shape functions, we obtain the expression

$$x = \boldsymbol{F}X, \tag{90}$$

where $\boldsymbol{F}$ is the deformation gradient. This serves to illustrate that the relationship between the configurations is just a linear function, which may include affine transformations such as rotations, shearing, and scaling. The last two are clear transformations we want to consider as deformations, but rotation is not one of them. This leads to the main problem with the infinitesimal strain tensor; it considers rotations to be deformations.

For example, consider the deformation given in Fig. 12. With the undeformed nodes being $\boldsymbol{p}_0 = (0,0), \boldsymbol{p}_1 = (1,0), \boldsymbol{p}_2 = (0,1)$ and the deformed nodes $\boldsymbol{q}_0 = (0,0), \boldsymbol{q}_1 =$



(a) Undeformed configuration X.

(b) Deformed configuration x.

Figure 12: A linear triangle where the deformed configuration is a rotation of 90° counter-clockwise.

$(0, 2)$, $\boldsymbol{q}_2 = (-1, 0)$ we obtain the deformation gradient

$$\boldsymbol{F} = \begin{bmatrix} \cos 90^\circ & -\sin 90^\circ \\ 2\sin 90^\circ & \cos 90^\circ \end{bmatrix} = \begin{bmatrix} 0 & -1 \\ 2 & 0 \end{bmatrix}, \tag{91}$$

which is clearly the rotation matrix with the node $\boldsymbol{p}_1$ elongated in one direction. If we compute the infinitesimal strain tensor from this deformation gradient we get

$$\boldsymbol{F} = \begin{bmatrix} -1 & \frac{1}{2} \\ \frac{1}{2} & -1 \end{bmatrix}, \tag{92}$$

which is inaccurate. The strains ϵ_{xx} and ϵ_{yy} are negative and we also obtain non-zero shear stress. The expected strain is $\epsilon_{xx} = 1$ and 0 everywhere else.

The question is then, why is the infinitesimal strain tensor useful, to which the answer is simplicity. Since this tensor is linear, the solution for the displacement weights can be obtained through a single linear system, therefore it is commonly used in structural mechanics where deformations are very tiny, such as loading on heavy metals. For materials that undergo heavy deformations, finite strain tensors are used instead.

To analyze finite strain tensors it is often useful to separate rotations from the deformation gradient. This gives the expression

$$x = \boldsymbol{F} x = \boldsymbol{R}\boldsymbol{\delta} X, \tag{93}$$

where $\boldsymbol{R}$ is a rotation matrix and $\boldsymbol{\delta}$ is the actual deformation.

Corotated strain

One way to extract this rotation is by using SVD, which is a method for decomposing a matrix into its singular values and eigenvectors such that

$$\boldsymbol{F} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T, \tag{94}$$

where $\boldsymbol{U}$ and $\boldsymbol{V}$ are the normalized eigenvectors of $\boldsymbol{F}$ in columns and $\boldsymbol{\Sigma}$ is a matrix where the diagonal are the eigenvalues of $\boldsymbol{F}$. The rigid rotation of $\boldsymbol{F}$ is then defined as

$$\boldsymbol{R} = \boldsymbol{V}\boldsymbol{U}^T. \tag{95}$$

With this information we can compute a modified deformation gradient which has no rotation by multiplying the inverse of the rotation matrix

$$\boldsymbol{F}^* = \boldsymbol{R}^T\boldsymbol{F}, \tag{96}$$

and compute the linear strain tensor with the new deformation gradient $\boldsymbol{F}^*$. The resulting tensor is known as the co-rotated strain tensor.

In the example above, the rotation matrix is given by

$$\boldsymbol{R} = \begin{bmatrix} \cos 90° & -\sin 90° \\ \sin 90° & \cos 90° \end{bmatrix} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}, \tag{97}$$

which gives the co-rotated deformation gradient

$$\boldsymbol{F}^* = \boldsymbol{R}^T \boldsymbol{F} = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix}, \tag{98}$$

which leads to the wanted strain values.

Green-Lagrange strain tensor

While the co-rotated strain tensor is accurate for finite strains and maintains a linear relationship between strain and deformation, it is very expensive to compute.

For cases of finite deformations where we are willing to sacrifice the linear relationship between strain and deformation for computational speed, we can resort to the Green-Lagrange strain tensor. It is defined as

$$\boldsymbol{\epsilon} = \frac{1}{2}\left(\boldsymbol{F}^T \boldsymbol{F} - \boldsymbol{I}\right). \tag{99}$$

It exploits the fact that rotations of a matrix are eliminated when it is squared, which can be easily proven.

$$\boldsymbol{F}^T \boldsymbol{F} = (\boldsymbol{R}\boldsymbol{\delta})^T (\boldsymbol{R}\boldsymbol{\delta}) = \boldsymbol{\delta}^T \boldsymbol{R}^T \boldsymbol{R} \boldsymbol{\delta} = \boldsymbol{\delta}^T \boldsymbol{\delta}. \tag{100}$$

The clear drawback is that the deformation terms are now squared, which is usually acceptable for common material models.

If we compute the Green-Lagrange strain from the example above we can see that

$$\boldsymbol{F}^T\boldsymbol{F} = \begin{bmatrix} 4 & 0 \\ 0 & 1 \end{bmatrix}, \tag{101}$$

which effectively gets rid of the rotation and the strain tensor:

$$\boldsymbol{\epsilon} = \begin{bmatrix} \frac{3}{2} & 0 \\ 0 & 0 \end{bmatrix}, \tag{102}$$

which is not exactly what is expected for ϵ_{xx} assuming a linear strain-deformation relationship but is acceptable for more complex relationships.

3.2.3 Hyper-Elasticity

Until this section, we have only considered materials with linear stress-strain relationships with the elastic weak form. For problems with complex stress-strain relationships, the use of a hyper-elastic constitutive model is preferred. The idea is to derive the stress-strain relationship from a scalar strain energy density function (Holzapfel [25]) per element

$$\psi = \int \sigma d\epsilon. \tag{103}$$

For Hooke's Law, this expression simplifies to:

$$\psi = \frac{1}{2}\epsilon^T\sigma. \tag{104}$$

The total energy of the element is evaluated as the integral of the strain density over the domain Ω as

$$U = \int_\Omega \psi dA. \tag{105}$$

With this formulation, the weak form of elasticity can be solved as an energy minimization problem as opposed to solving a system of linear equations. The nodal force is computed as the derivative of the energy with respect to the deformed nodal vertices $\hat{\boldsymbol{q}}$ as

$$\boldsymbol{f} = \frac{\partial U}{\partial \hat{\boldsymbol{q}}}. \tag{106}$$

Using the weak form notation, Hooke's Law hyper-elastic material using linear elements can be expressed as

$$U = \frac{1}{2}\Omega\epsilon^T\sigma = \frac{1}{2}\Omega\hat{\boldsymbol{u}}^T\boldsymbol{B^T}\boldsymbol{E}\boldsymbol{B}\hat{\boldsymbol{u}}, \tag{107}$$

where Ω is the area or volume of the element in case we are using 2D or 3D elements.

If we obtain the forces from Eq. 107 for the 2D case, the weak form of Eq. 82 is obtained. That is to say,

$$\boldsymbol{f} = \frac{\partial U}{\partial \hat{\boldsymbol{q}}} = A\boldsymbol{B^T}\boldsymbol{E}\boldsymbol{B}\hat{\boldsymbol{u}}, \tag{108}$$

keeping in mind that the nodal displacements are computed as $\hat{\boldsymbol{u}} = \hat{\boldsymbol{q}} - \hat{\boldsymbol{p}}$.

Hooke's Law can also be expressed in terms of the strain tensor in what is known as the Saint-Venant Kirchhoff (StVK) hyper-elastic material, whose strain energy density function is defined as

$$\psi = \frac{1}{2}\lambda\mathrm{tr}(\boldsymbol{\epsilon})^2 + \mu\mathrm{tr}(\boldsymbol{\epsilon}^2), \tag{109}$$

where $\boldsymbol{\epsilon}$ is a strain tensor (e.g. Green-Lagrange tensor). This last equation introduces two new material parameters λ and μ. They are known as the Lame parameters. A direct relationship exists between these parameters and the known Young and Poisson's ratio

(E, ν). This relationship is expressed as

$$\begin{aligned} \lambda &= \tfrac{E\nu}{(1+\nu)(1-2\nu)} \\ \mu &= \tfrac{E}{2(1+\nu)} \end{aligned}\ . \tag{110}$$

3.2.4 Stress Measures

With the definition of the hyper-elastic energy, we saw that we could compute the nodal forces as the derivative of the strain energy. However, we can also compute per-element measures that help analyze internal stresses. These measures are expressed in the form of tensors. Given a direction vector in a given frame of reference, the stress tensor relates a traction vector in a certain frame of reference to the given direction.

The most commonly used measure is the Cauchy stress tensor. This tensor relates tractions in the deformed configuration to a direction also given in the deformed configuration. Since we are usually interested in analyzing stresses in the deformed configuration, this tensor is also known as true stress $\boldsymbol{\sigma}$. This tensor is derived from the strain energy density function with respect to the deformation gradient as

$$\boldsymbol{\sigma} = \frac{1}{J}\frac{\partial \psi}{\partial \boldsymbol{F}}\boldsymbol{F}^T, \tag{111}$$

where $J = \det(\boldsymbol{F})$. This tensor is equivalent to the one defined in Eq. 81 for Hooke's Law.

Other stress measures exist, such as the First Piola-Kirchhoff $\boldsymbol{P}$ stress tensor, which relates a traction in the deformed configuration to a direction in the undeformed configuration as

$$\boldsymbol{P} = \frac{\partial \psi}{\partial \boldsymbol{F}}, \tag{112}$$

or the Second Piola-Kirchhoff stress tensor $\boldsymbol{S}$, which relates tractions in the undeformed configuration to directions in the undeformed configuration as

$$\boldsymbol{S} = \boldsymbol{F}^{-1}\frac{\partial \psi}{\partial \boldsymbol{F}}. \tag{113}$$

A direction or a traction can be brought to the undeformed or deformed configurations using the deformation gradient. For example, the expression

$$\boldsymbol{d}_d = \boldsymbol{F}\boldsymbol{d}_u \tag{114}$$

brings a direction in the undeformed configuration $\boldsymbol{d}_u$ to the deformed configuration $\boldsymbol{d}_d$.

A traction is computed from the stress tensor and a direction in the right frame of reference as

$$\boldsymbol{\tau}_d = \boldsymbol{\sigma}\boldsymbol{d}_d. \tag{115}$$

As is the case with the direction, the traction can be brought back and forth to the desired configuration. Also, notice that as the traction is a vector, it has a normal and a tangential component to the direction. These components are known as normal and shear tractions.

Chapter 4

Related Work

We divide the related works into four sections: 3D shape decomposition for 2D parameterization, modeling coupled physical systems, garment modeling, and tight-fitting clothing design.

4.1 3D Shape Decomposition for 2D Parameterization

Decomposing an arbitrary 3D shape into 2D patches is a fundamental problem in graphics at the confluence of a large number of practical applications. Early solutions, such as Sander et al. [48], Lévy et al. [29], Sorkine et al. [53], Zhou et al. [69], Julius et al. [26], and Yamauchi et al. [67], use a bottom-up approach to grow quasi-developable regions until a certain developability threshold is reached. While these methods offer little control over the patch boundaries (i.e. seams) or the number of patches to be used, more recent works, such as Poranne et al. [44] and Li et al. [32], overcome these limitations through joint optimization of the seams and distortion of the parameterization. All these methods focus on geometric criteria that are insufficient for fabrication where additional physical reasoning needs to be incorporated in the model.

Quasi-developable object decomposition for fabrication adds additional layers of complexity to the problem: the physical shape obtained through the manufacturing process embedded in the physical world needs to conform to the virtual target shape. Mori et al. [35] developed a system for plush toys, and Skouras et al. [52] developed a system for fabrication of inflatable structures. Sharp et al. [49] frame the problem of mesh decomposition as a 3D boundary optimization over a continuous domain. While each of these methods is effective for the specific application they pursue, none of them extends to the tightly-coupled mechanics of clothing and body that we target with our work.

We add additional information about these applications in the following subsections.

4.1.1 2D Parameterization

The process of cutting a 3D surface and flattening it in a 2D embedding where the cuts become the embedding boundary is known as 2D parameterization, which is often used in applications such as remeshing, texturing among others. These kinds of parameterizations focus on minimizing distortion based on two measures: angles and areas. Parameterizations that minimize the angle difference are called conformal. Given a pre-defined set of cuts, LSCM by Lévy et al. [29] (Fig. 13) can achieve such maps by minimizing an energy function based on the difference in angles between the 3D surface and its map. A common example of a conformal map is the Mercator projection of Earth, which is commonly used for navigational purposes, as paths traced on the map have the same angles as those traced on the globe. In contrast, parameterizations that minimize areas, often do so by dropping the conformal property. In instances where a 2D map can be found such that both areas and angles can be preserved, the surface is said to be developable and their parameterization is said to be isometric. Such surfaces are limited in number (examples include a cube or an open cylinder), and finding a 2D parameterization involves often a compromise between minimizing total angle or area distortion. Methods such as Sorkine et al. [53] and Zhou et

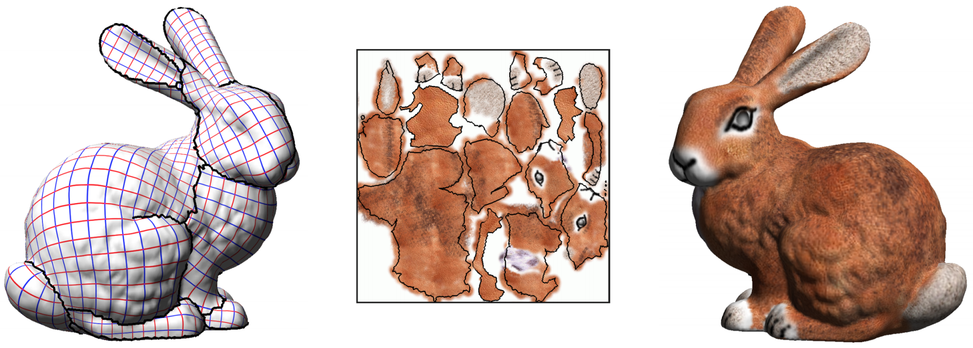


Figure 13: Least-Squares Conformal Maps. Angles traced in these maps are the same as the ones traced on the 3D surface (Left: Parameterization on the 3D surface. Middle: 2D parameterization. Right: Textured 3D surface.). Figure taken from Lévy et al. [29]

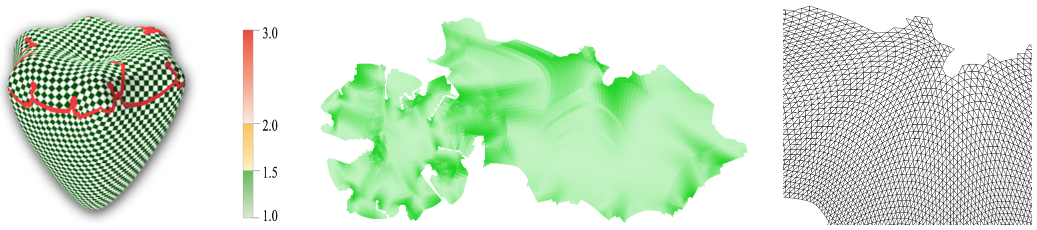


Figure 14: Bounded distortion parameterizations. This method tries to keep distortion below a threshold. (Left: 3D surface with cuts. Center and right: 2D parameterization.) Figure taken from Sorkine et al. [53].

al. [69] try to keep the area distortion to a minimum (Fig. 14).

4.1.2 Mesh Segmentation

In the previous subsection, we focused on methods that found a 2D parameterization based on a pre-defined set of cuts. However, we can turn that around and find a set of cuts, such that when flattened, result in a set of quasi-isometric patterns. Works, such as Julius et al. [26] and Yamauchi et al. [67] focus on this direction. The work done by Dong et al. [18] can generate quasi-isometric rectangular patterns based on a spectral decomposition of the Laplace function of the surface (Fig. 15). These quasi-isometric patterns play a fundamental part in a Eulerian-on-Lagrangian coupling of skin and body done by Li et al. [30] which will be presented in Sec. 4.2.2.

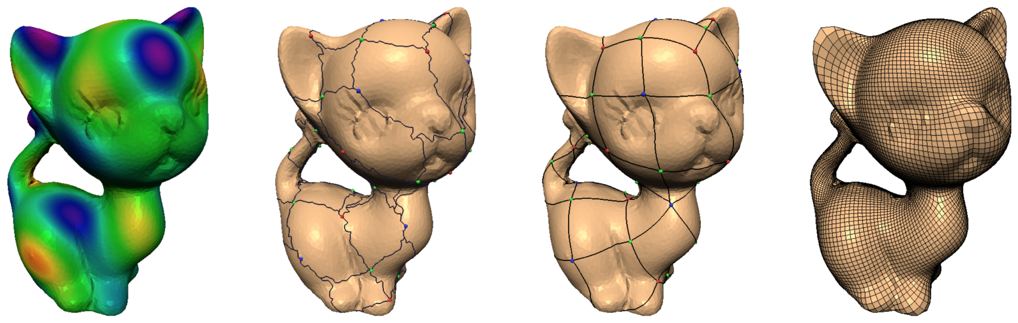


Figure 15: Spectral Surface Decomposition. 3D surface is cut in quasi-developable regions. (Left: Eigenfunctions of the Laplacian, Center left: Morse-Smale Complex of one eigenvector of the Laplacian, Center right: Spectral decomposition, Right: Remeshed surface). Figure taken from Dong et al. [18].

4.1.3 Physically-based 2D Parameterization

While distortion methods generated 2D parameterizations based on geometric constraints, physically-based methods can find 2D parameterizations that satisfy physical constraints on the 3D surface. In Skouras et al. [52], they propose a framework for inflatable balloons to match a desired shape by optimizing a set of 2D patterns. They achieve this by introducing a simulation framework where the patterns are inflated by being exposed to uniform pressure. Changes to the patterns naturally introduce changes to the simulation. Through optimization, they minimize the distance from the simulation to the desired target shape. While they can find generally good patterns for most shapes, they do not prevent self-intersections in the patterns during optimization. In their process, while the seams are not static, the patch layout is pre-defined and cannot be altered.

4.2 Modeling Coupled Physical Systems

We are surrounded by objects that are naturally in close physical contact with each other, with garment-body interaction being a prime example. Methods that simulate independently the objects typically either solve the contact problem in a local domain, such as Baraff et

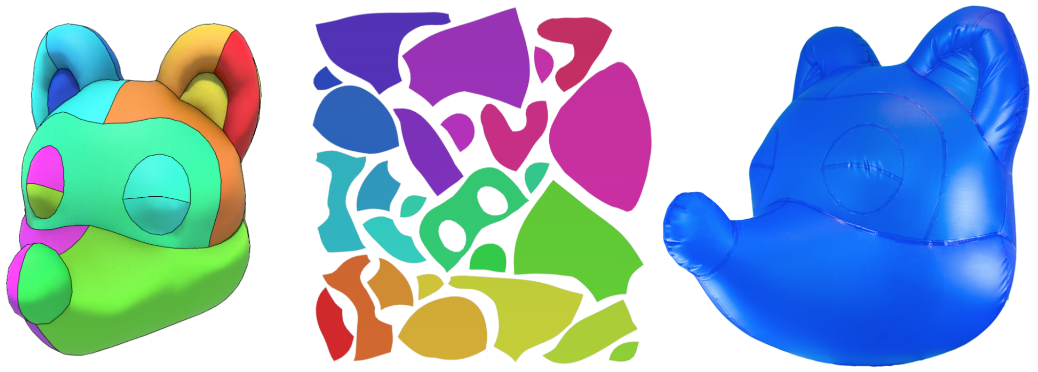


Figure 16: Inflatable Balloons. Framework to design inflatable balloons based on a shape objective.

al. [3] and Volino et al. [59], or use penalty forces, such as Baraff and Witkin [2], Bridson et al. [10], and Harmon et al. [24], or use explicit constraints, such as Otaduy et al. [40], Müller et al. [36], and Li et al. [31]. These models are both complex and cannot handle very well sliding effects between the objects in contact.

Due to these challenges, many proposed solutions use a coupled computational model of the two objects where the contact is modeled explicitly by a reduced model where one object is embedded in the space of the other, such as Sueda et al. [56], Li et al. [30], Fan et al. [20], Cirio et al. [14, 15], and Weidner et al. [66], commonly referred to as Eulerian-on-Lagrangian methods. The main advantage of such a model is that the contact, one of the most complex aspects of coupled physical systems, is implicitly handled.

A natural way to approach this coupling is the Lagrangian view that explicitly embeds one object into another. For instance, in a garment-body coupling, the garment vertices can be represented as points on the 3D body mesh. A fundamental limitation of this approach stems from the piece-wise linear discretization of the models, which creates problems especially when the cloth slides on the body. One solution for this is to use a smooth underlying surface such as a Loop subdivision surface as in Zehnder et al. [68]. Another is to use a Eulerian-on-Lagrangian view for the embedding. The first does not account for two-way elasticity between the coupled systems, and the Eulerian-on-Lagrangian view does not account for cloth with arbitrary 2D patterns. We further elaborate on the Eulerian-on-Lagrangian methods in the following subsections.

4.2.1 Eulerian Methods

In solid mechanics, we normally discretize the object, apply boundary conditions to vertices of the resulting mesh, and solve for a set of equations. This is known as a Lagrangian embedding. However, we could alternatively discretize the space instead, and treat the object as flowing through the resulting mesh. This is known as a Eulerian embedding and it is traditionally used to simulate fluids, where the lack of a clearly defined structure can often discourage the use of Lagrangian embeddings.

In recent years, interest has grown for the use of Eulerian embeddings for solids to handle collisions, such as Levin et al. [28]. From a traditional Lagrangian perspective, detecting colliding objects involve solving expensive node-node, edge-edge, face-face collision queries or combinations of them. From a Eulerian perspective, contact can be easily detected if multiple objects share a single spatial element (Fig. 17). However, applying boundary conditions in Eulerian embeddings can be cumbersome, and simulating thin objects requires dense discretizations of the space. Also, space between objects needs to be discretized as well which often leads to a waste of memory. Some of these problems are addressed in Eulerian-on-Lagrangian embeddings.

