Rationalization in architecture with surfaces foliated by elastic curves

Nørbjerg, Toke Bjerge; Gravesen, Jens; Brander, David

Rationalization in architecture with surfaces foliated by elastic curves

Toke Bjerge Nørbjerg

1 July 2016

PHD-2016-419

DTU Compute
Department of Applied Mathematics and Computer Science

Technical University of Denmark
Department of Applied Mathematics and Computer Science
Richard Petersens Plads, building 324,
2800 Kongens Lyngby, Denmark
Phone +45 4525 3031
compute@compute.dtu.dk
www.compute.dtu.dk

Abstract

We develop methods for rationalization of CAD surfaces using elastic curves, aiming at a cost-effective fabrication method for architectural designs of complex shapes. By moving a heated flexible metal rod though a block of expanded polystyrene, it is possible to produce shapes with both positive and negative Gaussian curvature, either for direct use or for use as moulds for concrete casting. If we can control the shape of the rod while it is moving, we can produce prescribed shapes.

The flexible rod assumes at all times the shape of an Euler elastica (or elastic curve). The elastica are given in closed analytic form using elliptic functions. We use a gradient-driven optimization to approximate arbitrary planar curves by planar elastic curves. The method depends on an explicit parameterization of the space of elastic curves and on a method for finding a good initial guess for the optimization.

We approximate CAD surfaces by first extracting a collection of planar surface curves and approximating these by elastica. Providing the data for these curves to robots holding the flexible rod, we can produce an elastica-foliated surface that approximates the given CAD surface. Since not all surfaces can be closely approximated by an elastica-foliated surface, an arbitrary CAD surface must first be subdivided into segments that can be approximated. We discuss strategies for subdividing an arbitrary surface into segments that can be closely approximated, taking into account the aesthetics of the segmentation and the production constraints. If the given surface is smooth, we want the approximating surface to be smooth as well, so we must ensure smooth transition between the surface segments of the final result.

As an alternative to rationalization of arbitrary designs, we also present a method for direct generation of design surfaces using foliated Euler elastica. Here we work from a grid of blocks, so the segmentation is given, but we must still ensure smooth transition between segments.

Resumé (Abstract in Danish)

I afhandlingen udvikles metoder til rationalisering af CAD-flader ved brug af elastiske kurver med det formål at kunne fremstille komplekse arkitektoniske former indenfor en relativt lav budgetramme. Ved at føre en opvarmet fleksibel metalstang gennem en blok ekspanderet polystyren kan man skabe flader med både positiv og negativ Gausskrumning. Hvis man løbende kan kontrollere metalstangens form under bevægelsen, kan man fremstille foreskrevne former. De fremkomne polystyrenelementer kan enten anvendes, som de er, eller bruges som støbeforme til beton.

De former, den fleksible metalstang kan antage, kaldes Euler elastica (eller elastiske kurver) og kan udtrykkes analytisk ved brug af elliptiske funktioner. Vi benytter gradientbaseret optimering til at tilnærme vilkårlige plane kurver med plane elastiske kurver. Vores metode bygger på en eksplicit parametrisering af rummet af elastiske kurver og på en metode til at finde et godt første gæt til optimeringen.

Vi tilnærmer en given CAD-flade ved først at udvælge et sæt plane kurver på fladen, som vi så kan tilnærme med elastiske kurver. Ved at give data for disse kurver til et par robotter, som holder den fleksible stang, kan man fremstille en tilnærmelse af den givne CAD-flade, som er frembragt af elastiske kurver. Det er ikke alle CAD-flader som kan tilnærmes godt med en sådan flade, så en vilkårlig CAD-flade skal først opdeles i dertil velegnede stykker. Vi diskuterer forskellige segmenteringsstrategier, hvor vi tager højde for, hvor godt stykkerne kan tilnærmes, segmenteringens æstetiske kvalitet samt begrænsninger tilknyttet produktionsmetoden. Hvis den givne CAD-flade er glat, ønsker vi også at tilnærmelsen skal være glat, så vi må sikre glat overgang mellem segmenterne i den nye flade.

Som alternativ til rationalisering af et vilkårligt design, præsenterer vi en fremgangsmåde til direkte design af flader frembragt af elliptiske kurver. Her arbejdes der ud fra en given segmentering i blokke, men man skal stadig sikre glat overgang mellem segmenterne.