4.2.2 Eulerian-on-Lagrangian Methods

Eulerian-on-Lagrangian (EoL) embeddings have seen increasing number of research works in recent years. Fan et al. [20] addressed some of the problems of raw Eulerian solid simulation by embedding the surrounding Eulerian space of an object into a Lagrangian embedding (Fig. 18). Colliding objects then first go through the process of detecting collisions in their Lagrangian embeddings before detecting collisions the Eulerian way, at

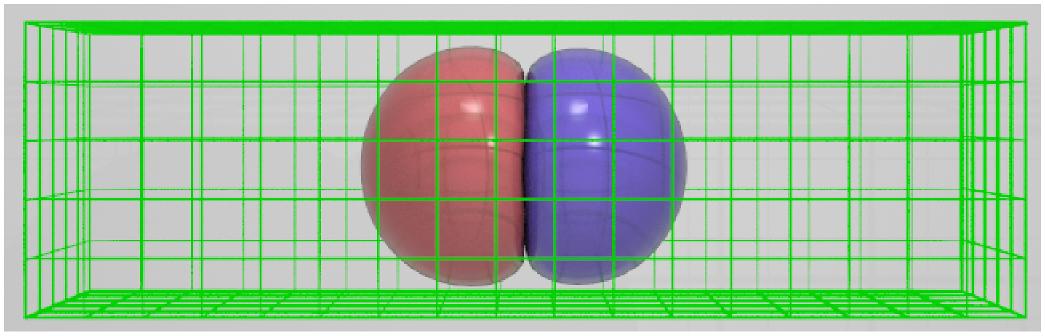


Figure 17: Eulerian solid simulation with contacts. Space is discretized instead of the solid. The solid flows through the mesh. Figure taken from Fan et al. [20].

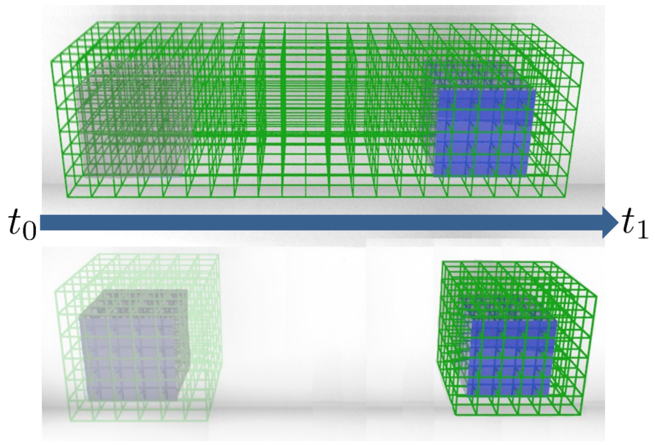


Figure 18: Eulerian-on-Lagrangian simulation. Each object has its own Eulerian space embedded in a Lagrangian mesh. (Top: Traditional Eulerian simulation. Bottom: EoL simulation.) Figure taken from Fan et al. [20].

the cost of some additional computation. However, Eulerian-on-Lagrangian has more a more diverse set of applications, including skin simulation.

The work done by Li et al. [30] is perhaps the most relevant to study for our Lagrangian-on-Lagrangian embedding (Fig. 19). They introduce a coupled system between body and skin, where the skin can slide on the body regardless of body motion. They do so by modeling the body as a Lagrangian mesh, which also serves as an Eulerian embedding for the skin. That is to say, the skin flows through the body mesh. The body is parameterized in 2D using the quasi-isometric patterns from Dong et al. [18]. The skin is then set in texture space where the Eulerian body nodes are mapped. To properly set the Eulerian

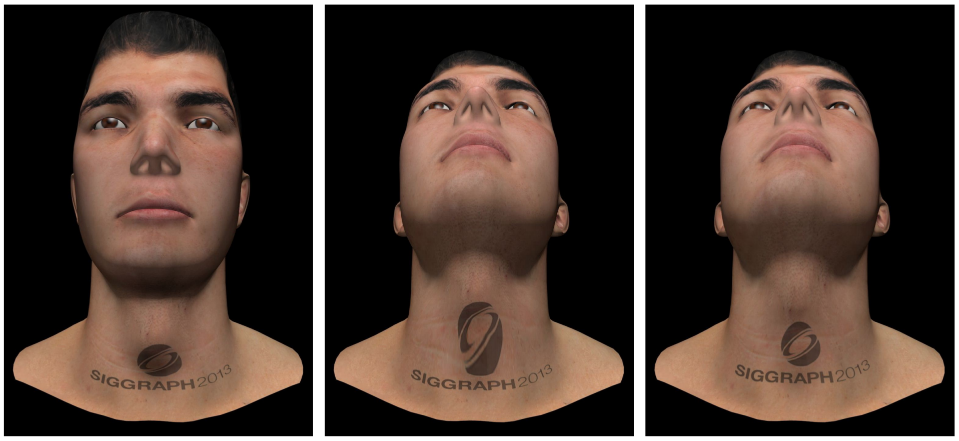


Figure 19: EoL (Eulerian-on-Lagrangian) Skin Simulation. Left: Unstretched skin on the body. Center: Body motion with skin not properly stretched. Right: Result of EoL skin. Figure taken from Li et al. [30].

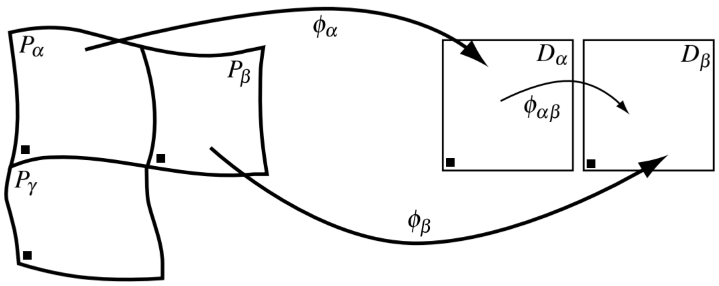


Figure 20: Quasi-Isometric patches. Tracing distances between patterns can be done with a straight line by stacking them together. Figure taken from Li et al. [30].

embedding, distances should be easily mapped in texture space which is precisely what the quasi-isometric patterns accomplish. These rectangular patterns may be stacked next to each other (Fig. 20) and the distances between body nodes can be traced as straight lines. Such an embedding could not be achieved as easily for arbitrary patterns.

4.2.3 The Case for an Eulerian-on-Lagrangian Optimization Framework

Given its virtues, the Eulerian-on-Lagrangian embedding by Li et al. [30] seems like the perfect candidate to use for our optimization scheme. In this section, we enumerate the unresolved problems with the this embedding for optimization. We divide these problems in the following categories: simulating cloth with deformable bodies, moving cloth boundaries, complex cloth patterns, and optimizing the patterns.

Simulating Cloth with Deformable Bodies In their original work, the EoL embedding did not consider the deformation of the body. However, it is very easy to extend their scheme to incorporate it without the need of a smooth body surface, which again, is why their embedding is so attractive. Under the EoL embedding, there is only one mesh for the body. Each vertex of the body contains information about its position in 3D ($\boldsymbol{y}$), and coordinates ($\bar{\boldsymbol{x}}$) that map to a point of the patterns in texture space. Notice that the cloth as a discretized mesh does not exist, and the patterns are a continuous space, where each vertex of the body is mapped. The energy of the system is then defined as

$$U = U_{\text{cloth}}(\boldsymbol{y}, \bar{\boldsymbol{x}}) + U_{\text{body}}(\boldsymbol{y}, \bar{\boldsymbol{y}}). \tag{116}$$

From the equation above, we can see that the cloth and the body share the same deformed configuration $\boldsymbol{y}$. The cloth slides on the body by changing the texture coordinates $\bar{\boldsymbol{x}}$. To add in the deformation of the body, we just have the contribution of the energy of the cloth with respect to the deformed body configuration $\boldsymbol{y}$, leading to the equilibrium

$$\frac{\partial U_{\text{cloth}}}{\partial \bar{\boldsymbol{x}}} = 0, \text{ and } \frac{\partial U_{\text{cloth}}}{\partial \boldsymbol{y}} + \frac{\partial U_{\text{body}}}{\partial \boldsymbol{y}} = 0, \tag{117}$$

which is an elegant formulation. However, as we move in deeper, some problems start to arise.

Moving Cloth Boundaries In their original work, the skin (cloth) covered the entirety of the body, which means that every vertex of the body ($\boldsymbol{y}$) is mapped to a point of the patterns ($\bar{\boldsymbol{x}}$) that is considered to be skin. Suppose we relax that constraint, and assume that there are sections of the body not covered by cloth. Even more, let's say the boundary of the cloth is allowed to move on the body. To incorporate these changes, a seemingly straightforward solution is to divide the pattern space into sections of cloth and no-cloth. This simple addition results in some interesting conceptual challenges.

Under a Lagrangian-on-Lagrangian embedding, every deformed vertex of the cloth is mapped to a correponding undeformed vertex. But now, under the Eulerian-on-Lagrangian embedding, it is possible for areas of the cloth not to be mapped by any vertex of the body. As an example, consider the simple case of every body vertex mapping to no-cloth points of the patterns. At first, it may seem like this problem can be avoided by having a good initial configuration. However, there is nothing enforcing that all the area of the cloth must be covered during the simulation. What is more, some numerical anomalies start to appear. Consider the case of a body surface triangle, where one vertex is in a cloth section of the patterns, and the other two are in no-cloth sections. The position of the boundary on the triangle can be inferred from its location on pattern space. The cloth energy of that triangle is

$$U^e_{\text{cloth}} = \int_{\Omega^e_{\text{cloth}}} \psi^e_{\text{cloth}} d\Omega^e_{\text{cloth}}, \tag{118}$$

where Ω^e_{cloth} corresponds to the area of the triangle covered by cloth, as no-cloth sections have 0 energy density. This area on a Lagrangian-on-Lagrangian embedding is constant, so the energy depended entirely on the energy density, and therefore the only way to minimize its energy was to minimize its strain. Here, we have two options; one, we can minimize its strain, or two, we can minimize the area covered by cloth. If we push the vertex in a cloth section towards the boundary, the strain may increase, but the area covered by cloth decreases, thus reducing the energy. This results in an energy per triangle with a local maximum and two local minima (one of which is undesired), and therefore unsuitable for simulation.

Complex Cloth Patterns In their work, the body was segmented into quasi-developable, rectangular patterns, which were easy to align and form elements in pattern space. Cloth patterns are seldom developable. While, vertices at the interior of the same patterns are easy to connect, vertices that lie on different patterns are a different problem. To compute the cloth energy of the triangle, one must find the three points of the cloth in pattern space $\bar{\boldsymbol{x}}$, and create a triangle to act as the undeformed configuration. If the patterns are not developable, they cannot be aligned to consistently create this triangle, as the borders of these patterns can only be aligned in 3D.

Optimizing the patterns Finally, we saw that under the Eulerian-on-Lagrangian embedding, there is no cloth mesh, and the patterns are a continuous space. Therefore, defining the design parameters in this case is an open problem, as the body references to the cloth in pattern space does not depend directly on the pattern borders.

4.3 Garment Modeling

Garment modeling is a time-intensive process that requires highly trained professionals. The automation of garment design roughly falls into two categories: sketch-based methods and adaptation methods. In sketch-based methods, sketching is typically used on top of the 3D character to generate a garment, such as in Turquin et al. [58], Wang et al. [63], Decaudin et al. [17], and Robson et al. [46]. Adaptation methods adapt existing garment designs to bodies of different shapes and sizes, such as in Meng et al. [34], Brouet et al. [11], and Bartle et al. [6]. For example, Chen et al. [13] propose a method for adapting a garment using the 3D body geometry obtained from a depth camera. The method by Berthouzoz et al. [8] parses patterns made by professional designers, automatically identifies the seams, and adapts them to virtual characters. Guan et al. [23] propose a method that uses deep learning to generate the animation of a garment worn by a virtual character of arbitrary shape.

4.4 Tight-fitting Clothing Design

Tight-fitting clothing is widely used for casual fashion, functional sportswear, medical compression garments, and many other applications. Kwok et al.[27] present a method for full-body skintight clothing design, focusing on optimal patch configuration that minimizes fitness energy based on heuristic distortion measures. Wang et al. [61, 62] present methods to optimize for the 2D patterns that achieve prescribed pressure distributions. However, their method considers only the 2D pattern shapes during the design, and not the location of the 3D seams. Furthermore, it does not take into account the deformation of the body under the garment, which is an important limitation.

The design and modeling of tight-fitting clothing as an optimization problem is necessarily holistic. It has to account for the size and shapes of the 2D patches, the material properties of the fabrics, the elastic behavior along the seams, the deformable 3D shape of the body, position of the seams on the 3D body, and the physical interaction between body and garment. In this work, we present a method that incorporates all of these components in one unified framework, enabling the optimization of advanced design objectives related to shape, comfort, and mechanical function.

Chapter 5

Cloth Body Model

Soft body dynamics is probably one of the most researched areas in computer graphics. Today, most systems are based on the finite element method, which compared to particle systems, are superior in terms of physical accuracy due to their capacity to model the soft body as a continuum (Bargteil and Shinar [5]).

This advantage has been exploited in a diverse set of applications, such as real-time animation of generic bodies (Barbič and Doug [4], and Müller and Gross [37]), and simulation of human flesh through geometric design (Teran et al. [57]) or through data-driven acquisition (Pons-Moll et al. [43]), among others. Finite element methods have also dominated cloth simulation. Focus has been given to simulating anisotropic fabrics (Volino et al. [60]), wrinkling (Rohmer et al. [47]), virtual try-ons and coupling with the human body. In these two latest areas, considerable effort has been given to detecting collisions, while modeling the body as a rigid body (Baraff and Witkin [2]). Modeling body deformations is important in the fashion industry, where some garments are designed to shape the body. A try-on physical model also allows for the design of clothing based on physical quantities, such as desired pressure or stretch.

In this chapter, we present a physically-based method that accurately couples the cloth to the body, where the body experiences deformation due to cloth stretch.

5.1 Representation

Clothes, in general, are fabricated by tracing a set of patterns on a fabric sheet, which is later cut and sewn together along the seams. The cloth is then worn by a person which deforms it by stretching it. The body in turn experiences deformation under the influence of the cloth. We introduce undeformed and deformed configurations for the cloth and the body to model this behavior.

For the cloth, we use a 3D membrane discretized by linear triangular elements where the undeformed configuration is a 2D mesh. This approach is commonplace nowadays, not only for cloth simulation but also for all kinds of thin shells such as rubber balloons (Skouras et al. [51]) or biological membranes (Massabò and Gambarotta [33]).

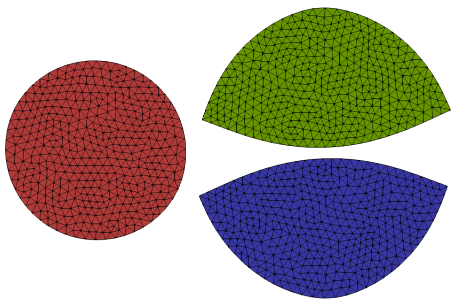
We use $\boldsymbol{x}$ (Fig. 21b) and $\hat{\boldsymbol{p}}$ (Fig. 21a) to represent the deformed and undeformed configurations of the cloth respectively. The mesh is composed of n elements, where $\bar{\boldsymbol{x}}$ is a planar mesh that shares the same number of elements as $\boldsymbol{x}$. This planar mesh $\bar{\boldsymbol{x}}$ represents the cloth patterns.

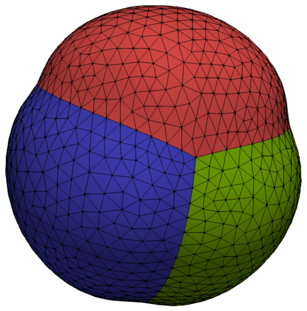
For the body, we use a standard volumetric mesh discretized by linear tetrahedra. This is a standard model frequently used in soft body dynamics.

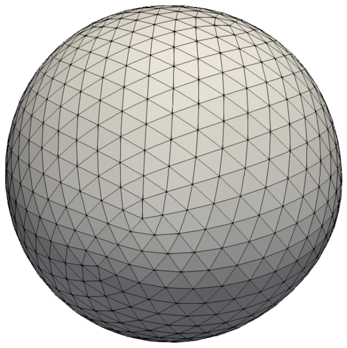
We use $\boldsymbol{y}$ (Fig. 21d) and $\bar{\boldsymbol{y}}$ (Fig. 21c) to represent the deformed and undeformed configurations of the body respectively. The mesh is composed of m elements and both configurations share the same connectivity.

While for each vertex in $\bar{\boldsymbol{y}}$ there exists a vertex in $\boldsymbol{y}$, the same does not hold true for $\bar{\boldsymbol{x}}$ and $\boldsymbol{x}$. The existence of seams means that for every vertex in $\boldsymbol{x}$, there may be 2 vertices in $\bar{\boldsymbol{x}}$ if it belongs to a seam or more than 2 if it belongs to a seam intersection. For every vertex in $\bar{\boldsymbol{x}}$, there exists only one vertex in $\boldsymbol{x}$.

Next, we want to establish a coupling between the cloth and the body. Since we are focusing on skintight clothing, we can avoid complex collision detection and response by having the cloth be always in contact with the body. That is to say, each deformed cloth


(a) Undeformed cloth configuration $\bar{\boldsymbol{x}}$.


(b) Deformed cloth configuration $\boldsymbol{x}$.


(c) Undeformed body configuration $\bar{\boldsymbol{y}}$.

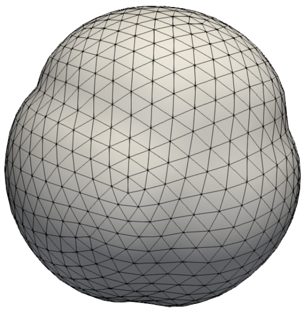

(d) Deformed body configuration $\boldsymbol{y}$.

Figure 21: Representation of the cloth and the body.

vertex lies somewhere on a deformed body element, and therefore can be expressed as a function of some barycentric coordinates $\boldsymbol{s} \in \mathbb{R}^2, \boldsymbol{s}_1 + \boldsymbol{s}_2 \leq 1$ and body vertices as $\boldsymbol{x}(\boldsymbol{s}, \boldsymbol{y})$.

To simplify notation, we choose the symbol χ to represent the set including the 3 values $\{\boldsymbol{s}, \boldsymbol{y}\}$. Therefore, for each cloth vertex $\boldsymbol{x}$, we have $\boldsymbol{q} = \{\boldsymbol{s}, \boldsymbol{y}\}$ and $\boldsymbol{x}(\boldsymbol{q})$.

The choice of the function $\boldsymbol{q} \to \boldsymbol{x}$ is an important one. The simplest choice is to simply use linear barycentric interpolation, which means that each cloth vertex depends only on the three body vertices (Fig. 22) of the element as:

$$\boldsymbol{x} = s_1 \boldsymbol{y}_1 + s_2 \boldsymbol{y}_2 + (1 - s_1 - s_2)\boldsymbol{y}_3 \tag{119}$$

Overall, each cloth triangle depends on a total of 9 body vertices (3 body vertices for each cloth vertex) .

While it is convenient and easy to implement, it has a problem. We want the cloth to be

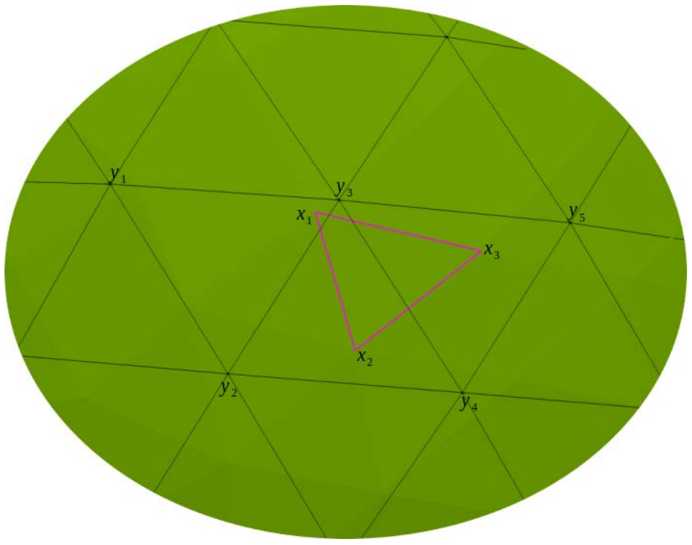


Figure 22: Linear coupling for a cloth element (in magenta) to the body. The cloth vertices are defined by $\boldsymbol{q}_1 = \{\boldsymbol{s}_1, \boldsymbol{y}_1, \boldsymbol{y}_2, \boldsymbol{y}_3\}$, $\boldsymbol{q}_2 = \{\boldsymbol{s}_2, \boldsymbol{y}_2, \boldsymbol{y}_4, \boldsymbol{y}_3\}$, $\boldsymbol{q}_3 = \{\boldsymbol{s}_3, \boldsymbol{y}_4, \boldsymbol{y}_5, \boldsymbol{y}_3\}$

able to slide on the body, thus a smooth body surface is ideal for this purpose. Sliding on a linear triangular mesh implies sharp changes in the trajectory of the cloth vertex.

A continuous approach is to have the body surface mesh be a subdivision surface (Appendix A). Since each point on the limit surface depends on an average of 12 vertices, under this scheme each cloth triangle depends on average on a total of 36 elements. The coupling between the body and the cloth using Loop subdivision surfaces is 4 times as stiff when compared to linear barycentric interpolation, but in return, we obtain continuity across the surface.

5.2 Cloth Physics

Given that the undeformed configuration of the cloth $\bar{\boldsymbol{x}}$ belongs in $\mathbb{R}^2 = (X, Y)$ and the deformed configuration $\boldsymbol{x}$ belongs in $\mathbb{R}^3 = (x, y, z)$ we interpolate the deformed space in terms of the undeformed space

$$(x, y, z) = \sum_i N_i(X, Y)\boldsymbol{x}_i, \tag{120}$$

where N_i are the linear shape functions of the triangle in the undeformed configuration. This gives us a 3×2 deformation gradient

$$\boldsymbol{F} = \begin{bmatrix} \frac{\partial x}{\partial X} & \frac{\partial x}{\partial Y} \\ \frac{\partial y}{\partial X} & \frac{\partial y}{\partial Y} \\ \frac{\partial z}{\partial X} & \frac{\partial z}{\partial Y} \end{bmatrix}, \tag{121}$$

which results in a symmetric 2×2 right Cauchy strain tensor $\boldsymbol{C} = \boldsymbol{F}^T\boldsymbol{F}$.

We start by defining our cloth energy based on the Saint-Venant Kirchhoff strain energy density function per element as

$$\psi_{cloth,e} = \frac{1}{2}\lambda \mathrm{tr}(E)^2 + \mu \mathrm{tr}(E^2), \tag{122}$$

where λ, μ are the first Lame parameter and shear modulus respectively, and $\boldsymbol{E}$ is the non-linear Green-Lagrange strain tensor $\boldsymbol{E} = \frac{1}{2}(\boldsymbol{C} - \boldsymbol{I}_2)$ where $\boldsymbol{I}_2$ is the 2-dimensional identity matrix.

The elastic energy per element is then

$$U_{cloth,e} = t \int_\Omega \psi_{cloth,e} dA = tA\psi, \tag{123}$$

where A is the area of the undeformed element and t is its thickness. We assume that the thickness of the element remains constant regardless of deformation.

This energy is normally the hyperelastic isotropic version of Hooke's Law, however, since we are using the non-linear Green-Lagrange strain tensor, the energy is quartic on the principal strains of the deformation gradient $\boldsymbol{F}$ instead of quadratic.

If we express $\boldsymbol{C}$ in terms of its principal stretches we get

$$\boldsymbol{C} = \lambda_1 \boldsymbol{M}\boldsymbol{M}^T + \lambda_2 \boldsymbol{N}\boldsymbol{N}^T, \tag{124}$$

where $\boldsymbol{M}, \boldsymbol{N}$ are orthonormal bases, and $\lambda_1, \lambda_2, \lambda_2 \leq \lambda_1$ are the principal stretches of $\boldsymbol{C}$, which are the squared principal stretches of $\boldsymbol{F}$.

This definition of $\boldsymbol{C}$ allows us to rewrite the Saint-Venant Kirchhoff strain density function in terms of the principal stretches of $\boldsymbol{C}$ as

$$\psi_{cloth,e} = \frac{1}{2}\lambda(\lambda_1 + \lambda_2 - 2)^2 + \mu((\lambda_1 - 1)^2 + (\lambda_2 - 1)^2), \tag{125}$$

where the relationship between ψ and the principal stretches of $\boldsymbol{C}$ becomes more evident. We want to use this definition to make an important observation: If both principal stretches are 1, the energy is of course 0. However, if one or both of the principal stretches is lesser than 1, there is compression resulting in a positive energy outcome, which translates to compressive forces.

In solid mechanics, objects can be compressed under external forces and compressive forces are a natural response. Cloth, as other membranes, do not experience this behavior. Suppose a piece of cloth is stretched beyond the tolerance of its Poisson's ratio. To compensate for this deformation, cloth experiences an out-of-plane deformation devoid of any compressive forces where a solid would just compress. This out of plane deformation is known as wrinkling.

To have an accurate representation of the cloth forces, we need a model that is capable of representing wrinkling.

Tension Field Theory

While wrinkling in itself is a complex behavior, tension field theory [42] offers an acceptable, simplified alternative for isotropic materials. The idea is simple; we want to divide the energy into 3 regimes depending on the state of the principal stretches: taut, wrinkled, and slack.

The unchanged definition of our energy assumes that both principal stretches are tensile and becomes our definition for the taut energy.

If the smallest principal stretch becomes smaller than the cloth's tolerance to the stretch experienced by the biggest principal stretch ($\lambda_1 \geq 1$), the cloth is considered wrinkled and the energy has to be changed accordingly.

The tolerance point of the smallest principal stretch can be found by simply finding the energetic minimum of λ_2 with respect to λ_1. More specifically, $\bar{\lambda_2}(\lambda_1) = \text{argmin}_{\lambda_2}\psi$, where $\bar{\lambda_2}$ is the energetic minimum.

For the case of the Saint-Venant Kirchhoff (StVK) energy density we have

$$\bar{\lambda_2}(\lambda_1) = -\frac{\lambda\lambda_1 - 2\lambda - 2\mu}{\lambda + 2\mu}, \tag{126}$$

and replacing this expression into our strain energy density function, we get the wrinkled energy density function

$$\psi_{wrinkled} = \frac{\mu(\lambda + \mu)}{2\lambda + 4\mu}(\lambda_1 - 1)^2. \tag{127}$$

Notice that this energy only depends on λ_1 and therefore the compression experienced in λ_2 is completely ignored.

In the slack case, both principal stretches are smaller than 1 and therefore the cloth does not experience any force. Then

$$\psi_{slack} = 0. \tag{128}$$

Unifying the 3 regimes we obtain the tension field compliant piecewise strain energy density function for Saint-Venant Kirchhoff as

$$\psi_{cloth,e} = \begin{cases} \psi_{taut} = \frac{\lambda}{8}(\lambda_1 + \lambda_2 - 2)^2 + \frac{\mu}{4}((\lambda_1 - 1)^2 + (\lambda_2 - 1)^2) & \lambda_1 \geq 1, \lambda_2 \geq \bar{\lambda}_2(\lambda_1) \\ \psi_{wrinkled} = \frac{\mu(\lambda+\mu)}{2\lambda+4\mu}(\lambda_1 - 1)^2 & \lambda_1 \geq 1, \lambda_2 \leq \bar{\lambda}_2(\lambda_1) \\ \psi_{slack} = 0 & \lambda_1 \leq 1 \end{cases} \cdot \tag{129}$$

This piecewise energy is C_1 continuous on the principal stretches, which means that the forces are continuous but the Hessian is not. This behavior is shown in Fig. 23.

Given that the energy is defined in terms of the principal stretches, the forces per element may be computed using the chain rule as

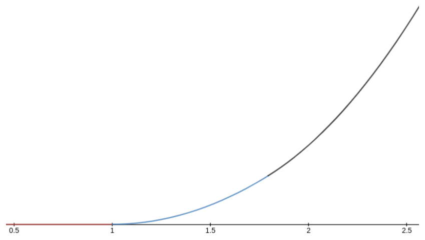
$$\boldsymbol{f}_e = -\frac{dU_e}{d\boldsymbol{x}} = -\frac{\partial U_e}{\partial \lambda_1}\frac{\partial \lambda_1}{\partial \boldsymbol{x}} - \frac{\partial U_e}{\partial \lambda_2}\frac{\partial \lambda_2}{\partial \boldsymbol{x}}. \tag{130}$$

Since $\boldsymbol{C}$ is a 2×2 matrix, its eigenvalues (principal stretches) can be found analytically, and the relationship between λ_1, λ_2 and $\boldsymbol{x}$ can be expressed through $\boldsymbol{C}$ as

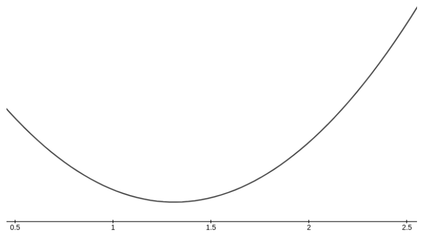
$$\lambda_1 = \frac{1}{2}\mathrm{tr}(\boldsymbol{C}) + \sqrt{\frac{1}{4}\mathrm{tr}(\boldsymbol{C})^2 - \det(\boldsymbol{C})}, \text{ and} \tag{131}$$

$$\lambda_2 = \frac{1}{2}\mathrm{tr}(\boldsymbol{C}) - \sqrt{\frac{1}{4}\mathrm{tr}(\boldsymbol{C})^2 - \det(\boldsymbol{C})}, \tag{132}$$

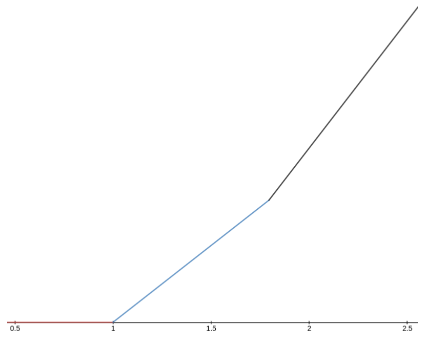
where λ_1 is always going to be the biggest principal stretch and λ_2 the smallest.



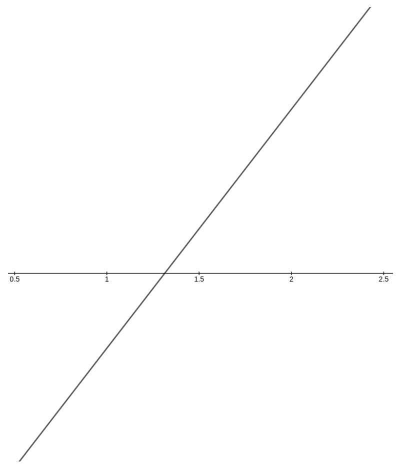
(a) Relaxed Stvk energy. It shows the three regimes (red=slack, blue=wrinkled, black=taut).



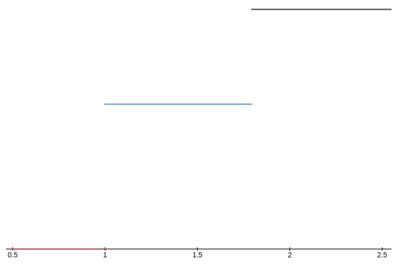
(b) Standard StVK, it shows non-zero energy under compression.



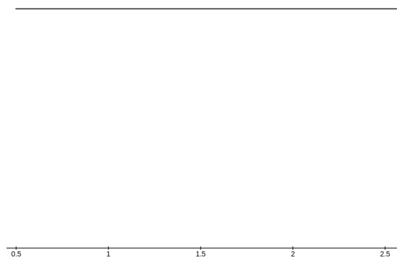
(c) The gradient for the three regimes. It is shown to be piece-wise continuous.



(d) Gradient for standard StVK. Under compression the gradient becomes negative



(e) The second derivative of the relaxed energy. It is discontinuous and each regime is a constant.



(f) The second derivative is a constant for StVK with respect to the principal stretches.

Figure 23: Energy and derivatives of the energy with respect to the biggest principal stretch where the smallest principal stretch is $\frac{1}{2}$.

The total energy of the cloth is

$$U_{cloth} = \sum_e U_{cloth,e}. \tag{133}$$

5.3 Coupled Simulation

We set the strain energy density of the body per tetrahedron as an isotropic compressible Neo-Hookean material, defined per element as

$$\psi_{body,e} = \frac{\mu}{2}(\bar{I}_1 - 3) + \frac{\lambda}{2}(J-1)^2, \tag{134}$$

where λ, μ are the first Lame parameter and shear modulus respectively, $\bar{I}_1 = J^{-\frac{2}{3}} I_1$, I_1 is the first invariant of the right Cauchy stress tensor $\boldsymbol{C}$ and J is the determinant of the deformation gradient.

For the body we use a straightforward volumetric definition and therefore the deformation gradient $\boldsymbol{F}$ and the right Cauchy tensor $\boldsymbol{C}$ are 3×3 matrices. The energy per element is then defined as

$$U_{body,e} = \int_{\Omega} \psi_{body,e} dV = V\psi_{body,e}, \tag{135}$$

where V is the volume of the undeformed tetrahedron.

The total energy of the body is

$$U_{body} = \sum_e U_{body,e}. \tag{136}$$

If treated separately, the body and cloth would have as forces

$$\boldsymbol{f}_{body} = -\frac{dU_{body}}{d\boldsymbol{y}}, \text{ and} \tag{137}$$

$$\boldsymbol{f}_{cloth} = -\frac{dU_{cloth}}{d\boldsymbol{x}}, \tag{138}$$

where the cloth forces are computed as shown in section 5.2. In section 5.1, we introduced the assumption that the cloth vertices never separate from the body, and a representation where the cloth vertices depend on the body vertices $\boldsymbol{x}(\boldsymbol{q} = \{\boldsymbol{s}, \boldsymbol{y}\})$. If we redefine our cloth forces in terms of this new definition we get

$$\boldsymbol{f}_{cloth} = -\frac{dU_{cloth}}{d\boldsymbol{q}} = \left\{-\frac{dU_{cloth}}{d\boldsymbol{s}}, -\frac{dU_{cloth}}{d\boldsymbol{y}}\right\}, \tag{139}$$

which gives a set of two types of forces, one for the barycentric coordinates $\boldsymbol{s}$ and another one for the body vertices $\boldsymbol{y}$.

These forces can be derived from the Cartesian forces using the chain rule:

$$\frac{dU_{cloth}}{d\boldsymbol{q}} = \frac{\partial \boldsymbol{x}}{\partial \boldsymbol{q}}^{T} \frac{\partial U_{cloth}}{\partial \boldsymbol{x}}, \tag{140}$$

where $\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{q}}$ is known as the surface derivative.

The force $-\frac{dU_{cloth}}{dy}$ is self-explanatory. It is the force that the cloth exerts on the body causing it to deform, while $-\frac{dU_{cloth}}{ds}$ are the tangential forces of the cloth on the body, causing it to slide. The projection of the cartesian cloth force to the plane with normal $\boldsymbol{n}_{body}$, where $\boldsymbol{n}_{body}$ is the normal of the body at that point.

With forces for the body and cloth defined, we can set our equilibrium conditions. Consider a stretched piece of cloth surrounding an elastic body, currently undeformed. Given the deformation of the cloth, it will exert forces on the body attempting to regain its original shape. But as the body is compressed, it will generate forces of its own trying to keep its shape until the two opposing forces come to an equilibrium. This relationship is expressed in the equation

$$-\frac{dU_{body}}{d\boldsymbol{y}} - \frac{dU_{cloth}}{d\boldsymbol{y}} = 0. \tag{141}$$

If left like that, the body would experience tangential forces from the cloth. Assuming there is no friction between the body and the cloth, the cloth will slide freely on the body until it reaches an equilibrium configuration. In this case, all the cloth forces exerted on the body are normal to the body surface. This equilibrium condition is expressed as

$$-\frac{dU_{cloth}}{d\boldsymbol{s}} = 0. \tag{142}$$

Equations 141 and 142 are the two desired equilibrium conditions. The strength of this approach is that the physical coupling of the body and the cloth can be achieved through merely applying the chain rule.

Solving for the equilibrium configuration

Having set our framework, we now focus on obtaining an equilibrium configuration given a set of cloth patterns $\bar{\boldsymbol{x}}$, known material parameters λ, μ for both the cloth and the body, an undeformed body $\boldsymbol{y}$ and an initial guess for the coupled system $\boldsymbol{x}$ and $\boldsymbol{y}$.

Assuming that the cloth vertices of the initial guess are sufficiently close to the deformed body vertices, we can find their parameterization $\boldsymbol{q}$ in the following manner. First, we retrieve the body vertex whose distance is closer to our cloth vertex and select its faces as potential candidates. Then we find barycentric coordinates of our cloth vertex for each face candidate by solving the system

$$\begin{bmatrix} y_{2x} - y_{1x} & y_{3x} - y_{1x} \\ y_{2y} - y_{1y} & y_{3y} - y_{1y} \\ y_{2z} - y_{1z} & y_{3z} - y_{1z} \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \end{bmatrix} = \boldsymbol{x}, \tag{143}$$

which being a 3×2 system, can be solved through QR decomposition. If the solution to this system gives barycentric coordinates such that $0 \leq s_1 \leq 1, 0 \leq s_2 \leq 1, 1 - s_1, s_2 \leq 1$, then the cloth vertex projection to the body triangle lies within its boundaries and is chosen to contain the cloth vertex with the barycentric coordinates $\boldsymbol{s} = (s_1, s_2)$. If not, we search in the other candidates until an appropriate one is found. A closed body mesh guarantees at least one viable candidate.

Now, we have all the ingredients necessary to find the equilibrium configuration. Given the definition of our equilibrium conditions we can frame our problem as an energy minimization problem

$$\boldsymbol{q}_m = \text{argmin}_{\boldsymbol{q}}(U_{body} + U_{cloth}), \tag{144}$$

where $\boldsymbol{q}_m$ is the deformed configuration where the system is at equilibrium.

This system can be solved using a gradient based optimization method. Since, we are able to compute sparse second order derivatives for this problem, we choose damped Newton to solve it. The Newton iteration system is defined as

$$\left(\frac{d^2U_{cloth}}{d\boldsymbol{q}^2} + \frac{d^2U_{body}}{d\boldsymbol{q}^2} + \alpha\boldsymbol{I}\right)\Delta\boldsymbol{q} = -\frac{dU_{body}}{d\boldsymbol{q}} - \frac{dU_{cloth}}{d\boldsymbol{q}}. \tag{145}$$

Keeping in mind that the portion of the derivatives of U_{body} with respect to $\boldsymbol{s}$ is 0, since the energy of the body only depends on the body vertices $\boldsymbol{y}$, the right part are the equilibrium conditions, and the system is considered converged when it is 0.

The reason behind using damped Newton is to regulate the Hessian. When the current guess is far away from the solution, the Hessian may become indefinite, thus we must force the system to take a descent direction by following the gradient. It can be seen that by increasing the damping parameter α, the Hessian becomes less relevant and the step $\Delta \boldsymbol{q}$ is tilted towards the direction opposite to that of the gradient. The damping parameter is adjusted after every iteration following the rules specified in section 2.1.2.

Taking the Newton step

Next is the problem of advancing the current guess $\boldsymbol{q}_i$ to $\boldsymbol{q}_{i+1}$. For the body vertices, it is enough to just add the current step to the current guess

$$\boldsymbol{y}_{i+1} = \boldsymbol{y}_i + \Delta \boldsymbol{y}. \tag{146}$$

In the case of the barycentric coordinates $\boldsymbol{s}$, where the vertex is traveling across the elements of a piece-wise mesh, we must adopt a different strategy.

Suppose a cloth vertex has barycentric coordinates $\boldsymbol{s}_i$ at the beginning of the step and has a current step of $\Delta \boldsymbol{s}$.

First, we apply the step as usual and we get a new set of coordinates

$$\boldsymbol{s}_{i+1} = \boldsymbol{s}_i + \Delta \boldsymbol{s}. \tag{147}$$

If the conditions $0 \leq \boldsymbol{s}_{i,1}$, $0 \leq \boldsymbol{s}_{i,2}$ and $0 \leq 1 - \boldsymbol{s}_{i,1} - \boldsymbol{s}_{i,2}$ are sustained, it means that the resulting coordinates remain within the boundaries of the current body face and accept it as the result of applying the step (Note that we do not check for the coordinates being less than 1, as that condition is implicitly checked in the third condition).

If a condition is violated, it means the cloth vertex must cross the boundary of the current face to a neighboring face across an edge. If more than one condition are violated at the same time, we must identify which one is violated first as the first edge the vertex crosses on its path. To identify the closest edge, we use the expressions

$$d_1 = -\frac{\boldsymbol{s}_{i,1}}{\Delta \boldsymbol{s}_{i,1}}, \tag{148}$$

$$d_2 = -\frac{\boldsymbol{s}_{i,2}}{\Delta \boldsymbol{s}_{i,2}}, \text{ and} \tag{149}$$

$$d_3 = \frac{1 - \boldsymbol{s}_{i,1} - \boldsymbol{s}_{i,2}}{\Delta \boldsymbol{s}_{i,1} + \Delta \boldsymbol{s}_{i,2}}. \tag{150}$$

It is clear that if a current path never crosses an edge, d is negative. If d is bigger than 1, it means the path is not long enough to cross the edge. Therefore we choose the smallest positive d and bring the current vertex to the closest edge. This is achieved by the expression

$$\boldsymbol{s}_n = \boldsymbol{s} + \min(d_1, d_2, d_3)\Delta \boldsymbol{s}. \tag{151}$$

So far, we have taken the step clamped up to a bordering edge. We want to take the full step by following the path on the neighboring triangles as well. First, we subtract the distance already traveled from our initial position to the border as

$$\Delta \boldsymbol{s}_n = (1 - \min(d_1, d_2, d_3))\Delta \boldsymbol{s}. \tag{152}$$

Now, it is convenient to remember that this step is not taken in Cartesian coordinates, but in barycentric coordinates specific to the current body face. In order to continue the path we must transform both the current coordinates $\boldsymbol{s}_n$, and the current shortened step $\Delta \boldsymbol{s}_n$ to the new face. This is done by solving the system

$$\frac{d\boldsymbol{x}}{d\boldsymbol{s}_{new}}\boldsymbol{s}_m = \frac{d\boldsymbol{x}}{d\boldsymbol{s}_{old}}\boldsymbol{s}_n \tag{153}$$

$$\frac{d\boldsymbol{x}}{d\boldsymbol{s}_{new}}\Delta\boldsymbol{s}_m = \frac{d\boldsymbol{x}}{d\boldsymbol{s}_{old}}\Delta\boldsymbol{s}_n, \tag{154}$$

where $\boldsymbol{s}_m$ and $\Delta\boldsymbol{s}_m$ are the current position and step in coordinates of the new face, and $\frac{d\boldsymbol{x}}{ds}$ is the surface derivative as introduced in section 5.1. The surface derivative is a 3×2 matrix and the system can be solved using QR decomposition.

With the new step and new step direction, we proceed to repeat the process from Eq. 147 until all the barycentric coordinates are properly bounded, and the step has been taken in full.

5.3.1 Derivatives

The computation of the derivatives of the body elastic energy with respect to the body vertices is straightforward. For the case of the energy of the cloth, we first need to compute the derivatives with respect to the Cartesian cloth coordinates $\boldsymbol{x}$. Then we compute the derivatives of $\boldsymbol{x}$ with respect to $\boldsymbol{q}$ (the surface derivative). The final derivatives of the cloth energy with respect to $\boldsymbol{q}$ can be computed using the chain rule per cloth element as

$$\frac{dU_{cloth,e}}{d\boldsymbol{q}_e} = \frac{\partial \boldsymbol{x}_e}{\partial \boldsymbol{q}_e}^T \frac{\partial U_e}{\partial \boldsymbol{x}_e}, \tag{155}$$

and

$$\frac{d^2U_{cloth,e}}{d\boldsymbol{q}_e^2} = \frac{\partial \boldsymbol{x}_e}{\partial \boldsymbol{q}_e}^T \frac{\partial^2 U_e}{\partial \boldsymbol{x}_e^2} \frac{\partial \boldsymbol{x}_e}{\partial \boldsymbol{q}_e} + \sum_i \frac{\partial^2 \boldsymbol{x}_e}{\partial \boldsymbol{q}_e^2}_i \frac{\partial U_e}{\partial \boldsymbol{x}_{e\,i}}, \tag{156}$$

where Eq. 155 is the gradient and 156 is the Hessian.

The per-element surface gradient $\frac{\partial \boldsymbol{x}_e}{\partial \boldsymbol{q}_e}$ is constructed as

$$\frac{\partial \boldsymbol{x}_e}{\partial \boldsymbol{q}_e} = \begin{bmatrix} \frac{\partial \boldsymbol{x}_{e,1}}{\partial \boldsymbol{s}_{e,1}} & 0 & 0 & \frac{\partial \boldsymbol{x}_{e,1}}{\partial \boldsymbol{y}_{e,1}} & 0 & 0 \\ 0 & \frac{\partial \boldsymbol{x}_{e,2}}{\partial \boldsymbol{s}_{e,2}} & 0 & 0 & \frac{\partial \boldsymbol{x}_{e,2}}{\partial \boldsymbol{y}_{e,2}} & 0 \\ 0 & 0 & \frac{\partial \boldsymbol{x}_{e,3}}{\partial \boldsymbol{s}_{e,3}} & 0 & 0 & \frac{\partial \boldsymbol{x}_{e,3}}{\partial \boldsymbol{y}_{e,3}} \end{bmatrix}, \tag{157}$$

where each row corresponds to an element vertex. Each block $\frac{\partial \boldsymbol{x}_{e,i}}{\partial \boldsymbol{s}_{e,i}}$ is of size 3×2, and each $\frac{\partial \boldsymbol{x}_{e,i}}{\partial \boldsymbol{y}_{e,i}}$ is of size 3×9 for linear barycentric interpolation, 3×36 for a Loop regular patch with the size for irregular patches changed accordingly. So, for a cloth element lying on 3 regular patches, the size of the surface derivative is 9×114.