Preface

The work behind this thesis was conducted as part of my PhD studies at the Department of Applied Mathematics and Computer Science at the Technical University of Denmark from June 2013 to June 2016. The work was supervised by Associate Professor Jens Gravesen (main supervisor) and Associate Professor David Brander (co-supervisor). The PhD project is part of the industrial research project *BladeRunner* started in 2012 with funding from the Danish National Advanced Technology Foundation (now Innovation Fund Denmark).

During my studies I attended three conferences: SIAM Conference on Geometric and Physical Modeling (Denver, November 2013), Advances in Architectural Geometry (London, September 2014) and Robotic Fabrication in Architecture, Art and Design (Sydney, March 2016). At the first two, my work was presented at the poster session. At the latter, I co-organized a workshop with the other BladeRunner partners showcasing our technology.

From September 2014 to February 2015 I visited the Institute of Geometry at Graz University of Technology in Graz, Austria. My host for this research stay was Professor Johannes Wallner. During the semester in Graz I also attended a course at the University of Graz.

As part of my PhD studies I have coauthored five papers:

[BBE+15] D. Brander, A. Bærentzen, A. Evgrafov, J. Gravesen, S. Markvorsen, T. Nørbjerg, P. Nørtoft, and K. Steenstrup. Hot blade cuttings for the building industry. To appear. Preprint available on ResearchGate, 2015.

[BGN16] D. Brander, J. Gravesen, and T. B. Nørbjerg. Approximation by planar elastic curves. *Advances in Computational Mathematics*, 2016. To appear. DOI: 10.1007/s10444-016-9474-z.

[SFN+16] A. Søndergaard, J. Feringa, T. Nørbjerg, K. Steenstrup, D. Brander, J. Gravesen, S. Markvorsen, A. Bærentzen, K. Petkov, J. Hattel, K. Clausen, K. Jensen, L. Knudsen, and J. Kortbek., Robotic hot-blade cutting: An industrial approach to cost-effective production of double curved concrete structures. In *Robotic Fabrication in Architecture, Art and Design 2016*, pages 150–164. Springer, 2016.

[BBC+16] D. Brander, J. A. Bærentzen, K. Clausen, A. Fisker, J. Gravesen, M. N. Lund, T. B. Nørbjerg, K. Steenstrup, and A. Søndergaard. Designing for robotic hot-blade cutting. In *Advances in Architectural Geometry 2016*, 2016. To appear.

[SNS+16] K. H. Steenstrup, T. B. Nørbjerg, A. Søndergaard, J. A. Bærentzen, and J. Gravesen., Cuttable ruled surface strips for milling. In *Advances in Architectural Geometry 2016*, 2016. To appear.

Most of [BGN16] is included in Chapters 2 and 3 of this thesis and these chapters also cover the contents of [BBE+15]. My contributions to [BBC+16] and [SFN+16] are covered in Chapters 4 and 5, respectively. The contents of [SNS+16] is outside the scope of this thesis.

Chapter-by-chapter overview

In Chapter 1 I introduce the industrial project, BladeRunner, which my PhD is a part of, and I detail the desired fabrication process and the problems that need to be solved in order to achieve this. I then give a brief overview of the development of the theory of Euler elastica and spline curves. Finally, I introduce the software that has been used for implementation.

Chapter 2 concerns the mathematical formulation of Euler elastica. I first derive the Euler-Langrange equation for the elastica, and then I introduce elliptic functions and elliptic integrals and describe some of their properties. Using these I solve the Euler-Langrange equation and get an explicit parameterization of the space of elastic curve segments. Finally, I describe a way to recover the parameters of an elastic curve segment.

An important step towards the overall goal, is a way to approximate an arbitrary curve by a G^1 piecewise elastic curve. This is the focus Chapter 3 in which we find the elastica that has the smallest L^2-distance to a given curve. I also go through some considerations on the physical applicability and how these can be dealt with by introducing optimization constraints.

Chapter 4 deals with the problem of finding an elastic curve segment with prescribed endpoints, end tangent angles and length. This is done by first finding a discrete version of the desired curve and then recovering its control parameters. The algorithm for finding the discrete elastica was based on an implementation made by J. Andreas Bærentzen. I then describe how this method can be used for designing elastica-foliated surfaces.

The next three chapters concern surface rationalization. In Chapter 5 I describe our basic method for approximating a surface by an elastica-foliated surface. I also describe how we can evaluate the result.

An arbitrary surface can usually not be well approximated by a single elastica-foliated surface. We therefore need an algorithm for subdividing the surface into patches that can be approximated well. In Chapter 6 I discuss several segmenting strategies, which were developed in collaboration with Kasper Steenstrup.

Finally, in Chapter 7 I describe how to approximate each piece of a segmented surface by an elastica-foliated patch, such that the patches meet in a tangent continuous way. I describe two strategies for doing this. The first is based on the methods developed in Chapter 3, while the second introduces several new constraints to the optimization problem that needs to be solved.

Chapter 8 deals with a problem that came up in relation to the practical application of the methods described in this thesis. If we know the end points and tangent angles of an elastic curve segment, and we measure the curve height, can we determine the curve (without solving a complicated optimization problem)? It turns out that for a symmetric curve segment, this can be done by a bisection.

Acknowledgements

I gratefully acknowledge the financial support given by Innovation Fund Denmark to the Blade-Runner project and thus to my scholarship at DTU. For supporting my five month research stay in Graz, Austria, I am grateful to Otto Mønsted's Fund.

I would like to thank my colleagues from the Section for Mathematics at DTU Compute. The transition from student to researcher was made easy by the warm welcome I received, when I arrived at DTU. I felt that everyone took a real interest in my wellbeing as a PhD student and gave their individual take on the challenges that is part of being a researcher. More generally, I would like to thank the researchers and PhD students of Building 303B. I have enjoyed the informal conversations over lunch, the random meetings in the kitchen and the weekly beer club.

A very special thanks goes to my supervisors Jens Gravesen and David Brander. Jens' great enthusiasm is admirable, his knowledge of various mathematical fields impressive, and his way of quickly suggesting (good) ways of solving unforeseen problems is inspiring. David generously shares his experience as a researcher, which has taught me a lot about the academic world. His constructive criticism has improved the way I present my work and influenced the way I think about presenting it. Both Jens and David have given me excellent guidance and encouragement throughout my studies. They have been confident in me, which has kept me self-confident. While giving me the necessary pushes for moving forward, they have not pressured me. Thus I can gratefully say that most of the pressure on my shoulders have been put there by my own ambition, which was in turn inspired by the example set by Jens and David.

I would like to thank my fellow BladeRunner-PhD, Kasper Steenstrup, who embarked on this adventure with me back in 2013. Coming from different backgrounds, I believe we have both benefitted greatly from sharing an office and throwing ideas back and forth between us. Thanks also to Andreas Bærentzen. I have enjoyed and benefitted from our talks (most of which took place outside Denmark, I believe).

During my stay in Graz, I had the daily pleasure of a lunch break with Johannes Wallner and his PhD students, past and present. I really enjoyed these and our talks on everything from fantasy fiction to German and Danish grammar as well as Johannes' countless anecdotes about other mathematicians. I would like to thank the PhD students and researchers at the Institute of Geometry, who were all very nice and helpful, when I arrived at TU Graz. In particular I want to thank Johannes for showing me the city, for making me feel like part of the institute and for taking me rock climbing; an experience I had not expected would be part of my PhD studies!

I am also grateful to everyone that has been involved in the BladerRunner project: the partners from Odico, GXN, DTI, DTU Mek and CONFAC. I have learned a lot from working with all these people and it has bean a real pleasure!

Further, I want to thank Professor Jan Philip Solovej, who supervised my Master's thesis, for giving me the confidence to apply for PhD positions and for still, three years later, taking an interest in my career as a researcher.

Last but not least, I would like to thank Johanne Harsted-Simonsen for helping me compile the bibliography in BibTeX and for supporting me at home during my exciting, but sometimes also stressful time as a PhD student.

Toke Bjerge Nørbjerg
June 2016

Contents

Chapter 1

Introduction

One of the major challenges in contemporary architecture is the conflict between the architects' design ambitions and the budgetary constraints on the complexity of building elements. While modern CAD software makes the creation of advanced 3D models with double curved surfaces easy and available to everyone, the fabrication of such elements is often prohibitively expensive. However, an increasing number of projects make use of digital design tools and seek to move outside the architectural paradigm of flat-surface, right-angled buildings. Hence there is a great need for the development of cost-effective fabrication methods for building elements of complex geometry.

For large-scale concrete structures, the classic way of producing complex shapes involves manual fabrication of formwork, typically out of wood. More recently, large-scale CNC milling have been employed either for production of foam moulds or direct milling of construction materials. None of these processes, however, provide a cost-effective method for general construction.