For the Hessian, we need to compute the second surface derivative $\frac{\partial^2 \boldsymbol{x}_e}{\partial \boldsymbol{q}_e^2}$ which is a 3rd order tensor. For this reason, it is more convenient to slice it and add each contribution per cloth-vertex-component as it is shown in Eq. 156. Each i corresponds to a cloth vertex times 3 for each dimension x, y, z for a total of 9 additions. Since the surface function is linear on the body vertices, the Hessian computation can be simplified as $\frac{\partial^2 \boldsymbol{x}_e}{\partial \boldsymbol{y}_e^2} = 0$. The resulting Hessian has a size of 114×114 for regular patches.

The global gradient and Hessian are constructed from these element derivatives using the standard finite element method.

5.3.2 Results

In this section, we provide some simulation results along with its convergence for several examples.

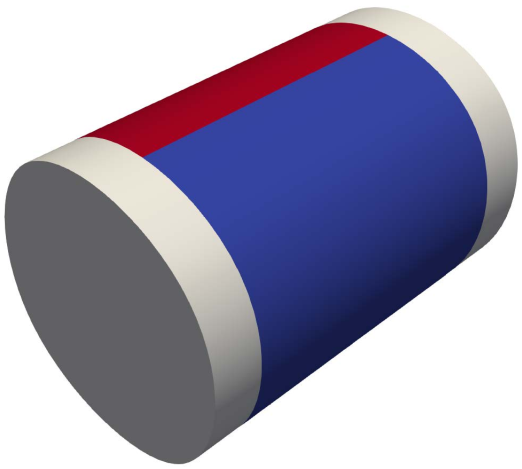
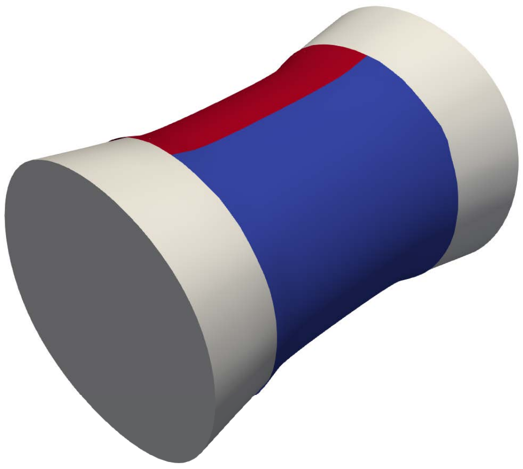
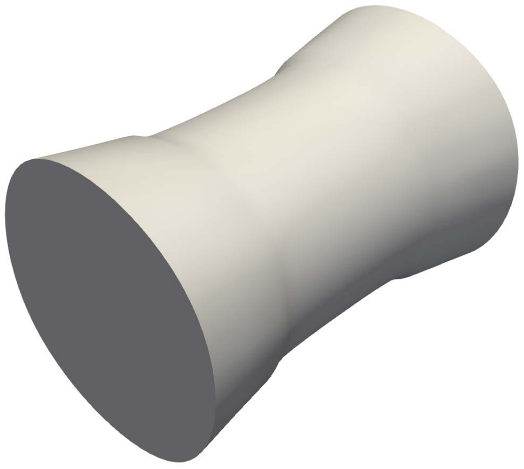
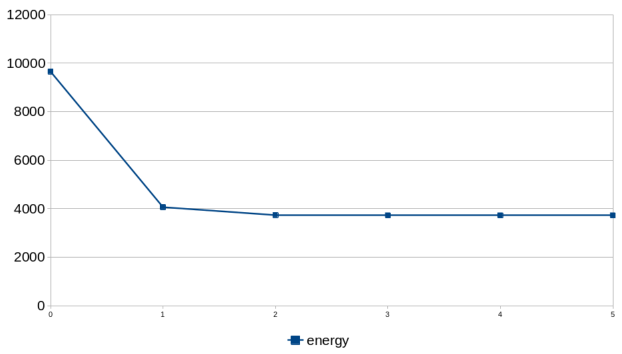


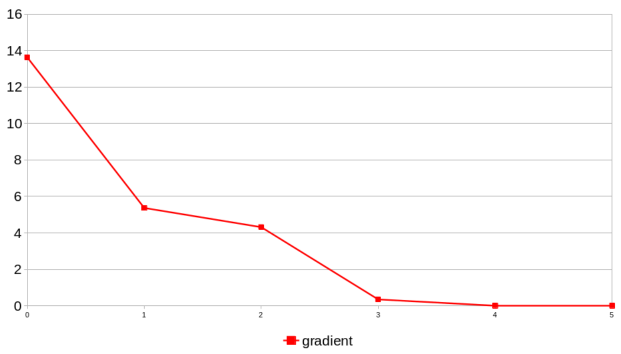
Figure 24: Initial guess for the cylinder deformation


(a) Deformation of the body with the cloth with all cloth boundaries fixed.


(b) Deformation of the body with all cloth boundaries fixed.

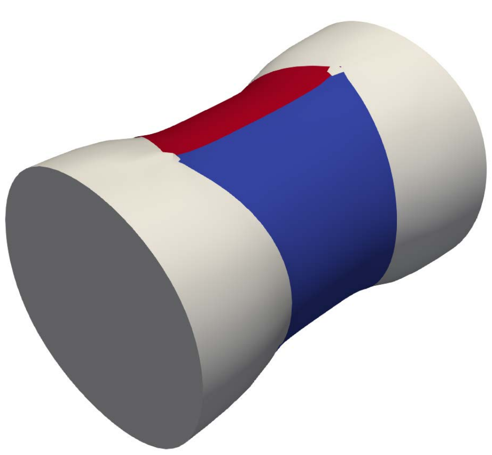


(c) Energy progression per iteration.

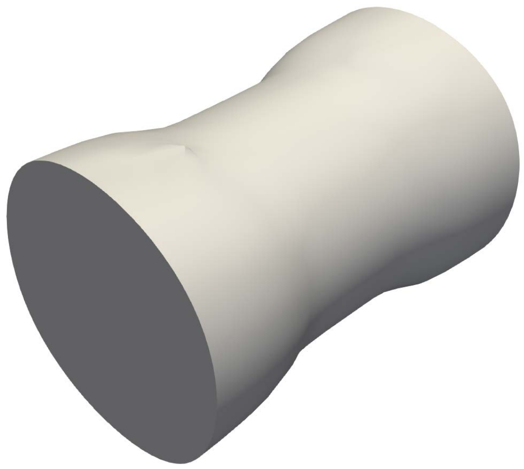


(d) Gradient progression per iteration.

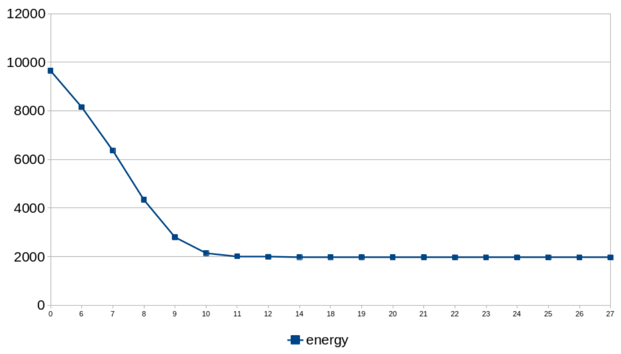
Figure 25: Result for a cylinder with a tight cloth with all boundaries fixed using the initial guess of Fig. 24.



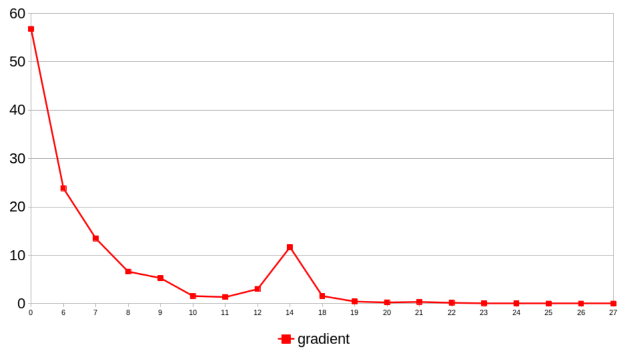
(a) Deformation of the body with the cloth with 4 cloth vertices fixed.



(b) Deformation of the body with 4 cloth vertices fixed.

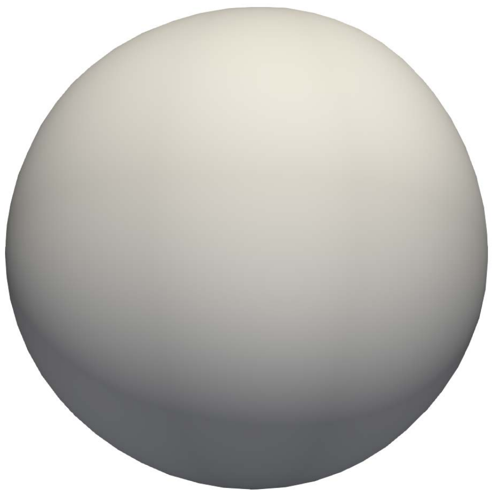


(c) Energy progression per iteration.



(d) Gradient progression per iteration.

Figure 26: Result for a cylinder with a tight cloth with 4 vertices fixed using the initial guess of Fig. 24.


(a) Initial guess for the body.

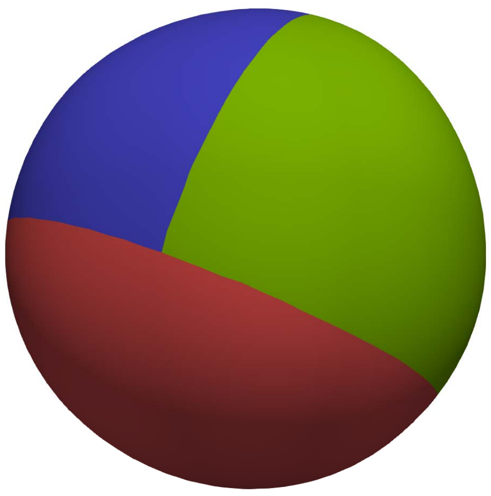
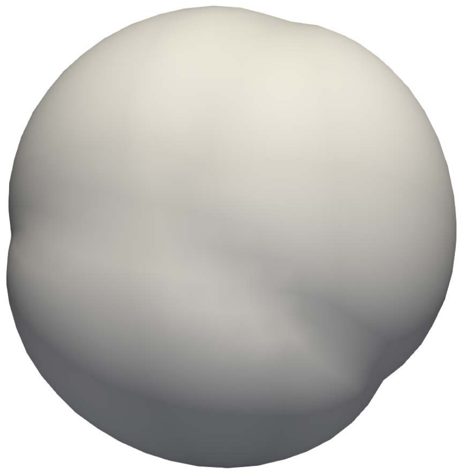
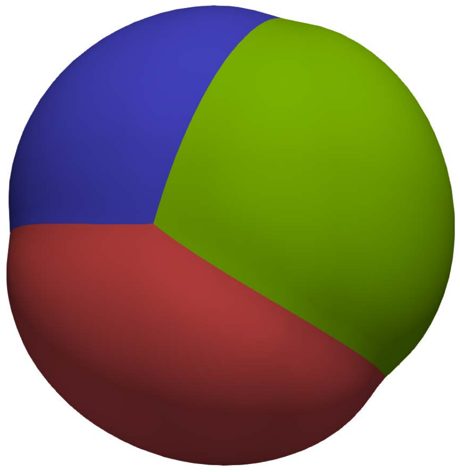
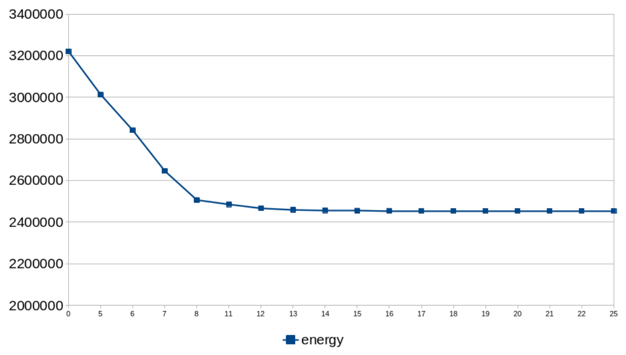

(b) Initial guess for the cloth.

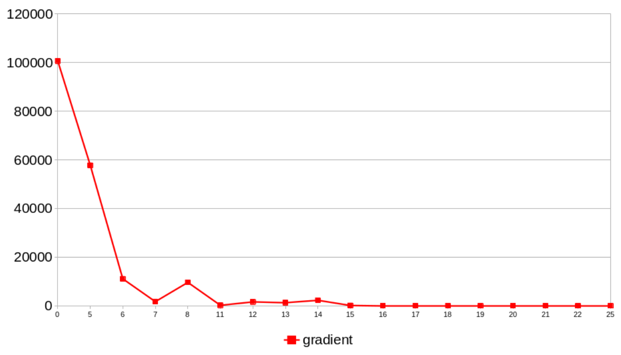
Figure 27: Initial guess of a fully covered sphere.


(a) Deformed body.


(b) Deformed cloth.

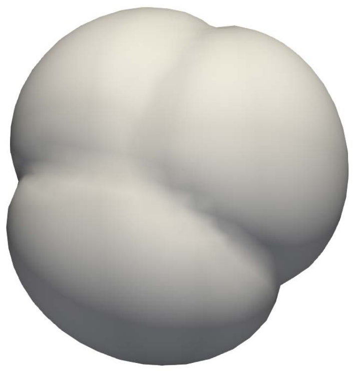


(c) Energy progression per iteration.

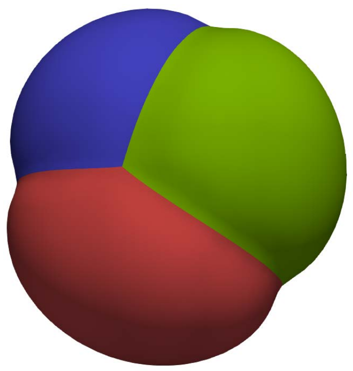


(d) Gradient progression per iteration.

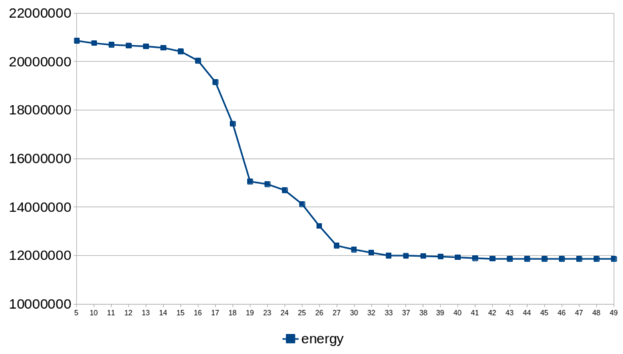
Figure 28: Result for a sphere with a tight cloth and stiff seams using the initial guess of Fig. 27.



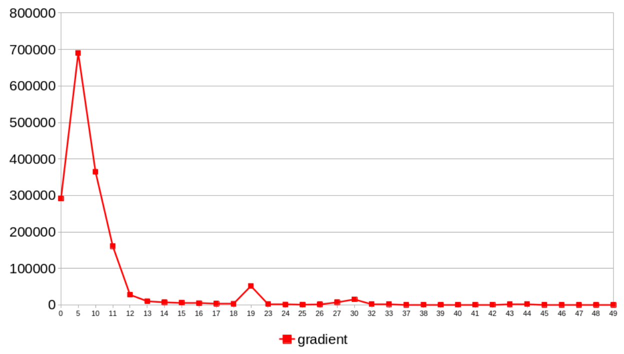
(a) Deformed body.



(b) Deformed cloth.



(c) Energy progression per iteration.



(d) Gradient progression per iteration.

Figure 29: Result for a sphere with a tight cloth twice as small as the one in Fig. 28 and stiff seams using the initial guess of Fig. 27.

Chapter 6

Optimization

In the simulation, the goal was to compute $\boldsymbol{y}$ and $\boldsymbol{x}$, but we can turn the tables and select other variables that we would like to solve for, allowing for complex modeling queries that can be used for a range of interesting applications in a single elegant numerical framework. For instance, we can optimize for the 2D patterns ($\bar{\boldsymbol{x}}$) and the embedding of the cloth on the 3D mesh that would effectively provide the locations of the seams ($\boldsymbol{x}$) assuming a $\bar{\boldsymbol{y}}$ and $\boldsymbol{y}$. Or we can optimize for the $\bar{\boldsymbol{x}}$ and $\boldsymbol{x}$ assuming any number of stretch constraints on the cloth. In general, given an arbitrary objective T, we frame the system as a constrained optimization problem:

$$\text{Minimize } T(\boldsymbol{x}(\boldsymbol{y}, \boldsymbol{s}), \boldsymbol{p}) \tag{158}$$

$$\text{s.t. } \frac{dU_{cloth}}{d\boldsymbol{y}} + \frac{dU_{body}}{d\boldsymbol{y}} = 0, \text{ and } \frac{dU_{cloth}}{d\boldsymbol{s}} = 0, \tag{159}$$

where $\boldsymbol{p}$ is a set of design parameters which controls the shape of the patterns $\bar{\boldsymbol{x}}$.

We optimize $\bar{\boldsymbol{x}}$ using L-BFGS [1, 39] and simulate $\boldsymbol{q}$ for force equilibrium at each iteration. L-BFGS is a quasi-Newton optimization method that requires only the gradient $\frac{dT}{d\bar{\boldsymbol{x}}}$. Additional to the partial gradient with respect to $\bar{\boldsymbol{x}}$ we need to take also into account the change induced by the simulation to the deformed cloth and body. This results in the gradient

$$\frac{dT}{d\boldsymbol{p}} = \frac{\partial T}{\partial \boldsymbol{p}} + \frac{\partial \boldsymbol{q}}{\partial \boldsymbol{p}}^T \frac{\partial T}{\partial \boldsymbol{q}}. \tag{160}$$

The matrix $\frac{\partial \boldsymbol{q}}{\partial \boldsymbol{p}}$ is found using sensitivity analysis (Sec. 2.2.3). Since at each iteration the force is at equilibrium $f = \frac{dU}{d\boldsymbol{q}} = 0$, we get that $\frac{d\boldsymbol{f}_{\boldsymbol{q}}}{d\boldsymbol{p}} = 0$, and

$$\frac{d\boldsymbol{f}}{d\boldsymbol{p}} = \frac{\partial \boldsymbol{f}_{\boldsymbol{q}}}{\partial \boldsymbol{p}} + \frac{\partial \boldsymbol{f}_{\boldsymbol{q}}}{\partial \boldsymbol{q}} \frac{\partial \boldsymbol{q}}{\partial \boldsymbol{p}} = 0, \tag{161}$$

which rearranged gives

$$\frac{\partial \boldsymbol{q}}{\partial \boldsymbol{p}} = -\frac{\partial \boldsymbol{f}_{\boldsymbol{q}}}{\partial \boldsymbol{q}}^{-1} \frac{\partial \boldsymbol{f}_{\boldsymbol{q}}}{\partial \boldsymbol{p}}. \tag{162}$$

This expression requires solving $dim(\boldsymbol{p})$ linear systems. However, using the self-adjoint method, we can simplify this to the single linear system

$$\frac{dT}{d\boldsymbol{p}} = \frac{\partial T}{\partial \boldsymbol{p}} - \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{p}}^T \frac{\partial \boldsymbol{f}_{\boldsymbol{q}}}{\partial \boldsymbol{q}}^{-1} \frac{\partial T}{\partial \boldsymbol{q}}. \tag{163}$$

Finally, we add each target or regularizer contributions to the two partial gradients $\frac{\partial T}{\partial \boldsymbol{p}}$ and $\frac{\partial T}{\partial \boldsymbol{q}}$, and compute the full gradient $\frac{dT}{d\boldsymbol{p}}$.

$\frac{\partial \boldsymbol{f}_{\boldsymbol{x}}}{\partial \bar{\boldsymbol{x}}}$ is obtained using symbolic differentiation software and is then later set in terms of $\boldsymbol{q}$ using the chain rule with the surface derivatives.

$$\frac{\partial \boldsymbol{f}_{\boldsymbol{q}}}{\partial \boldsymbol{p}} = \frac{\partial \boldsymbol{x}}{\partial \boldsymbol{q}}^T \frac{\partial \boldsymbol{f}_{\boldsymbol{x}}}{\partial \boldsymbol{p}}. \tag{164}$$

Pattern Parameterization The shapes of the cloth patterns are defined completely through their boundaries, which in turn are controlled by the design parameters $\boldsymbol{p}$. Nevertheless, evaluating elastic energies and their derivatives requires triangle meshes and we must therefore determine the position of interior vertices as a function of the boundary shape. For this purpose, we use bi-harmonic coordinates [65] and express the location of the pattern mesh vertices $\bar{\boldsymbol{x}}$ as a linear function of the design parameters $\boldsymbol{p}$ given by the matrix $\boldsymbol{W}$ of bi-harmonic weights, $\bar{\boldsymbol{x}} = \boldsymbol{W}\boldsymbol{p}$.

Under extreme deformations of the path boundary, the optimizer may fail to find a next iteration were the elements are well-shaped. When this happens, we resample the boundary and isotropically remesh the patches.

6.1 C_2-continuous Relaxed Cloth Energy

In section 5.2, we introduced a relaxed energy model based on the Saint-Venant Kirchhoff hyper-elastic strain density function. For the optimization to converge, the sensitivity matrix $\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{p}}$ must be continuous, which implies that the cloth energy must be at least C_2 continuous. Unfortunately, as it is, the relaxed Stvk energy is only C_1 continuous. In our case, we choose to smooth the discontinuities.

A quick solution is to smooth the forces with respect to the principal stretches λ_1, λ_2 of the Cauchy strain tensor $\boldsymbol{C}$. We can identify three discontinuous transitions (Fig. 30): The jump from slack to wrinkled and from wrinkled to taut by the first principal stretch $\frac{\partial \psi}{\partial \lambda_1}$, and the jump from slack to wrinkled by the second principal stretch $\frac{\partial \psi}{\partial \lambda_2}$. Therefore, we can smooth the transitions from the perspective of each principal stretch.