1.1 The BladeRunner project

Background The Danish company Odico specializes in robotic fabrication of expanded polystyrene (EPS) formwork for concrete casting of complex building elements. The primary fabrication method is hot wire cutting, done by a robot moving a heated metal wire through the EPS, thus cutting a prescribed shape, see Figure 1.1. We say "cutting" though in reality the EPS melts and evaporates. The cut shape can be coated with a polymer or other coating material to make the surface more smooth and hard, so it can be used as a mould for concrete casting. For the production of moulds, a negative form must be cut, but positive shape production in EPS is also relevant for some applications.

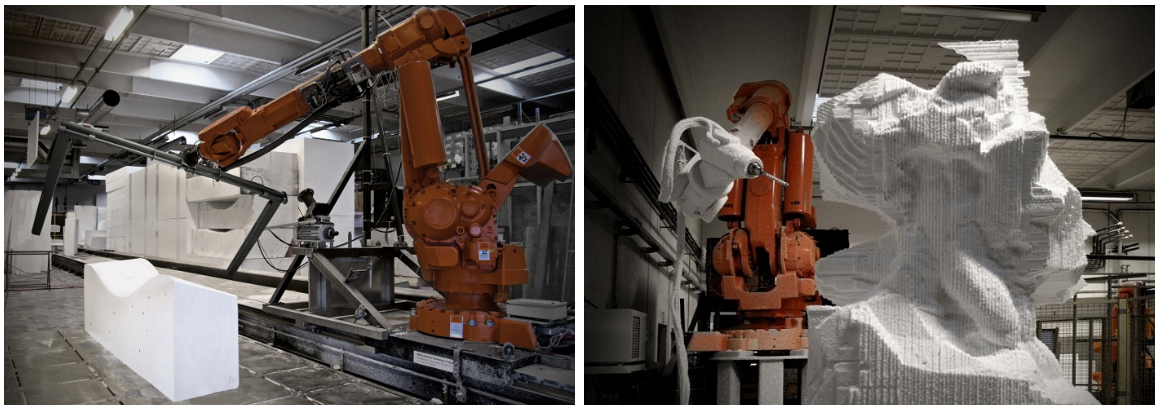


Figure 1.1: Robotic hot wire cutting and robot milling at Odico.

Hot wire cutting is a very fast and precise procedure. The main limitation of this fabrication method is that it can only produce ruled surfaces. A ruled surface has non-positive Gaussian curvature at all points, which means that no surface with positive Gaussian curvature can be produced by hot wire cutting, see Figure 1.2. For positively curved shapes and for elements of fine detail, which cannot be cut by wire, companies like Odico use CNC milling. Milling is a very precise way of producing formwork in complex shapes, but it is a very time consuming, and thus expensive, fabrication method. Moreover, a lot of work is needed in choosing the tool path for the cutter.

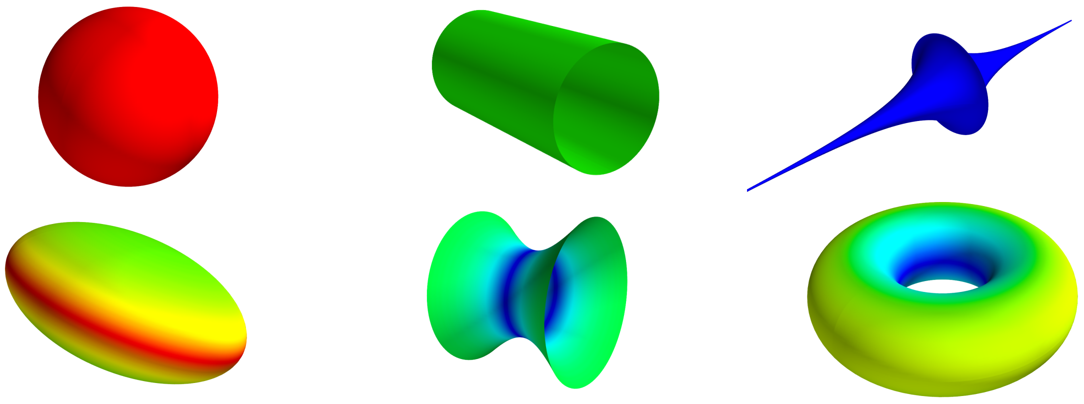


Figure 1.2: Surfaces shown with Gaussian curvatures between -1 (blue) and 1 (red). The sphere, cylinder and tractricoid have constant Gaussian curvatures of 1, 0 and -1, respectively. The Gaussian curvature of the ellipsoid is everywhere positive, while that of the hyperboloid is everywhere negative. The torus has areas of both positive and negative Gaussian curvature and two circles with zero Gaussian curvature. Note that the cylinder and hyperboloid are actually ruled surfaces.

BladeRunner The BladeRunner project is an industrial research project supported by Innovation Fund Denmark and involving several Danish companies and research institutions. The aim of BladeRunner is development of a new production method for EPS formwork. Instead of using a wire, the cutting will be done by a flexible cutting tool, the so called hot blade. The blade can be bent into different shapes during cutting, thus extending the class of shapes that can be produced. In particular it will enable fast fabrication of positively curved elements. To allow architects as much freedom as possible in the design phase, part of the project concerns approximating an arbitrary design by one that may be cut by the hot blade (the main focus of this thesis). Another aim of the BladeRunner project is approximating arbitrary designs by piecewise ruled surfaces, in order to use wire cutting as a rough first step before milling the finer details of the formwork, see [SNS$^+$16].

In developing these methods Odico have allied with the Department of Applied Mathematics and Computer Science (DTU Compute) at the Technical University of Denmark (DTU), developing algorithms for surface rationalization; Danish Technological Institute (DTI), developing the flexible cutting tool and robot setup, and testing; Department of Mechanical Engineering (DTU Mechanical Engineering) at DTU, developing thermal and thermomechanical models for hot wire and hot blade cutting, predicting kerf width, and comparing the hot blade shapes predicted by mathematical and mechanical models with those measured for physical blades; 3XN architects, in particular their innovation unit GXN, providing field knowledge of the architectural world, ideas for commercialization, and producing test cases for the developed algorithms; and finally the concrete factory CONFAC delivering and casting concrete, and evaluating results of casting in different mould types (i.e. different EPS densities and coatings).

Hot blade cutting The process of robotic hot blade cutting is similar to wire cutting, except that instead of a wire held in a frame, we have a flexible cutting tool held by two robots that move in synchronization. For the experimental phase, a third robot is used for holding and moving the block of EPS that will be cut, see Figure 1.3. The more permanent setup may instead have the EPS on a conveyor belt that passes between the robots holding the cutting tool.

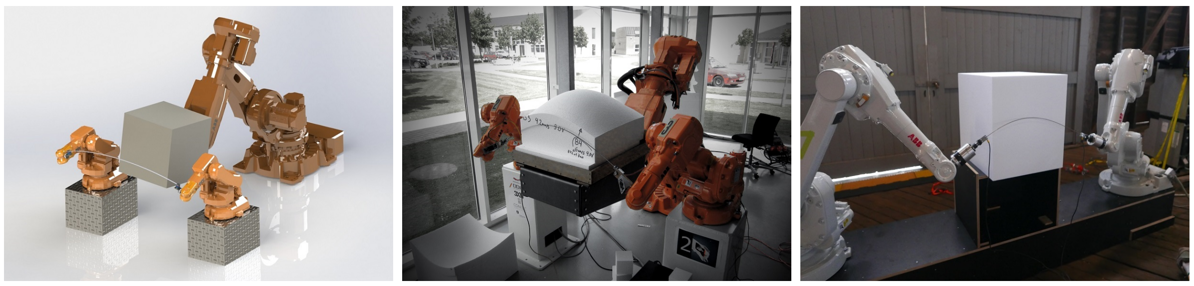


Figure 1.3: Left: A rendering of the idea for the robot setup. Center: The actual experimental setup at DTI in Odense, Denmark. Right: Robot setup for a workshop organized by the BladeRunner team for RobArch 2016 in Sydney, Australia. Here the EPS block is fixed, so the two robots holding the cutting tool do all the work.

During the course of the BladeRunner project, DTI and DTU Mechanical Engineering have experimented with two versions of the cutting tool. The first is a flat, nickel-chromium blade with a rectangular cross section of 5 mm$\times$1 mm, the second a stainless steel tube with outer/inner diameter 3 mm/2.5 mm.[1] In this thesis, we shall refer to both of these cutting tools as *blades*, and to the process as *hot blade cutting*. When we need to distinguish the two types, we will talk about the *flat blade* and the *tube*. The blades were between 0.8 m and 1.5 m in length.