Smoothing the forces We provide the procedure to smooth the forces with respect to the first principal stretch as an example. As mentioned, this force presents two discontinuities: The jump from slack to wrinkled, which meet at $\lambda_1 = 1$ and wrinkled to taut, which meet at $\lambda_1 = \frac{2\lambda + 2\mu - \lambda\lambda_2 - 2\mu\lambda_2}{\lambda}$ (obtained by clearing λ_1 from the energetic minimum of λ_2). We choose a smoothing epsilon ϵ and fit a quadratic function between the two regimes between the interval $[\lambda_1 - \epsilon, \lambda_1 + \epsilon]$ of the form $a\lambda_1^2 + b\lambda_1 + c$. We solve for the coefficients (a, b, c) for the two transitions by solving a linear system. We give the transition from slack to wrinkled as an example.

We find

$$\frac{\partial \psi}{\partial \lambda_1}_{\text{sl}\rightarrow\text{wr}} = a\lambda_1^2 + b\lambda_1 + c\,, \text{ such that} \tag{165}$$

$$a(1-\epsilon)^2 + b(1-\epsilon) + c = \frac{\partial \psi}{\partial \lambda_1}_{\text{slack}}, \tag{166}$$

$$a(1+\epsilon)^2 + b(1+\epsilon) + c = \frac{\partial \psi}{\partial \lambda_1}_{\text{wrinkled}}, \text{ and} \tag{167}$$

$$2a(1-\epsilon) + b = \frac{\partial^2 \psi}{\partial \lambda_1^2}_{\text{slack}}. \tag{168}$$

Notice that the 3 equations represent endpoints of the transitions for the forces and its derivatives. It can be seen that the endpoint for the wrinkled derivative is missing. Since we only have 3 degrees of freedom, 3 equations are needed for the system. Fortunately, the missing derivative is guaranteed and no further action is necessary. Notice also that the 3 regimes meet at the point $(\lambda_1, \lambda_2) = (1, 1)$. For this reason, we need a shrinking ϵ that vanishes at the point $(1, 1)$. Given that the transition from slack to wrinkled only depends on λ_1, we choose $\epsilon_{\text{sl}\rightarrow\text{wr}} = \frac{1}{2}\delta(\lambda_1 - 1)$, where δ is a small constant. For the transition between wrinkled and taut, we have to be careful to incorporate the special case where the Poisson ratio is 0. In such a case, the discontinuity is equal to the $\lambda_2 = 1$ line. Considering this, we choose the ϵ for that transition to be $\epsilon_{\text{wr}\rightarrow\text{ta}} = \frac{1}{2}\delta(1 - \lambda_2)$. A small value of δ should be chosen so the smoothed regions do not overlap each other. The point $(1, 1)$ is, unfortunately, left as a discontinuity.

Smoothing the energy In some cases, just smoothing the forces is not enough. In our case, we would like to use the energy as well, to take advantage of Newton. One quick attempt to obtain the energy, may be to integrate the transitions with respect to λ_1 or λ_2

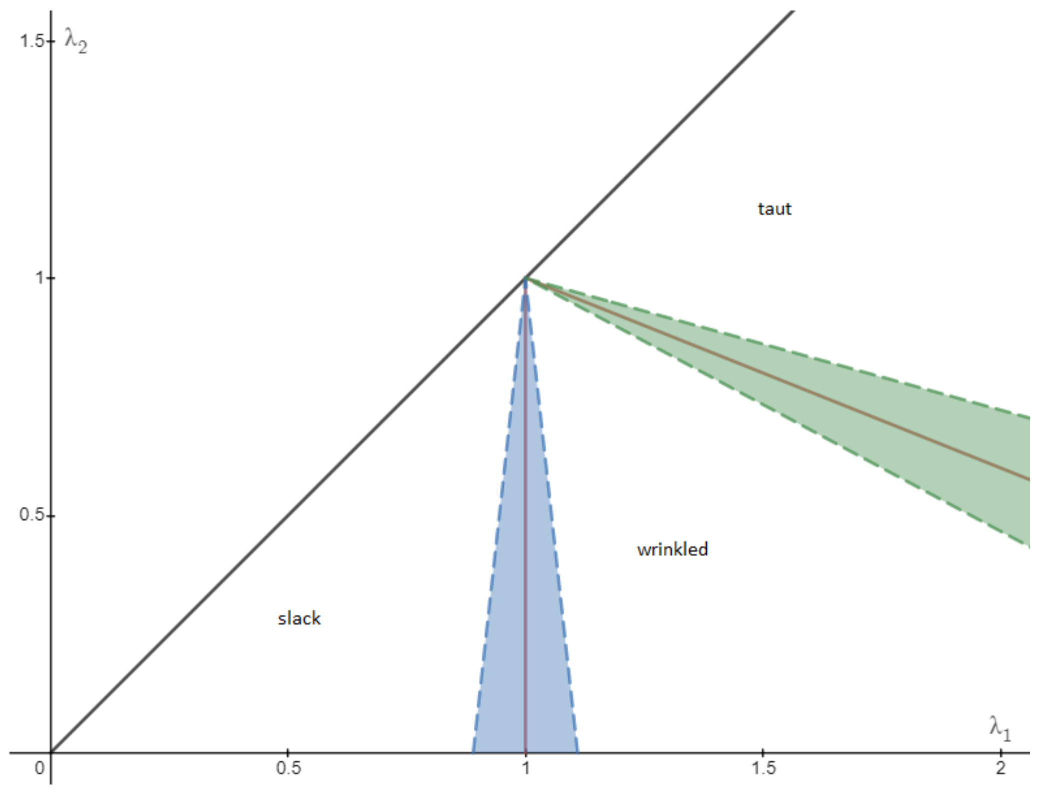


Figure 30: Discontinuities in the relaxed energy model and their smoothed regions. The black line represents the $\lambda_1 = \lambda_2$ line. The red lines represent the discontinuities. The blue area is the smoothed transition between slack and wrinkled. The green area is the smoothed transition between wrinkled and taut.

to obtain their respective energies. Unfortunately, it is not enough. By interpreting the energy as the area below the curve made by the force function (work), one can notice that the smoothed forces generate more work than the original un-smoothed ones. If we were to obtain the energy by integration, we would find that it does not coincide with the endpoints of the interpolated regimes. We introduce additional terms to the energy, to account for these discrepancies. These additions result in the modified energy regimes

$$W_{taut,shifted} = W_{taut} + s_1(\lambda_1 - 1)^2 + s_1(\lambda_2 - 1)^2, \text{ and} \tag{169}$$

$$W_{wrinkled,shifted} = W_{wrinkled} + s_1(\lambda_2 - 1)^2 + s_2(\lambda_1 - 1)^2, \tag{170}$$

where s_1 and s_2 are the additional terms meant to represent the shift caused by the interpolatory smoothing function. These additions satisfy that the taut energy remains symmetric with respect to the line $\lambda_1 = \lambda_2$. Such a property is important to maintain in the case the principal stretches switch places. Another important detail, is the dependence of the wrinkled energy to λ_1. While inconvenient, it is necessary to properly smooth the energy transitions. With these additions, now it is possible to smooth the energies, using the method to smooth the forces with the additional degrees of freedom s_1 and s_2.

6.2 Objectives

Our optimization framework allows for a wide range of objectives, we showcase our method using objectives inspired from fashion, medical garment and athletic garment industries.

While on-line sales gain dominance in most industries, the fashion industry lags behind primarily because of two main reasons: no tools to predict look and fit of a garment on a specific body. Due to the complex shape space of our bodies it is very difficult to find a proper fit without a physical try-on, and due to lack of a realistic virtual mirror, the customers cannot have a realistic preview before purchasing. Therefore, the fashion industry experiences a very large rate of return for on-line purchases. Our model can be used to optimize the patterns and the seam in order to achieve a comfortable fit as well as a specific shape of the body. We demonstrate our approach by automatically grading a given pattern layout to optimally fit diverse body shapes, computing patterns that shape body contours and minimize traction on seams.

In the case of medical garments the goal may be different such as minimize overall pressure as it is the case in diabetic wear or create a constant uniform pressure as it is the case in cosmetic surgery masks. In some sports-ware design such as running, cycling or swimming, the goal may be to produce the most aerodynamic or hydrodynamic shape of the body while wearing the garment. The remainder of this section goes over all the different objective examples.

6.2.1 Shape Objective

Within the limits set by physics and comfort, skintight clothing often provides room for shaping the underlying body. Our coupled model allows us to exploit this ability in our automated pattern design framework. For this purpose, we introduce a shape objective that measures the distance between the current deformed cloth and a given target shape. For better shape approximation, we avoid restrictive per-vertex correspondence and instead use a distance-field approach based on implicit moving least squares (IMLS) surfaces [41]. The corresponding objective is defined as

$$T_{\text{shape}} = \sum_i \sum_k \left(\frac{\sum_k \boldsymbol{n}_k \cdot (\boldsymbol{x}_i - \boldsymbol{c}_k)\phi(\|\boldsymbol{x}_i - \boldsymbol{c}_k\|)}{\phi(\|\boldsymbol{x}_i - \boldsymbol{c}_k\|)} \right)^2 , \tag{171}$$

where $\boldsymbol{c}_k$ are the vertices of the target shape and ϕ is a locally supported kernel function,

$$\phi(r) = \left(1 - \frac{r^2}{h^2}\right)^4 , \tag{172}$$

that vanishes beyond the cut-off distance h.

6.2.2 Stretch Objective

The amount of stretch that a garment experiences once worn is an important design consideration for skintight clothing. For instance, a tight fit can improve aerodynamic efficiency by reducing wind resistance in applications such as cycling. Then again, excessive stretch can cause material fatigue and reduce the lifetime of a garment. We therefore introduce an objective that allows designers to impose target values for minimum and maximum stretch. To this end, we first define a per-element objective as

$$T^e_{\text{stretch}}(\lambda^e_i) = \begin{cases} A^e(\lambda^e_i - \lambda_{\min})^2 & \lambda^e_i < \lambda_{\min} , \\ A^e(\lambda^e_i - \lambda_{\max})^2 & \lambda^e_i > \lambda_{\max} , \\ 0 & \lambda_{\min} < \lambda^e_i < \lambda_{\max} , \end{cases} \tag{173}$$

where are λ^e_i with $i = \{1, 2\}$ are the eigenvalues of the element's right Cauchy-Green tensor $\boldsymbol{C}_{\text{C}}$, $\lambda_{\min} \leq \lambda^e_i \leq \lambda_{\max}$ is the range of admissible stretch and A^e is the area of the undeformed element. The total objective simply sums up all per-element contributions as

$$T_{\text{stretch}} = \sum_e \sum_{i \in \{1,2\}} \left[T^e_{\text{stretch}}(\lambda^e_i)\right] .$$

It should be noted that, since $\boldsymbol{C}_{\text{C}}$ is a 2×2 matrix, its eigenvalues and their derivatives can be determined analytically.

In addition to defining the range of preferred stretch values, this objective also serves the purpose of penalizing wrinkled and slack elements. The latter are particularly troublesome as they can induce ill-conditioning in the Hessian at equilibrium, which causes problems for both simulation and optimization. To avoid these degenerate cases, we use the stretch objective for bounding the minimum stretch in all our examples.

6.2.3 Pressure Objective

Controlling the garment pressure on the body is an important feature for medical garments such as diabetic wear or post-surgery pressure masks used especially in cosmetic surgery. Therefore, we propose a design objective that allows the user a prescribed pressure range.

Pressure is defined as $\frac{F}{A}$ where F is the perpendicular force applied to a surface and A is the area on which that force is applied. In the context of a finite element simulation, pressure can be computed as the normal force acting on a vertex over the area that "belongs" to that vertex.

In our case, the resulting forces are known to be normal to the subdivision surface due to our requirement that the tangential forces be zero at equilibrium. However, computing accurately the area of a vertex in a piece-wise linear triangle mesh is less trivial. The obvious choice of having the area be the sum of a third of the area of its surrounding elements leads to often inaccurate and mesh-dependent behaviour. A better approach is given by using cotangent weights which alleviates some of the problems. But one should be restricted to using regular meshes as cotangent weights can lead to big variations for irregular vertices.

A better solution is achievable from fluids, in which the change of pressure is given by the Young-Laplace equation $\Delta p = \gamma H$, where H is the mean normal curvature of the surface and γ is the surface tension. This expression assumes that the interface is devoid of any shear stresses, which is true for fluids but not for membranes. The Young-Laplace equation is then generalized [64] for membranes by the expression

$$p = d\mathrm{tr}(\boldsymbol{\sigma}\boldsymbol{\Lambda}), \tag{174}$$

where $\boldsymbol{\sigma}$ is the surface stress tensor, $\boldsymbol{\Lambda}$ is the shape operator and d is the thickness of the membrane. Then, we can compute the interface pressure per cloth element where σ is the Cauchy stress tensor and $\boldsymbol{\Lambda}$ is the mid-edge normal shape operator [22], defined as

$$\boldsymbol{\Lambda} = \sum_{i=1,2,3} \frac{\frac{\theta_i}{2} + s_i\phi_i}{Al_i}\boldsymbol{t}_i \otimes \boldsymbol{t}_i, \tag{175}$$

where i is each edge of the element, l_i is the length of the edge, $\boldsymbol{t}_i$ is the normal vector to the edge with length l_i, A_i is the area of the element, θ_i is the average signed angle between the normals of the faces shared by the edge (known as the mid-edge normal). s_i is 1 for a half-edge and -1 for the opposite half-edge, ϕ_i are additional degrees of freedom that represent corrections to the angles that represent the mid-edge normals.

The ϕ are obtained through minimizing the energy

$$\psi = \sum_i A_i \mathrm{tr}(\boldsymbol{\Lambda}_i^2), \tag{176}$$

which is quadratic on the ϕ and can therefore be optimized by solving a single linear system $\frac{\partial^2\psi}{\partial\boldsymbol{\phi}^2}\Delta\boldsymbol{\phi} = -\frac{\partial\psi}{\partial\boldsymbol{\phi}}$ where $\frac{\partial^2\psi}{\partial\boldsymbol{\phi}^2}$ is unique and therefore non-singular for a sufficiently refined mesh.

Given the interface pressure, we can define a per element pressure objective for optimization as

$$T_{\text{Pressure}} = \sum_i \bar{A}_i(p - \mathcal{T})^2, \tag{177}$$

where $\bar{A}$ is the area of the undeformed element and $\mathcal{T}$ is a user specified target pressure. The gradient of this objective with respect to the undeformed vertices $\bar{\boldsymbol{x}}$ considering the additional degrees of freedom ϕ is

$$\frac{dT}{d\bar{\boldsymbol{x}}} = \frac{\partial T}{\partial \bar{\boldsymbol{x}}} + \frac{\partial \boldsymbol{x}}{\partial \bar{\boldsymbol{x}}}^T \left(\frac{\partial T}{\partial \boldsymbol{x}} + \frac{\partial \boldsymbol{\phi}}{\partial \boldsymbol{x}}^T \frac{\partial T}{\partial \boldsymbol{\phi}} \right), \tag{178}$$

where $\frac{\partial \boldsymbol{x}}{\partial \bar{\boldsymbol{x}}}$ and $\frac{\partial \boldsymbol{\phi}}{\partial \boldsymbol{x}}$ are obtained through sensitivity analysis as

$$\frac{\partial \boldsymbol{x}}{\partial \bar{\boldsymbol{x}}} = -\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}^{-1} \frac{\partial \boldsymbol{f}}{\partial \bar{\boldsymbol{x}}}, \tag{179}$$

where

$$\frac{\partial \boldsymbol{\phi}}{\partial \boldsymbol{x}} = -\frac{\partial^2 \psi}{\partial \boldsymbol{\phi}^2}^{-1} \frac{\partial^2 \psi}{\partial \boldsymbol{\phi} \partial \boldsymbol{x}}. \tag{180}$$

Instead of evaluating the entire sensitivity matrices $\frac{\partial \boldsymbol{x}}{\partial \bar{\boldsymbol{x}}}$ and $\frac{\partial \boldsymbol{\phi}}{\partial \boldsymbol{x}}$, we use adjoint sensitivity analysis for both corresponding terms, which amounts to an additional linear solve per gradient evaluation.

Unlike the discrete pressures based on per-vertex areas, our new approach leads to accurate pressure distributions even for unstructured meshes with many irregular vertices and a wide range of element aspect ratios; see Fig. 31 for a comparison.

6.2.4 Seam Stress Objective

The most vulnerable areas of sewn garments are typically along the seams. It is therefore a natural goal to optimize for patterns that minimize seam stress and thus increase garment life span and reliability.

For each cloth element adjacent to a seam, we evaluate the second Piola-Kirchhoff stress tensor $\boldsymbol{S}_i$ [9], which relates traction forces to areas in the undeformed configuration, and compute the traction in the direction $\boldsymbol{n}$, perpendicular to the seam. We then penalize seam

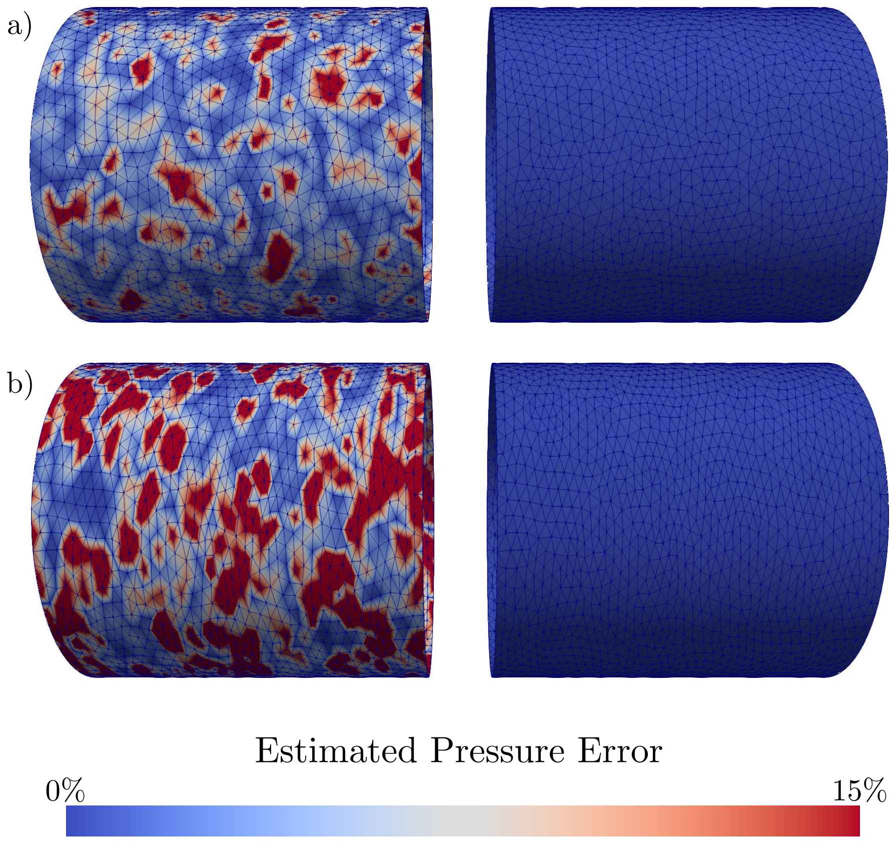


Figure 31: Estimated pressure error on a cylindrical cloth stretched around the circumference of a cylinder of radius 1 with a Poisson's ratio equal to 0. For such an example, the pressure is constant everywhere. a) Pressure error estimated on elements stretched by 33% using $\frac{F}{A}$ (left) and Young-Laplace with the midedge normal operator (right). b) Pressure error estimated on elements stretched by 100% using $\frac{F}{A}$ (left) and Young-Laplace with the midedge normal operator (right). It can be observed that the shape of the elements influences the accuracy of the pressure computed with $\frac{F}{A}$ while pressure computed with Young-Laplace with the midedge normal operator remains accurate.

traction as

$$T_{Seam} = \sum_{i \in \mathcal{T}} l_0^i d \boldsymbol{S}_i \boldsymbol{n}_i \cdot \boldsymbol{n}_i \,, \quad \text{with} \quad \boldsymbol{S}_i = 2 \frac{\partial \phi^i_{\text{cloth}}}{\partial \boldsymbol{C}_{\text{C}}} \,, \tag{181}$$

where $\mathcal{T}$ is the set of all seam-adjacent elements, l_0^i is the length of the seam in the undeformed configuration, and d is the thickness of the fabric.

6.2.5 Multiple Poses Sliding Objective

While the objectives introduced so far all refer to a single body pose, they can readily be extended to incorporate multiple poses. However, an additional aspect that arises when considering a range of motion instead of a single pose is garment sliding. Running and cycling, for instance, are applications where sustained tangential motion of the garment relative to the body can lead to discomfort and even injury.

Motivated by this example, we introduce a design objective that aims at minimizing garment sliding for a given range of motion. To this end, we start by defining a neutral body pose $\boldsymbol{y}_0$ and a set of target poses $(\boldsymbol{y}_1, \ldots, \boldsymbol{y}_n)$ that represent the range of motion to be considered. We first compute the equilibrium configuration $\boldsymbol{q}_0$ for the main pose and use the resulting local coordinates $\boldsymbol{x}_0$ to determine deformed cloth positions $(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$ for each target pose. In the absence of friction, these configurations are generally not in equilibrium and would lead to cloth sliding over the body. By accounting for friction, however, we can determine whether the unbalanced tangential forces can be compensated by friction forces.

To this end, we implement a simple, isotropic Coulomb-type friction model and define tangential and friction forces magnitudes for each cloth vertex j and each pose i as

$$f^t_{i,j} = ||\boldsymbol{f}_{i,j} - (\boldsymbol{f}_{i,j} \cdot \boldsymbol{n}_{i,j})\boldsymbol{n}_{i,j}|| \,, \quad \text{and} \quad f^c_{i,j} = \mu \boldsymbol{f}_{i,j} \cdot \boldsymbol{n}_{i,j} \,,$$

respectively, where $\boldsymbol{f}_{i,j}$ is the elastic force from the cloth, $\boldsymbol{n}_{i,j}$ is the normal given by the subdivision surface and μ is the friction coefficient. To penalize excessive tangential forces that would lead to sliding, we define the objective

$$T^i_{Sliding} = \sum_i (T_{i,j})^2 , \quad T_{i,j} = \begin{cases} f^t_{i,j} - f^c_{i,j} & f^t_{i,j} \geq f^c_{i,j} \\ 0 & \text{otherwise} . \end{cases} \tag{182}$$

The gradient for this objective depends only on the local coordinates of the neutral pose and is obtained as

$$\frac{dT_{Sliding}}{d\boldsymbol{p}} = \sum_i \frac{\partial T^i_{Sliding}}{\partial \boldsymbol{p}} + \left(\frac{\partial \boldsymbol{x}_i}{\partial \boldsymbol{x}_0} \frac{\partial \boldsymbol{x}_0}{\partial \boldsymbol{p}}\right)^T \frac{\partial T^i_{Sliding}}{\partial \boldsymbol{x}_i} , \tag{183}$$

where $\frac{\partial \boldsymbol{x}_0}{\partial \boldsymbol{p}}$ is the sensitivity matrix of the neutral pose, and $\frac{\partial \boldsymbol{x}_i}{\partial \boldsymbol{x}_0}$ maps changes in local coordinates for the neutral pose to corresponding world-space changes for the target poses.

6.3 Regularizers

In addition to the design goals that we describe in Sec. 6.2, the objective function T in Eq. (158) also includes several regularizers that ensure well-shaped patterns and manufacturability.

Seam Compatibility Connecting two patches in a given seam requires the corresponding patch boundaries to have the same length. To enforce this compatibility condition during optimization, we penalize length deviations for corresponding edges $\mathbf{e}_{\text{left}}$ and $\mathbf{e}_{\text{right}}$ from different sides of the seam have as

$$R_{\text{SeamLen}} = \sum_i \left[\left(\left\| \mathbf{e}^i_{\text{left}} \right\|_2 - \left\| \mathbf{e}^i_{\text{right}} \right\|_2 \right)^2 \right] . \tag{184}$$

Patch Boundary We generally prefer patch boundaries that are smooth and, excepting corners, discourage sharp features with a penalty term based on the discrete bending energy from [7] as

$$R_{\text{Smooth}} = \sum_{(i,j)\in\mathcal{B}} \frac{\kappa_{ij}^2}{\|\mathbf{e}_i\| + \|\mathbf{e}_j\|}, \tag{185}$$

where $\mathcal{B}$ is the set of pairs of consecutive edges on the pattern boundary with integrated curvature

$$\kappa_{ij} = \frac{2\mathbf{e}_i \perp \mathbf{e}_j}{\|\mathbf{e}_i\| \|\mathbf{e}_j\| + \mathbf{e}_i \cdot \mathbf{e}_j}.$$

To prevent too close approach between neighboring boundary vertices, we furthermore use the energy defined in Eq. (184) to penalize differences in length between two consecutive edges.

Patch Compactness To prevent patterns from becoming arbitrarily thin during optimization, we introduce a compactness term that penalizes small ratios of Euclidean and on-boundary distance between two boundary vertices. Since for any pair of boundary vertices there are two possible on-boundary paths, we choose the initially shortest one and keep it for the remainder of the optimization. For two given boundary vertices $\boldsymbol{p}_i$, $\boldsymbol{p}_j$ we compute this ratio as

$$\rho_{ij} = \frac{\|\boldsymbol{p}_i - \boldsymbol{p}_j\|_2}{\sum_{k=i+1}^{k=j} \|\boldsymbol{p}_k - \boldsymbol{p}_{k-1}\|_2}, \tag{186}$$

where $\boldsymbol{p}_k$ represent vertices in the path from $\boldsymbol{p}_i$ to $\boldsymbol{p}_j$. The corresponding penalty term is defined as $R_{\text{Comp}} = \sum_i \sum_j R_{\text{Comp}}^{ij}$, where

$$R_{\text{Comp}}^{ij} = \begin{cases} (\rho_{ij} - r_D)^4 & \rho_{ij} \leq r_D \\ 0 & \rho_{ij} > r_D \end{cases}, \tag{187}$$

where r_D is a threshold modeling the minimum admissible ratio.

Forces on Boundary When designing skintight clothing, it is often desirable to prescribe the location of the garment boundaries relative to the body. Examples include the top of the waistband for a pair of pants, or the boundaries of a wet-suit close to the knee. To implement these constraints, we fix vertices on the boundary of the garment to corresponding target locations on the body. However, doing so can lead to large tangential forces on the garment boundary. We therefore encourage traction-free boundaries through the penalty term

$$R_{\text{Fixed}} = \sum_{i \in \mathcal{F}} \|\mathbf{b}_i\|_2^2 \quad \text{where} \quad \mathbf{b}_i = \frac{dU}{d\boldsymbol{s}_i}\,, \tag{188}$$

and $\mathcal{F}$ is the index set of fixed boundary vertices.

Chapter 7

Results and Discussion

We demonstrate our optimization-driven pattern design method on a set of examples inspired from fashion, sportswear, and medical garment applications. We start with examples that highlight the impact of individual design objectives, then proceed to further evaluation and performance data.

7.1 Impact of Design Objectives

The design objectives can be combined arbitrarily as required by the application. All our examples use the principal stretch objective in order to discourage the formation of slack elements. However, we choose to activate only one additional objective per example in order to provide a clearer impression of their individual impact.

Shape Objective We demonstrate our shape objective on two examples. The first one, shown in Fig. 32, uses a cylindrical shape as the initial body pose (Fig. 32a-top) and two target shapes: a conically tapered cylinder (Fig. 32b) and an hourglass shape (Fig. 32c). For both of these examples the cloth is composed of two rectangular patches (Fig. 32a-bottom) which conform to the surface of the cylinder. The optimized patterns are shown

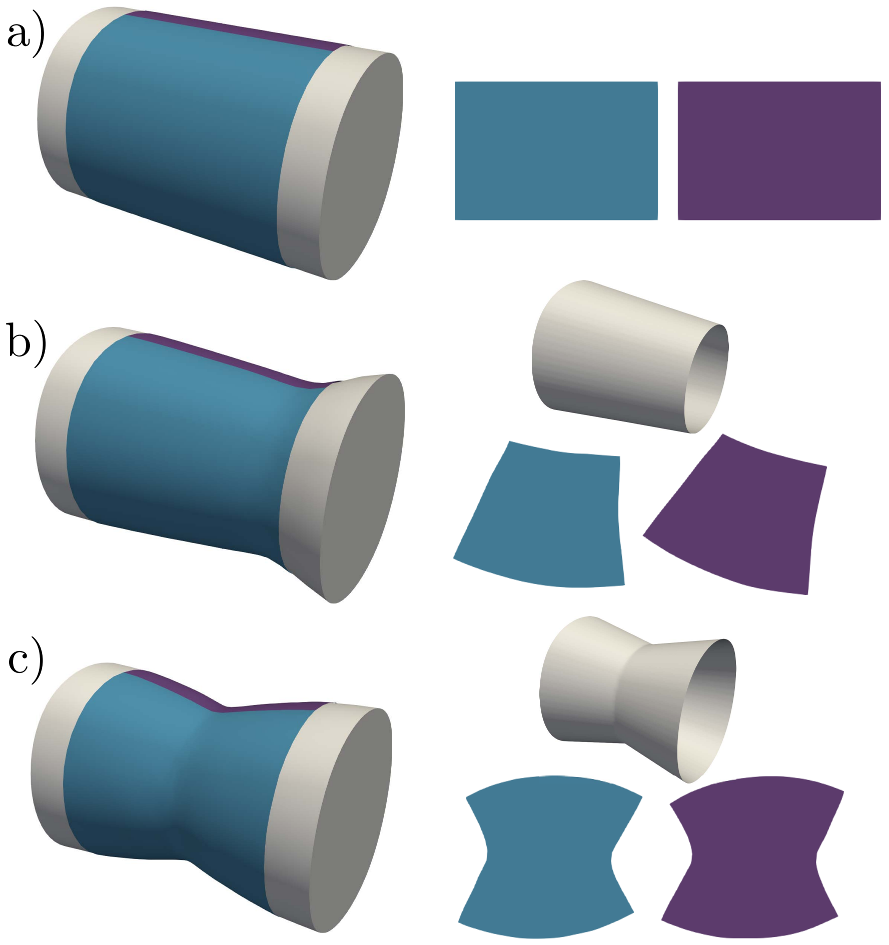


Figure 32: Pattern optimization for a deformable cylinder based on shape targets. a) Rest shape of the cylinder with two patterns. b) Optimization result for a tapered target. c) Optimization result for a hourglass target.

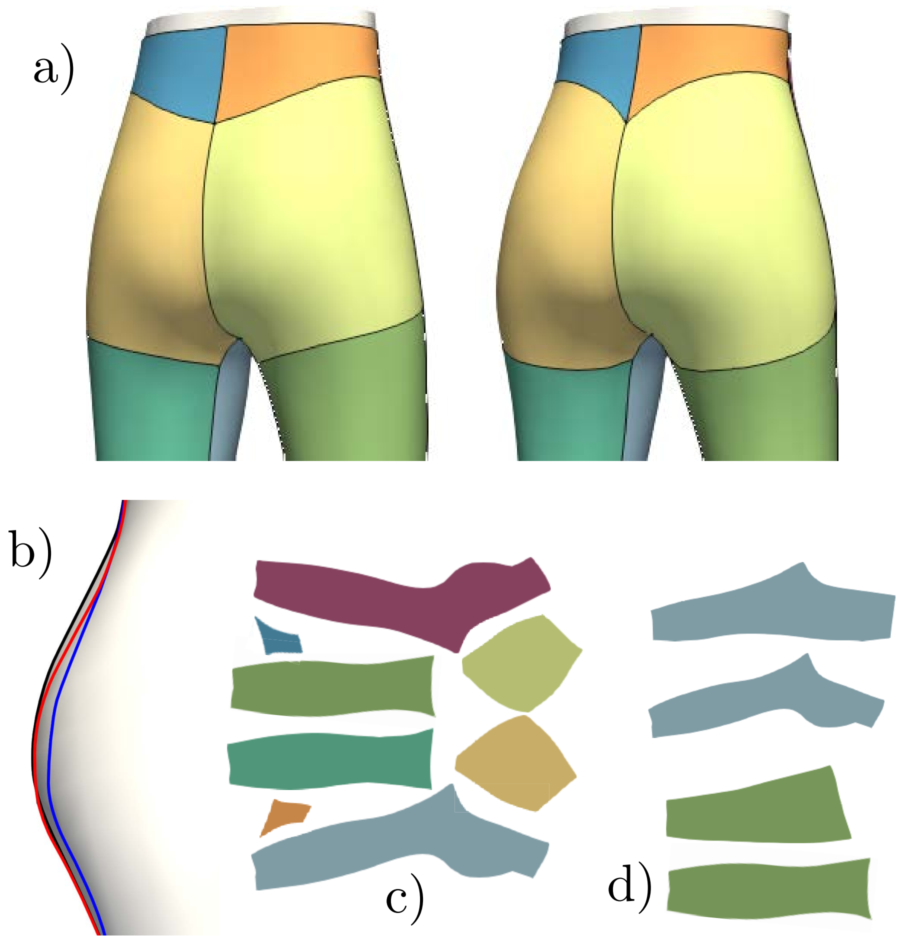


Figure 33: Pattern optimization for a pair of tight pants with a given target shape. a) Comparison before optimization (left) and after optimization (right). b) Silhouette before optimization (blue), after optimization (red), compared to the target (black). c) Patterns after optimization. d) Comparison between two individual patches before and after optimization

in Fig. 32b,c-bottom, and the corresponding simulation results indicate that optimizing patterns with our shape objective is an effective means of controlling body deformations. It should be noted, however, that the space of physically-feasible body deformations is fairly constrained as, e.g., volume changes cannot be achieved in this way. Nevertheless, our method is able to approximate mostly feasible target shapes with good accuracy.

The second example for the shape objective inspired by shapewear applications. Fig. 33 shows a pair of pants with its pattern atlas before and after optimization. The difference between the target shape, initial simulation result, and simulation result after optimization

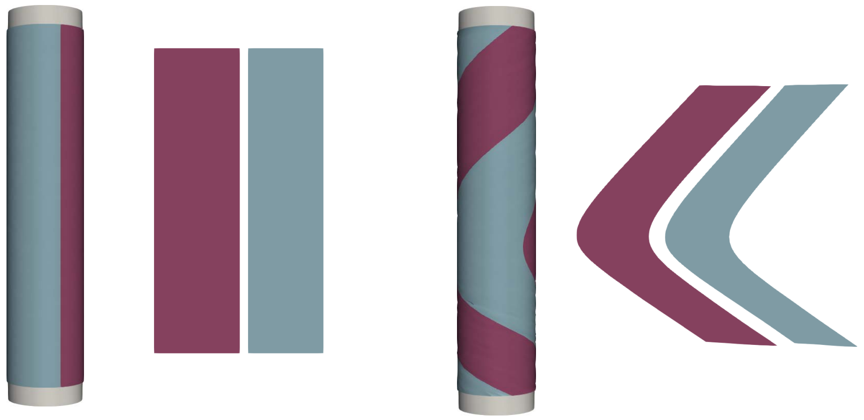


Figure 34: Cylindrical cloth stretched in the direction of the cylinder. The image shows the initial cloth with its patterns (left) and its optimized version using the seam traction objective (right).

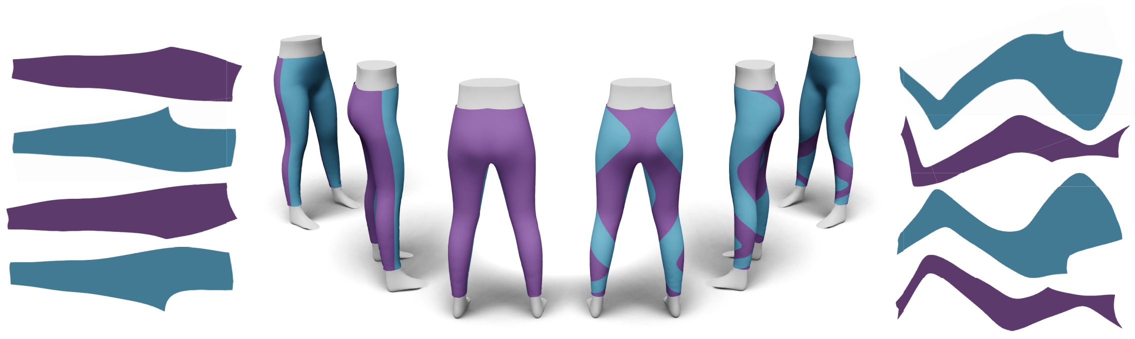


Figure 35: Pants stretched in the direction of the circumference of the legs. The image shows the initial cloth with its patterns (left) and its optimized version using the seam traction objective (right).

is show in Fig. 33b. While the changes in geometry are perhaps less obvious than in the previous example, the changes in pattern shape are substantial and the corresponding simulation result closely approximates the target shape.

Seam Traction Objective To demonstrate the impact of our seam traction objective, we first show it on a cloth cylinder stretched around the circumference. Since the seam is completely orthogonal to the direction of the stretch, the optimization tries to find patterns in a way that balances the length of the seam (the longer the seam, the stronger the total seam traction), and the angle between the direction of stretch and the orientation of the

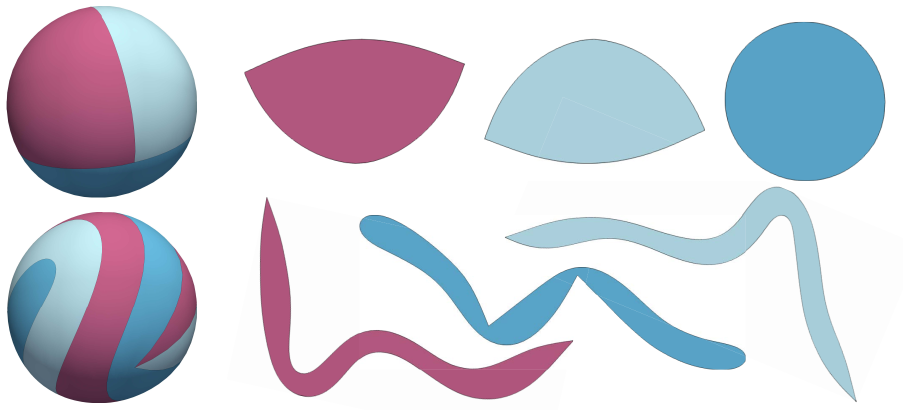


Figure 36: Patterns on a sphere change dramatically while optimizating for uniform pressure (Target pressure is set to 1000 $\frac{N}{m^2}$).

seam (the more parallel the seam is to the stretch direction, the weaker the traction). We show the result in Fig. 34

In a more practical example, we use it for optimizing the patterns of a pair of long pants as shown in Fig. 35. We start with an initial pattern set that leads to about 30% stretch along the circumference of the legs. Since the seams are initially perpendicular to the direction of maximum stretch, they experience excessive traction forces. We then optimize the patterns using our seam-traction objective as well as the stretch objective in order to maintain the initial deformation. The optimization yields a pattern layout that leads to helical seams spiraling around the legs. The increased length and changed orientation leads to an overall improvement of 13% in traction force density. The optimized design thus reduces the risk of material failure while offering an aesthetically-interesting seam layout.

Pressure Objective To evaluate the effectiveness of our pressure objective, we start with a simple example in which we optimize the patterns for a sphere such that the resulting pressure is as close as possible to $1000N/m^2$ everywhere.

As can be seen in Fig. 36, the patterns change drastically during optimization and converge to elongated, winding shapes whose seams form a complex interlocking pattern on the 3D surface. Interestingly, this result is very similar to the one obtained by Skouras

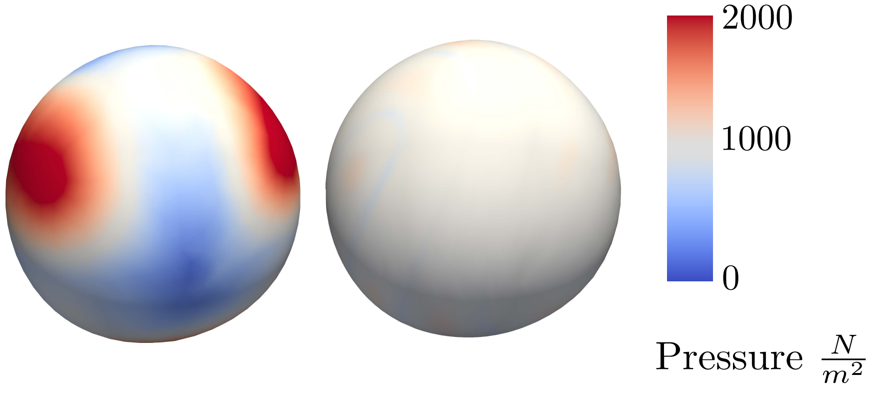


Figure 37: Pressure on the sphere before optimization and after optimization.

et al. [52], who optimized patterns such that the inflated shape is as spherical as possible. It is not surprising then that optimizing for constant pressure yields patterns that result in an almost spherical shape. The resulting pressure is sown in Fig. 37.

Our second example for the pressure objective is a post-surgery compression mask consisting of two patterns. We set the admissible range of pressure to $1250 - 3000\ N/m^2$, which is consistent with the target pressure of medical surgery masks. As shown in Fig. 38, the simulation result for the initial patterns exhibits excessive pressure for high-curvature regions such as the chin or the top of the head. Furthermore, there are many elements around the neck that exhibit negative pressure (shown in black), which occurs when cloth is stretching in concave regions. It should be noted that negative pressure is an artefact of our model and, rather than pulling on the body, the fabric would lift off the surface in reality. Nevertheless, it is an effective indicator for this problem and by penalizing negative pressure during optimization, we can prevent undesirable fabric lift-off. This effect can be observed in Fig. 38-right, where the optimized patterns achieve pressure values for sensitive areas within the desired range. While some elements with negative pressure remain in concave regions such as the temples or the hollows of the cheeks, their number is largely reduced.

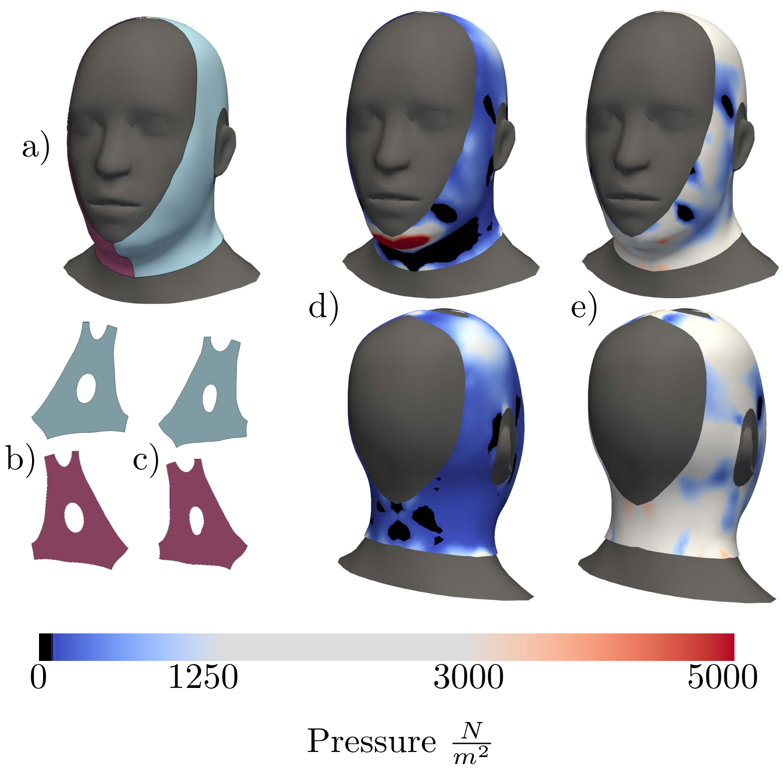


Figure 38: Optimization of a pressure mask to a given interval [1250-3000]. a) Patterns of the mask in 3D, b) Patches before optimization, c) Patches after optimization, d) Pressure distribution before optimization, e) Pressure distribution after optimization

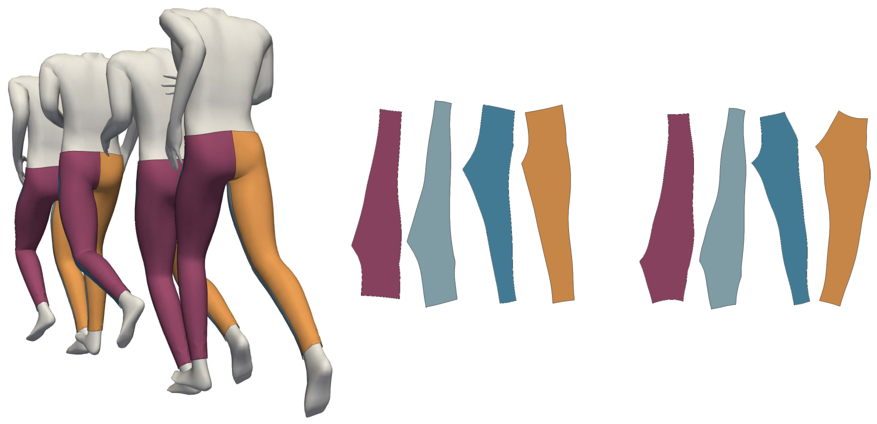


Figure 39: Running sequence used to optimize a pair of pants (left), Patterns before optimization (middle), patterns after optimization (right)

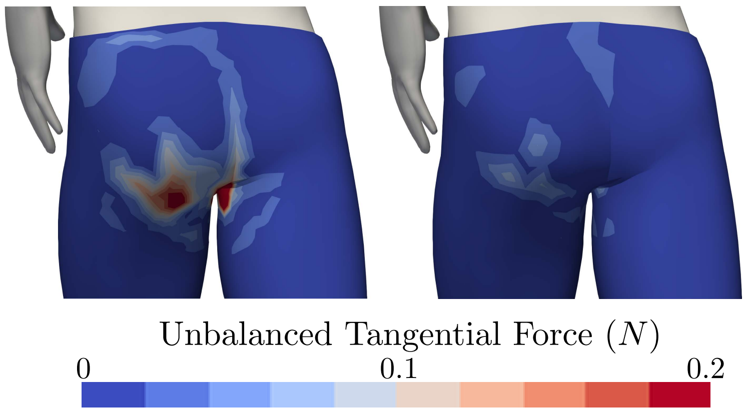


Figure 40: Unbalanced tangential forces before optimization (left) and after optimization (right)

Sliding Objective We demonstrate the effectiveness of our sliding objective on a pair of long pants and four poses selected from a running sequence as seen in Fig. 39. We again prescribe a stretch target in the circumferential direction of the legs of 30% and set the friction coefficient to $\mu = 0.3$. As can be seen in Fig. 40, the pelvic area exhibits substantial tangential forces which would translate into unwelcome sliding during the motion. The optimized patterns lead to greatly reduced unbalanced tangential forces.