The main advantage of the flat blade is that, since it has a specific thin dimension, it will always bend in the desired direction, while the tube is equally prone to bend in all directions. As mentioned, the cutting tool is supposed to melt its way through the EPS, so that it does not experience a force from the block. In reality it is difficult to maintain the correct cutting speed to ensure that no such force is applied to the cutting tool, since the optimal speed will depend on the temperature of the cutting tool. Using a very low speed is not a solution, since this will cause too much EPS melting, resulting in a bad surface quality. If the speed is too high, so the cutting tool experiences a force, the tube will be prone to bend in the direction opposite to its velocity, while the flat blade is less so.

The disadvantage of the flat blade is that its cross section is asymmetric, which means that it is supposed to cut in a specific direction. When the blade is changing shape while moving through the EPS, the different points in the blade will have different velocities, so there will be parts of the flat blade that are not moving in the optimal direction. In contrast, the tube has a symmetric cross section, so it cuts equally well in all directions. However, for both blade types one should consider the magnitude of the velocity at different points on the blade. Since cutting speed affects surface quality, it is important that it does not vary too much along the blade.

The blade shape is given by the positions of its end points and the orientations of its end tangents (and the length of the cutting tool, which is fixed). To control the cutting motion, we simply provide the data for movement of the robots' end effectors. These are given as a discrete set of data points, and the robot will automatically interpolate the data. We do not know how the robots interpolate the data, which means that we only have complete control over the blade at a discrete set of points.

[1]Experiments were also done with flat blades and tubes of other dimensions.

Surface rationalization The challenge addressed in this thesis is that of surface rationalization, i.e. going from an arbitrary surface design to one that can be produced by hot blade cutting. The possible shapes of the flexible cutting tool are the so-called Euler *elastica* or *elastic curves*. Our approach is to first describe a method for approximating a planar curve by an elastica. We can then use this method on a selection of planar curves on a surface, to get a parameterized surface σ, which resembles the original design and is foliated by elastica, i.e. it has the form

$$\boldsymbol{\sigma}\colon [a,b] \times [c,d] \to \mathbb{R}^3, \quad \boldsymbol{\sigma}(u,v) = \boldsymbol{\gamma}_u(v),$$

where for any $u \in [a,b]$, $\boldsymbol{\gamma}_u\colon [c,d] \to \mathbb{R}^3$ is a planar elastic curve.

Typically we cannot approximate an arbitrary surface well in this manner. Hence, we will describe methods for segmenting a surface into patches that can be well approximated. If the original surface is smooth we also want the rationalized design to be smooth (or at least G^1). Therefore we cannot just approximate the segments independently, but must find a way to make sure the elastica-foliated patches meet with (at least) tangent continuity.

To sum up, by *surface rationalization* we mean segmentation and approximation by elastica-foliated surfaces which meet with tangent continuity.

1.2 Short historical overview

We here give a short overview of the development of the theories of elastica and splines. We then outline the recent work regarding hot blade cutting and similar fabrication processes. For more details on the history of the elastica, see Levien [Lev09] and Truesdell [Tru83].

In mathematics the elastica has been investigated as early as 1691 when Jacob Bernoulli posed the following problem: suppose that an elastic rod is fixed vertically at one end and that a load is hung on the other, sufficient to bend it into a horizontal position. What shape does the rod assume? Bernoulli solved this problem in the sense that he derived the differential equation for the curve, which is known as the *rectangular elastica*. In the process he discovered the formula for the radius of curvature for a planar curve.

In 1738 Daniel Bernoulli (Jacob's nephew) realized that all elastica should be minimizers of the bending energy, which is proportional to the integral of the square of the curvature $\int \kappa(s)^2 \,\mathrm{d}s$. In 1742 he suggested to Leonhard Euler to solve this minimization problem for a curve of given length with fixed end points and fixed tangents at these points. Euler did this, thus finding all possible shapes of the elastica. In 1744, he included this as a chapter in his treatise on the calculus of variations [Eul44]. (Oldfather [OEB33] provides a translation into English.)

The elastica can be expressed in closed form using the elliptic functions introduced by Karl Gustav Jakob Jacobi in 1829 [Jac29]. The closed form solutions were first derived in 1880 by Saalschütz [Saa13] and were expressed in modern notation by Love in 1892 [Lov20].