7.2 Additional Evaluation

Impact of Seams By accounting for seam stiffening in simulation, our method can anticipate the corresponding effects during pattern optimization. We demonstrate the impact of seams the example shown in Fig. 41. We simulate an initial pattern set (Fig. 41c) on a deformable sphere without seam stiffening, resulting in a shrunk version of the sphere (Fig. 41a). However, when the same patches are simulated again with added seam stiffness, the resulting shape exhibits clearly visible deformations around the seams (Fig. 41b). When optimizing the patterns with seam stiffness, using the initially deformed sphere as target (Fig. 41d), our method adjusts the patterns such that the resulting shape approximates well

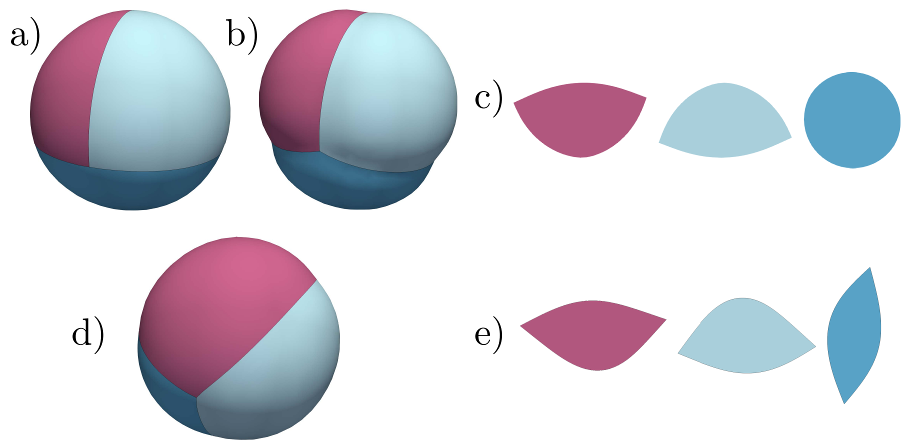


Figure 41: Example of optimizing a set of patterns on a compressible sphere. An initial simulation without seam stiffening produces the result shown in a). After taking seam stiffening into consideration, we obtain the result shown in b). The simulations presented in a) and b) use the patterns shown in c). After optimizing to obtain the same result of a) considering seam stiffening, we obtain the simulation shown in d), with the patterns shown in e).

the target (Fig. 41e).

Automatic Grading & Body Shape Variety One central advantage offered by our optimization-driven approach over manual pattern design is that a given pattern layout can be automatically customized to a variety of body shapes. We demonstrate this ability by optimizing the patterns of wet-suit design, shown in Fig. 42, for four different body shapes using the same objective (30% target stretch) in all cases. Our method automatically grades the patterns for each body shape such that the target stretch is maintained (Fig. 43).

Impact of Regularizers So far, we have shown results that focus solely on the design objectives. To achieve some of these results, the regularizers played a crucial role. In this section we demonstrate how two of these particular regularizers, curvature (Eq. 185) and compactness (Eq. 187), influenced the final results in two of the examples whose patterns experienced the most drastic deformations: the pressure sphere (Fig. 36) and the seam traction pants (Fig. 35).

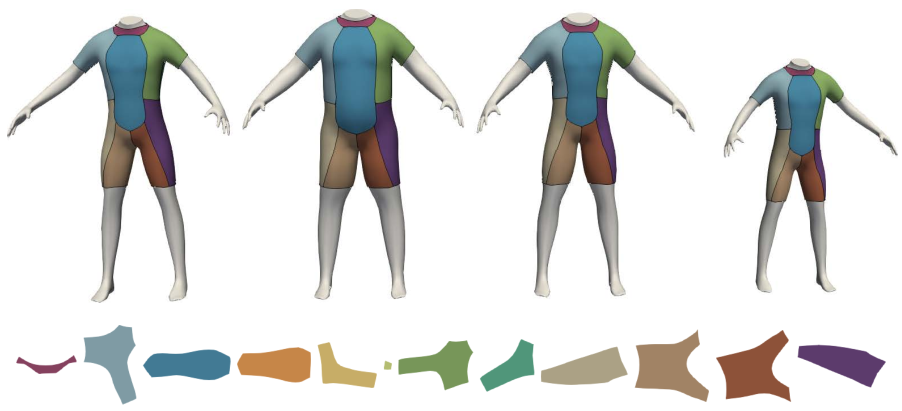


Figure 42: A wetsuit adapted for different body meshes (Optimized patterns shown for the body to the left)

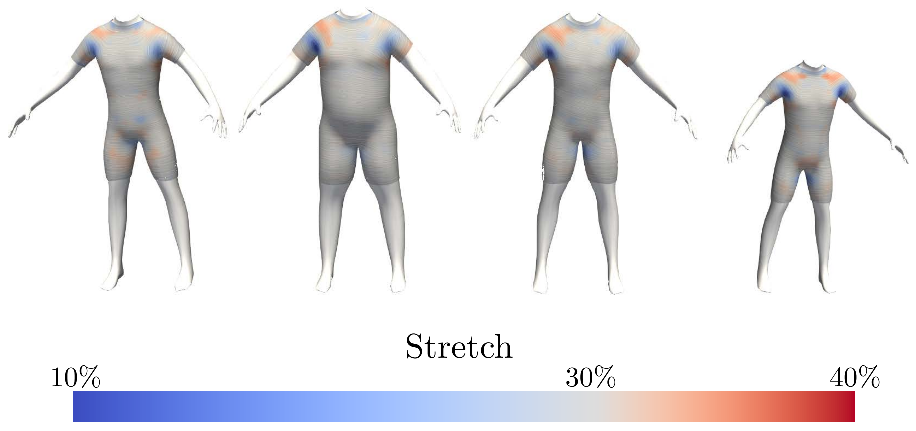


Figure 43: Resulting stretch after optimization for each body shape (streamlines represent direction of the stretch)

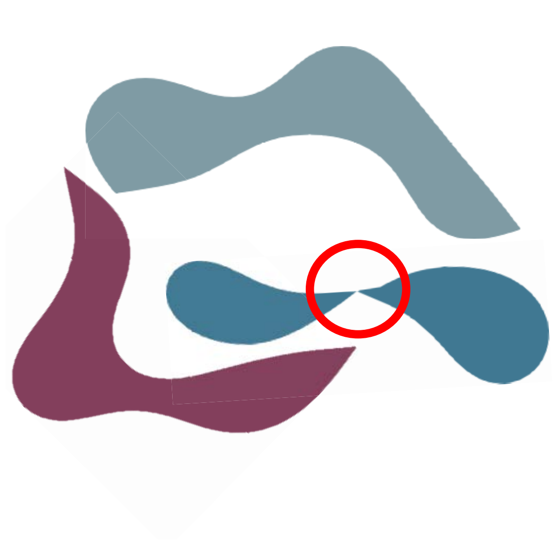
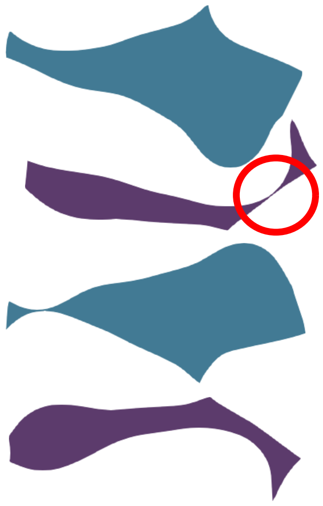



Figure 44: Patterns without the compactness regularizer tend to collapse on themselves. Patterns presented for a failed optimization for the sphere (left) and for the pants (right)

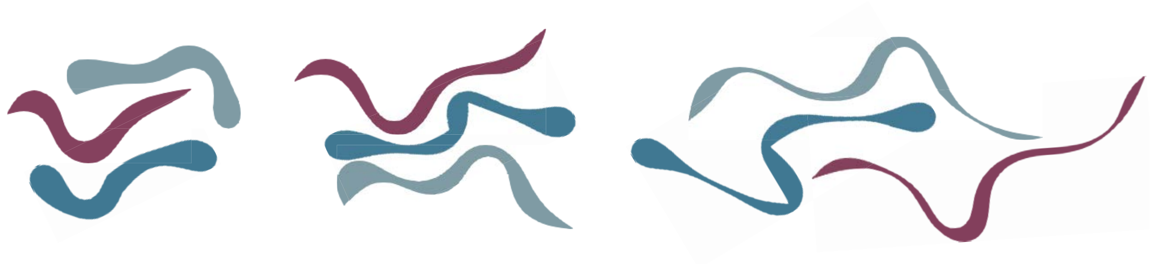


Figure 45: Sphere pressure optimization with varying ratios for the compactness term. Left: 10%, Middle 5%, Right 2.5%

We start by underlying the importance of the compactness term to even get a converged a result. Without it, there is not an incentive for the patterns borders to collapse on themselves. An example of this artefact is shown in Fig. 44. The compactness term not only allows us to avoid these kinds of collisions, but also allows us to specify arbitrary thinness for the patterns without compromising robustness. This is shown in Fig. 45. In a similar way, for high values for the curvature regularizer, the patterns tend to prefer straighter seams, while low curvature penalizations tend to allow more freedom in the designs. An example of this behaviour is found in Fig. 46, where the seam traction optimization example for the pants is run with different curvature penalization weights.

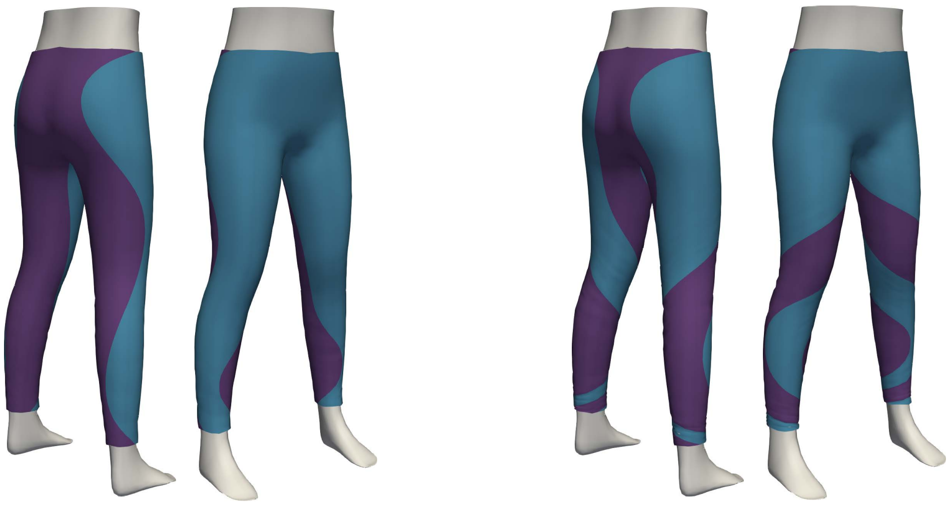
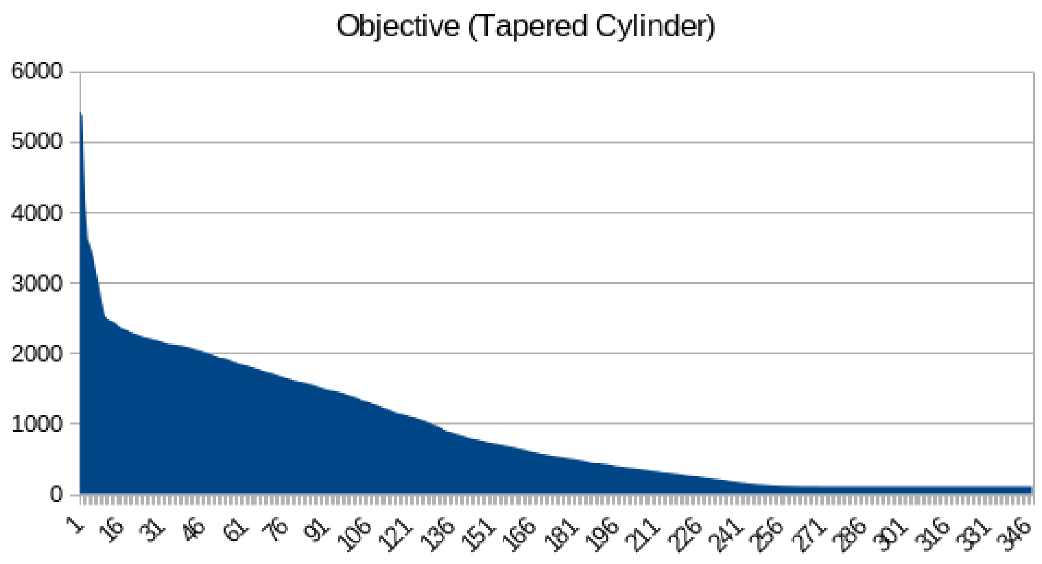


Figure 46: Seam traction optimization for the pants under two different curvature penalization weights. High curvature weight (Left) against Low curvature weight (Right).

Performance and Statistics In all our results, we start from an initial guess obtained by scaling down the patterns computed with a geometric flattening method [50]. By scaling down patterns, we avoid slack elements at the start of the optimization. Each simulation is considered converged once the norm of the unbalanced forces falls below 10^{-6}. The optimization is considered converged once the norm of the objective gradient falls below 10^{-3}. We ran our examples on a machine with an Intel Core i7-5820k processor and 8GB of RAM. In terms of material parameters, we use a Young modulus of $24.78KPa$ and a Poisson's ratio of 0.49 for the body. For the cloth, we chose a thickness of $0.1mm$ and a Young modulus of $30MPa$ for the mask, $5.4MPa$ for the pants with a Poisson's ratio of 0.33 for all examples. Statistics for selected experiments are listed in Table 1. Additionally, we include convergence graphs (objective and gradient) for two selected experiments.



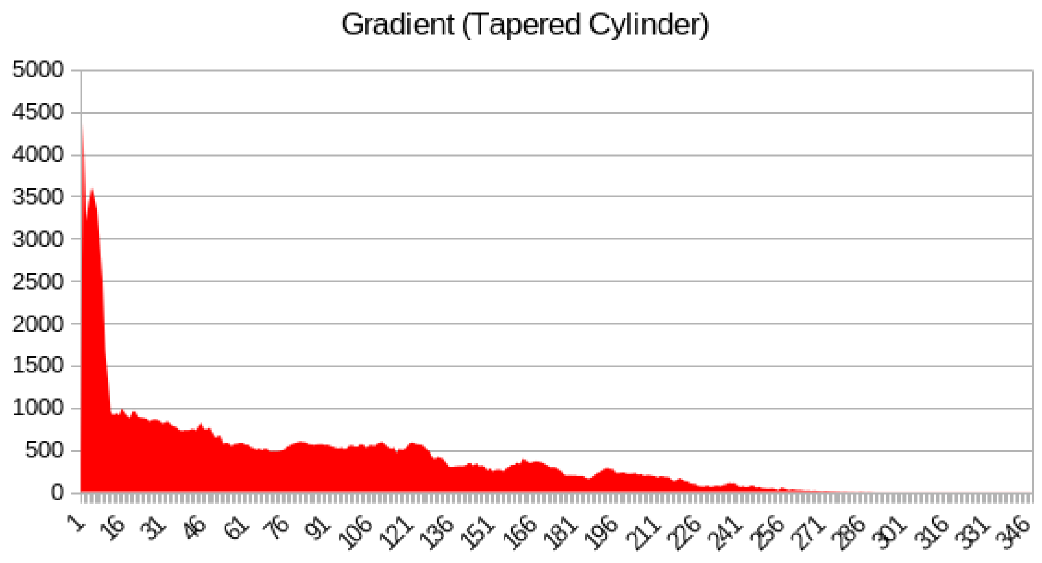
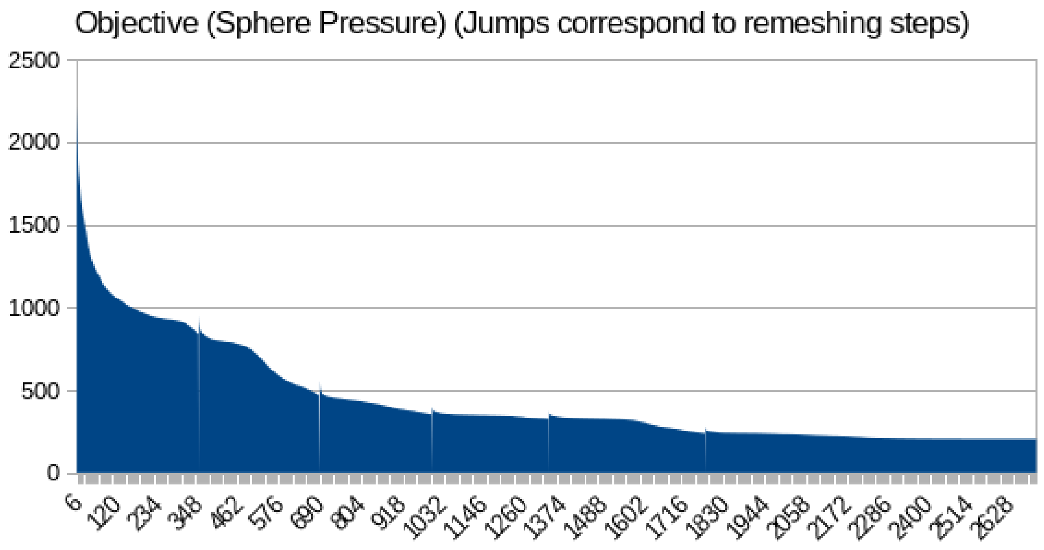


Figure 47: Convergence graphs for the tapered cylinder.



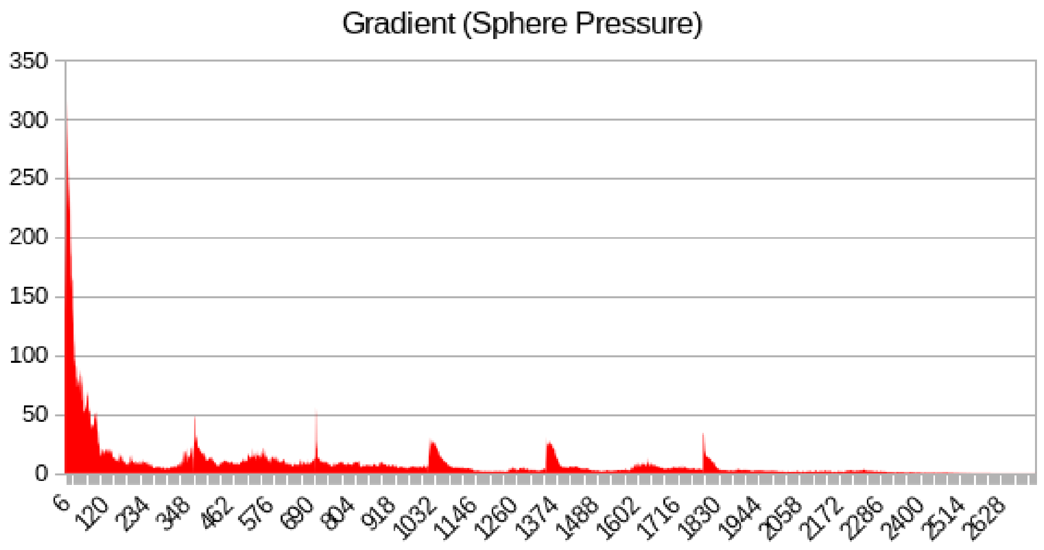


Figure 48: Convergence graphs for the pressure sphere (Example with multiple remeshing steps).

Table 1: Summary of parameters and performance for all our experiments. From left to right: Model, Number of Patterns, Number of Remeshing Steps, Number of Iterations, Time per Iteration [s], Initial Objective Value, Final Objective Value.

Cylinder tapered (Fig. 32b)	2	0	346	12.2	5433.58	106.95
Cylinder hourglass (Fig. 32c)	2	0	386	10.1	2241.66	222.64
Pants stretch (Fig. 35-left)	4	0	903	6.34	133.06	5.70
Pants traction (Fig. 35-right)	4	1	4160	5.87	38.62	28.45
Pants shape (Fig. 33)	8	0	1408	19.2	66.91	14.40
Sphere pressure (Fig. 36)	3	5	2714	4.66	9547.41	205.54
Sphere shape (Fig. 41)	3	0	1001	22.3	22524.6	217.66
Mask pressure (Fig. 38)	2	0	849	6.40	55.44	1.55
Sliding pants (Fig. 39)	4	0	914	7.48	120.57	5.23
Wetsuit regular (Fig. 42)	12	0	1629	5.84	130.22	19.70
Wetsuit athletic (Fig. 42)	12	0	1735	5.57	139.74	21.15
Wetsuit overweight (Fig. 42)	12	0	1353	6.60	177.55	25.81
Wetsuit kid (Fig. 42)	12	0	1561	5.72	115.96	18.35

Chapter 8

Conclusions, limitations and future work

We presented an optimization-driven approach to automatically generate patterns for skintight clothing. As the core of our method, we proposed a computational model that captures the mechanics of the clothing, the underlying body, and their mutual interaction within a unified approach. We furthermore described a set of design objectives that encode shape, comfort, and mechanical aspects of the garments. Our results indicate that our approach is able to reliably compute optimal patterns for a broad range of garments and body shapes, even when substantial design changes are required.

Our current approach has several limitations, most of which indicate promising directions for future work. Embedding the garment in the surface of the body simplifies both simulation and optimization. Nevertheless, it comes at the cost of introducing incorrect behaviour for concave surfaces: whereas real cloth will lift off the body when stretched over a concave region, in our model it generates traction forces that pull the body outwards as illustrated in Fig. 49. While we can detect these situations and optimize patterns to avoid negative pressure, not all cases can be resolved in this way. In the future, it would interesting to combine our approach with a conventional cloth model that is activated for regions in which the cloth separates from the body.

All our examples use a single isotropic cloth material for all patterns. Natural extensions

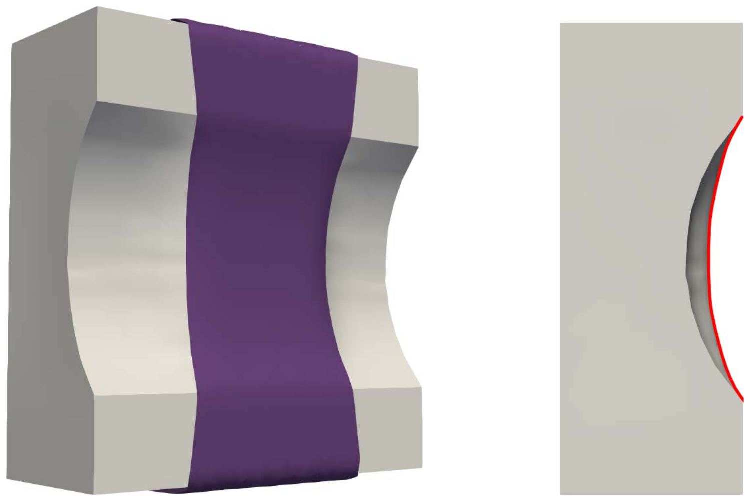


Figure 49: Stretched cloth on a concave surface (*left*) results in pulling forces that can produce artifacts in the body (*right*).

include accounting for material anisotropy, and to combine patterns with different materials for shaping and reinforcement.

We use subdivision surfaces to convert the piece-wise linear boundary of the body mesh into a continuous surface. While the improved smoothness facilitates simulation and optimization, we do currently not use the same representation for simulating the body. Increasing the resolution of the body mesh decreases the discrepancy between these representations, but a more elegant solution would be to use subdivision finite elements [12]. As a related limitation, the cloth mesh should have a higher resolution than the body mesh, since it could otherwise lead to body vertices not experiencing any coupling force.

Finally, while our method allows for bounding the maximum stretch in a garment once worn, we do not take into account deformations that occur during dressing. Especially for tight-fitting garments made of stiffer fabrics, this question can have an important impact on the design. Integrating dressing simulation into the design process is an interesting direction, and the work of Clegg et al. [16] seems a good starting point.

Appendices

Appendix A

Loop Subdivision Surfaces

Loop is a method of approximating subdivision surfaces to a linear triangular mesh. It works by recursively adding vertices to the middle all the edges of the mesh, creating 4 new faces for each face in the process. Each original vertex is then smoothed as the average of the surrounding new vertices. It is said to be approximating, as the resulting subdivided surface does not interpolate the original surface.

If this process is carried indefinitely, it will eventually reach a converged mesh which is known as the limit surface. It is possible to evaluate a given point on the limit surface along with its derivatives as long as the triangle containing the point has at most one irregular neighbour [55].

The evaluation of the limit surface of a point lying on an element depends on the regularity of the 3 vertices of the element. A vertex in a triangular mesh is said to be regular if it has exactly 6 neighbours. If all vertices in an element are regular, the limit point can be computed directly from the 3 triangle vertices and the 9 (fig. 1) direct neighbours of those

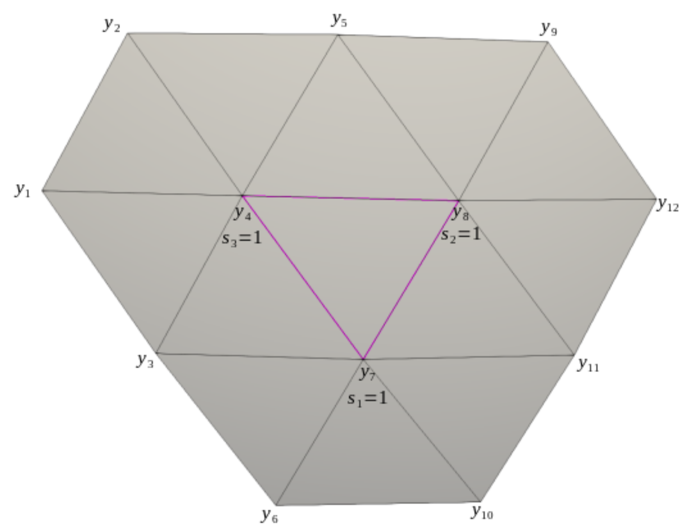


Figure 1: Regular patch with ordering for a limit point lying on the triangle made by vertices $\{\boldsymbol{y}_4, \boldsymbol{y}_7, \boldsymbol{y}_8\}$.

vertices as

$$\boldsymbol{x}^T = \frac{1}{12}\boldsymbol{b}^T \begin{bmatrix} y_{1x} & y_{1y} & y_{1z} \\ y_{2x} & y_{2y} & y_{2z} \\ y_{3x} & y_{3y} & y_{3z} \\ & \cdots & \\ y_{12x} & y_{12y} & y_{12z} \end{bmatrix}, \tag{189}$$

where $\boldsymbol{b}$ is a vector of basis functions defined as

$$
\begin{aligned}
b_1 &= s_3^4 + 2s_3^3s_1 \\
b_2 &= s_3^4 + 2s_3^3s_2 \\
b_3 &= s_3^4 + 2s_3^3s_2 + 6s_3^3s_1 + 6s_3^2s_1s_2 + 12s_3^2s_1^2 + 6s_3v^2s_2 + 6s_3s_1^3 + 2s_1^3s_2 + s_1^4 \\
b_4 &= 6s_3^4 + 24s_3^3s_2 + 24s_3^2s_2^2 + 8s_3s_2^3 + s_2^4 + 24s_3^3s_1 + 60s_3^2s_1s_2 \\
&\quad + 36s_3s_1s_2^2 + 6s_1s_2^3 + 24s_3^2s_1^2 + 36s_3s_1^2s_2 + 12s_1^2s_2^2 + 8s_3s_1^3 \\
&\quad + 6s_1^3s_2 + s_1^4 \\
b_5 &= s_3^4 + 6s_3^3s_2 + 12s_3^2s_2^2 + 6s_3s_2^3 + s_2^4 + 2s_3^3s_1 + 6s_3^2s_1s_2 + 6s_3s_1s_2^2 + 2s_1s_2^3 \\
b_6 &= 2s_3s_1^3 + s_1^4 \\
b_7 &= s_3^4 + 6s_3^3s_2 + 12s_3^2s_2^2 + 6s_3s_2^3 + s_2^4 + 8s_3^3s_1 + 36s_3^2s_1s_2 + 36s_3s_1s_2^2 \\
&\quad + 8s_1s_2^3 + 24s_3^2s_1^2 + 60s_3s_1^2s_2 + 24s_1^2s_2^2 + 24s_3s_1^3 + 24s_1^3s_2 + 6s_1^4 \\
b_8 &= s_3^4 + 8s_3^3s_2 + 24s_3^2s_2^2 + 24s_3s_2^3 + 6s_2^4 + 6s_3^3s_1 + 36s_3^2s_1s_2 + 60s_3s_1s_2^2 + 24s_1s_2^3 \\
&\quad + 12s_3^2s_1^2 + 36s_3s_1^2s_2 + 24s_1^2s_2^2 + 6s_3s_1^3 + 8s_1^3s_2 + s_1^4 \\
b_9 &= 2s_3s_2^3 + s_2^4 \\
b_{10} &= 2s_1^3s_2 + s_1^4 \\
b_{11} &= 2s_3s_2^3 + s_2^4 + 6s_3s_1s_2^2 + 6s_1s_2^3 + 6s_3s_1^2s_2 + 12s_1^2s_2^2 + 2s_3s_1^3 + 6s_1^3s_2 + s_1^4 \\
b_{12} &= s_2^4 + 2s_1s_2^3,
\end{aligned}
$$

where $s_3 = 1 - s_1 - s_2$. Compared to linear barycentric interpolation, we observe that the function remains linear on the surface vertices, but now is quartic on the barycentric coordinates $\boldsymbol{s}$.

For the case an element has one irregular vertex, the number of control points the limit point relies on depends on the valence of the irregular vertex and is defined by $K = N + 6$, where N is the valence of the irregular vertex. The limit surface and its derivatives can then be evaluated numerically using the algorithm proposed by Stam in Evaluation of Loop Subdivision Surfaces.

If one body element has more than one irregular vertex, it has to be preprocessed by manually subdividing it once.

This scheme results in a C_2 continuous surface with the exceptions of irregular vertices where it is C_1.

Bibliography

[1] Sameer Agarwal, Keir Mierle, and Others. Ceres solver. `http://ceres-solver.org`.

[2] David Baraff and Andrew Witkin. Large steps in cloth simulation. *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 43–54, 1998.

[3] David Baraff, Andrew Witkin, and Michael Kass. Untangling cloth. *ACM Transactions on Graphics (TOG)*, 22(3):862–870, 2003.

[4] Jernej Barbič and Doug L. James. Real-Time subspace integration for St. Venant-Kirchhoff deformable models. *ACM Transactions on Graphics*, 24(3):982–990, July 2005.

[5] Adam W. Bargteil and Tamar Shinar. An introduction to physics-based animation. *ACM SIGGRAPH 2019 Courses*, pages 1–57, July 2019.

[6] Aric Bartle, Alla Sheffer, Vladimir G. Kim, Danny M. Kaufman, Nicholas Vining, and Floraine Berthouzoz. Physics-driven pattern adjustment for direct 3d garment editing. *ACM Trans. Graph.*, 35(4):50:1–50:11, jul 2016.

[7] Miklós Bergou, Max Wardetzky, Stephen Robinson, Basile Audoly, and Eitan Grinspun. Discrete elastic rods. *ACM Transactions on Graphics*, 27, 09 2008.

[8] Floraine Berthouzoz, Akash Garg, Danny M. Kaufman, Eitan Grinspun, and Maneesh Agrawala. Parsing sewing patterns into 3d garments. *ACM Trans. Graph.*, 32(4):85:1–85:12, July 2013.

[9] Javier Bonet and Richard D. Wood. *Nonlinear Continuum Mechanics for Finite Element Analysis*. Cambridge University Press, 2 edition, 2008.

[10] Robert Bridson, Ronald Fedkiw, and John Anderson. Robust treatment of collisions, contact and friction for cloth animation. *ACM Transactions on Graphics (ToG)*, 21(3):594–603, 2002.

[11] Remi Brouet, Alla Sheffer, Laurence Boissieux, and Marie-Paule Cani. Design preserving garment transfer. *ACM Trans. Graph.*, 31(4):36:1–36:11, July 2012.

[12] Daniel Burkhart, Bernd Hamann, and Georg Umlauf. Iso-geometric Finite Element Analysis Based on Catmull-Clark Subdivision Solids. *Computer Graphics Forum*, 2010.

[13] Xiaowu Chen, Bin Zhou, Fei-Xiang Lu, Lin Wang, Lang Bi, and Ping Tan. Garment modeling with a depth camera. *ACM Trans. Graph.*, 34(6):203–1, 2015.

[14] Gabriel Cirio, Jorge Lopez-Moreno, David Miraut, and Miguel A Otaduy. Yarn-level simulation of woven cloth. *ACM Transactions on Graphics (TOG)*, 33(6):207, 2014.

[15] Gabriel Cirio, Jorge Lopez-Moreno, and Miguel A Otaduy. Efficient simulation of knitted cloth using persistent contacts. *Proceedings of the 14th ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 55–61, 2015.

[16] Alexander Clegg, Jie Tan, Greg Turk, and C. Karen Liu. Animating human dressing. *ACM Trans. Graph.*, 34(4):116:1–116:9, July 2015.

[17] Philippe Decaudin, Dan Julius, Jamie Wither, Laurence Boissieux, Alla Sheffer, and Marie-Paule Cani. Virtual garments: A fully geometric approach for clothing design. *Computer Graphics Forum*, 25(3):625–634, 2006.

[18] Shen Dong, Peer-Timo Bremer, Michael Garland, Valerio Pascucci, and John Hart. Spectral surface quadrangulation. *ACM Trans. Graph.*, 25:1057–1066, 07 2006.

[19] Hanno EssÃĺn and Arne Nordmark. Static deformation of a heavy spring due to gravity and centrifugal force. *European Journal of Physics*, 31(3):603âĂŞ609, Apr 2010.

[20] Ye Fan, Joshua Litven, David IW Levin, and Dinesh K Pai. Eulerian-on-lagrangian simulation. *ACM Transactions on Graphics (TOG)*, 32(3):22, 2013.

[21] S.J. Farlow. *Partial Differential Equations for Scientists and Engineers*. Dover books on advanced mathematics. Dover Publications, 1993.

[22] Eitan Grinspun, Yotam Gingold, Jason Reisman, and Denis Zorin. Computing discrete shape operators on general meshes. *Computer Graphics Forum (Eurographics)*, 25(3):547–556, 2006.

[23] Peng Guan, Loretta Reiss, David A. Hirshberg, Alexander Weiss, and Michael J. Black. Drape: Dressing any person. *ACM Trans. Graph.*, 31(4):35:1–35:10, July 2012.

[24] David Harmon, Etienne Vouga, Rasmus Tamstorf, and Eitan Grinspun. Robust treatment of simultaneous collisions. *ACM Transactions on Graphics (TOG)*, 27(3):23, 2008.

[25] G.A. Holzapfel. *Nonlinear Solid Mechanics: A Continuum Approach for Engineering*. Wiley, 2000.

[26] Dan Julius, Vladislav Kraevoy, and Alla Sheffer. D-charts: Quasi-developable mesh segmentation. *Computer Graphics Forum*, 24(3):581–590, 2005.

[27] Tsz-Ho Kwok, Yan-Qiu Zhang, Charlie C. L. Wang, Yong-Jin Liu, and Kai Tang. Styling evolution for tight-fitting garments. *IEEE Transactions on Visualization and Computer Graphics*, 22(5):1580–1591, May 2016.

[28] David Levin, Joshua Litven, Garrett Jones, Shinjiro Sueda, and Dinesh Pai. Eulerian solid simulation with contact. *ACM Trans. Graph.*, 30:36, 07 2011.

[29] Bruno Lévy, Sylvain Petitjean, Nicolas Ray, and Jérome Maillot. Least squares conformal maps for automatic texture atlas generation. *ACM transactions on graphics (TOG)*, 21(3):362–371, 2002.

[30] Duo Li, Shinjiro Sueda, Debanga R Neog, and Dinesh K Pai. Thin skin elastodynamics. *ACM Transactions on Graphics (TOG)*, 32(4):49, 2013.

[31] Jie Li, Gilles Daviet, Rahul Narain, Florence Bertails-Descoubes, Matthew Overby, George E Brown, and Laurence Boissieux. An implicit frictional contact solver for adaptive cloth simulation. *ACM Transactions on Graphics (TOG)*, 37(4):52, 2018.

[32] Minchen Li, Danny M. Kaufman, Vladimir G. Kim, Justin Solomon, and Alla Sheffer. Optcuts: Joint optimization of surface cuts and parameterization. *ACM Trans. Graph.*, 37(6):247:1–247:13, December 2018.

[33] Roberta Massabò and Luigi Gambarotta. Wrinkling of Plane Isotropic Biological Membranes. *Journal of Applied Mechanics*, 74(3):550–559, May 2007.

[34] Yuwei Meng, Charlie C. L. Wang, and Xiaogang Jin. Flexible shape control for automatic resizing of apparel products. *Comput. Aided Des.*, 44(1):68–76, January 2012.

[35] Yuki Mori and Takeo Igarashi. Plushie: An interactive design system for plush toys. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 2007.

[36] Matthias Müller, Nuttapong Chentanez, Tae-Yong Kim, and Miles Macklin. Air meshes for robust collision handling. *ACM Transactions on Graphics (TOG)*, 34(4):133, 2015.

[37] Matthias Müller and Markus Gross. Interactive virtual materials. *Proceedings of Graphics Interface 2004*, pages 239–246, May 2004.

[38] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, New York, 2nd edition, 2006.

[39] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer, New York, NY, USA, second edition, 2006.

[40] Miguel A Otaduy, Rasmus Tamstorf, Denis Steinemann, and Markus Gross. Implicit contact handling for deformable objects. *Computer Graphics Forum*, 28(2):559–568, 2009.

[41] A Cengiz Öztireli, Gael Guennebaud, and Markus Gross. Feature preserving point set surfaces based on non-linear kernel regression. *Computer Graphics Forum*, 28(2):493–501, 2009.

[42] Allen C Pipkin. The relaxed energy density for isotropic elastic membranes. *IMA journal of applied mathematics*, 36(1):85–99, 1986.

[43] Gerard Pons-Moll, Javier Romero, Naureen Mahmood, and Michael J. Black. Dyna: a model of dynamic human shape in motion. *ACM Transactions on Graphics*, 34(4):120:1–120:14, July 2015.

[44] Roi Poranne, Marco Tarini, Sandro Huber, Daniele Panozzo, and Olga Sorkine-Hornung. Autocuts: Simultaneous distortion and cut optimization for uv mapping. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)*, 36(6), 2017.

[45] A.C. Rencher. *Methods of Multivariate Analysis*. Wiley Series in Probability and Statistics. Wiley, 2003.

[46] Cody Robson, Ron Maharik, Alla Sheffer, and Nathan Carr. Context-aware garment modeling from sketches. *Computers & Graphics*, 35(3):604–613, 2011.

[47] Damien Rohmer, Tiberiu Popa, Marie-Paule Cani, Stefanie Hahmann, and Alla Sheffer. Animation wrinkling: augmenting coarse cloth simulations with realistic-looking wrinkles. *ACM Transactions on Graphics*, 29(6):157:1–157:8, December 2010.

[48] Pedro V Sander, John Snyder, Steven J Gortler, and Hugues Hoppe. Texture mapping progressive meshes. *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 409–416, 2001.

[49] Nicholas Sharp and Keenan Crane. Variational surface cutting. *ACM Trans. Graph.*, 37(4), 2018.

[50] Alla Sheffer, Bruno Lévy, Maxim Mogilnitsky, and Alexander Bogomyakov. Abf++: Fast and robust angle based flattening. *ACM Trans. Graph.*, 24(2), April 2005.

[51] M. Skouras, B. Thomaszewski, B. Bickel, and M. Gross. Computational design of rubber balloons. *Comput. Graphics Forum (Proc. Eurographics)*, 31(2), 2012.

[52] Mélina Skouras, Bernhard Thomaszewski, Peter Kaufmann, Akash Garg, Bernd Bickel, Eitan Grinspun, and Markus Gross. Designing inflatable structures. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 33(4), 2014.

[53] Olga Sorkine, Daniel Cohen-Or, Rony Goldenthal, and Dani Lischinski. Bounded-distortion piecewise mesh parameterization. *Proceedings of the conference on Visualization*, pages 355–362, 2002.

[54] M. Spivak. *A Comprehensive Introduction to Differential Geometry*. Number v. 1 in A Comprehensive Introduction to Differential Geometry. Publish or Perish, Incorporated, 1999.

[55] Jos Stam. Evaluation of loop subdivision surfaces. *SIGGRAPH*, 1998.

[56] Shinjiro Sueda, Garrett L Jones, David IW Levin, and Dinesh K Pai. Large-scale dynamic simulation of highly constrained strands. *ACM Transactions on Graphics (TOG)*, 30(4):39, 2011.

[57] Joseph Teran, Eftychios Sifakis, Geoffrey Irving, and Ronald Fedkiw. Robust quasistatic finite elements and flesh simulation. *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 181–190, July 2005.

[58] Emmanuel Turquin, Jamie Wither, Laurence Boissieux, Marie-Paule Cani, and John F Hughes. A sketch-based interface for clothing virtual characters. *IEEE Computer graphics and applications*, 27(1):72–81, 2007.

[59] Pascal Volino and Nadia Magnenat-Thalmann. *Resolving surface collisions through intersection contour minimization*, volume 25. ACM, 2006.

[60] Pascal Volino, Nadia Magnenat-Thalmann, and Francois Faure. A simple approach to nonlinear tensile stiffness for accurate cloth simulation. *ACM Trans. Graph.*, 28(4), September 2009.

[61] Charlie C. L. Wang and Kai Tang. Pattern computation for compression garment. *Proceedings of the 2008 ACM symposium on Solid and physical modeling*, pages 203–211, 2008.

[62] Charlie C. L. Wang and Kai Tang. Pattern computation for compression garment by a physical/geometric approach. *Comput. Aided Des.*, 42(2):78–86, February 2010.

[63] Charlie C. L. Wang, Yu Wang, and Matthew M. F. Yuen. Design automation for customized apparel products. *Comput. Aided Des.*, 37(7):675–691, June 2005.

[64] Duo Wang, Xiangmin Jiao, Rebecca Conley, and James Glimm. On the curvature effect of thin membranes. *Journal of Computational Physics*, 233:449–463, 01 2013.

[65] Yu Wang, Alec Jacobson, Jernej Barbič, and Ladislav Kavan. Linear subspace design for real-time shape deformation. *ACM Trans. Graph.*, 34(4):57:1–57:11, July 2015.

[66] Nicholas J. Weidner, Kyle Piddington, David I.W. Levin, and Shinjiro Sueda. Eulerian-on-Lagrangian cloth simulation. *ACM Transactions on Graphics*, 37(4):50:1–50:11, August 2018.

[67] Hitoshi Yamauchi, Stefan Gumhold, Rhaleb Zayer, and Hans-Peter Seidel. Mesh segmentation driven by gaussian curvature. *The Visual Computer*, 21(8-10):659–668, 2005.

[68] Jonas Zehnder, Stelian Coros, and Bernhard Thomaszewski. Designing structurally-sound ornamental curve networks. *ACM Trans. Graph.*, 35(4), 2016.

[69] Kun Zhou, John Synder, Baining Guo, and Heung-Yeung Shum. Iso-charts: stretch-driven mesh parameterization using spectral analysis. *Proceedings of the Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pages 45–54, 2004.

[70] O.C. Zienkiewicz, R. Taylor, and D. Fox. *The Finite Element Method for Solid and Structural Mechanics*. 01 2005.