Elastic curves have been used in design for a long time. Prior to the use of computers in design, the word "spline" referred to thin wooden rods, fixed at various points by so-called "ducks" to create the naturally smooth curves needed in e.g. the shipping and aviation industries. Between the ducks, the rod assumes the shape of a planar elastica. Nowadays, "spline" usually refers to piecewise polynomial or rational curves. The computation of polynomial splines and the idea of using them in design has been discussed by many authors, we mention Birkhoff and de Boor [BdB65], Mehlum [Meh74] and Malcolm [Mal77] to name a few.

The use of hot wire cutting in architecture has been discussed e.g. in 2013 by McGee et al.

[MFS13]. A rationalization process for ruled surfaces has been developed by Flöry and Pottmann in 2010 [FP10].

In the late 1990s and early 2000s a group at TU Delft has done some work on developing a flexible cutting tool, i.e. a hot blade, for rapid prototyping (de Smit et al. [dSKBH02], [dSMB$^+$00] and related papers). This group uses numerical methods for computing the blade shapes. Alternatives to the hot blade cutting process include the use of a flexible membrane as a casting surface (Raun et al. 2012, [RKK12]) and spatial wire cutting (Rust et al. 2016, [RJGK16]).

1.3 Software choices

The algorithms described in this thesis have for the most part been implemented in Python 2.7 using the software library IPOPT [WB05], by Andreas Wächter and Carl Laird, for optimization. It was known when the BladeRunner project began, that the algorithms would rely on a tool for nonlinear constrained optimization. IPOPT (short for **I**nterior **P**oint **Opt**imizer) was chosen because several people at DTU Compute (who were at the time involved with the BladeRunner project) were familiar with it and because it is released under an open license that allows for the developed algorithms to be used commercially. IPOPT is written in C, but it can be interfaced from other languages, e.g. MATLAB and Python. Python was chosen because it is a rather easy-to-learn high-level language[2] and because Odico was already using python-based software in the robot motion planning for hot wire cutting.

For the algorithms related to surfaces some work has been done using Rhino (Rhinoceros 3D) developed by Robert McNeel & Associates. Rhino is a NURBS-based CAD application often used by architects for design. The test surfaces created for this project by the architects from GXN and Odico were drawn in Rhino, and since Rhino has a lot of convenient built-in functions for manipulating curves and surfaces in 3D, it made sense to take advantage of this. Rhino also has a scripting tool, RhinoScript, that allows the user to write Rhino commands in a script and create algorithms for complicated tasks. Furthermore, a Python interface for Rhino scripting is integrated in the application, so the functionality of Rhino can be called using Python syntax. We have used RhinoPython for the algorithms concerned with surface design (Section 4.3) and segmentation (Chapter 6), and then exported surface and curve data before moving on to approximation.

[2]At the beginning of this project, the author of this thesis had very little programming experience.

Chapter 2

Mathematical description of Euler's elastica

In this chapter, we derive the Euler-Lagrange equation for the elastica using calculus of variations. We then introduce the Jacobi elliptic functions and the elliptic integrals of the first and second kind and state some of their properties. With these in our toolbox, we can solve the Euler-Lagrange equation and find an explicit parameterization of the elastica. We then choose a specific way to parameterize the space of elastic curve segments, such that any segment is described by seven control parameters. We then introduce an algorithm for numerically calculating the control parameters for an elastic curve segment.

Sections 2.1, 2.2 and 2.3 are more detailed versions of respectively Section 2.1, Appendix A and Section 2.2 of [BGN16], while Section 2.4 is identical to Section 3 of that paper except for small alterations to make notation consistent throughout this thesis.

2.1 The Euler-Langrange equation

Let $\boldsymbol{\gamma}\colon [0,\ell] \to \mathbb{R}^2$ be a plane curve segment[1] which is parameterized by arc length. Let $\theta(s)$ denote the tangent angle, i.e. the angle between $\dot{\boldsymbol{\gamma}}(s)$ and the x-axis. We have $\dot{\boldsymbol{\gamma}}(s) = (\cos\theta(s), \sin\theta(s))$, so if the curve starts at (x_0, y_0) and ends at (x_ℓ, y_ℓ), say, it satisfies

$$x_\ell = x_0 + \int_0^\ell \cos\theta \, \mathrm{d}s, \quad y_\ell = y_0 + \int_0^\ell \sin\theta \, \mathrm{d}s. \tag{2.1}$$

As Daniel Bernoulli discovered, an *elastica* (or *elastic curve*) is a curve that, among curves with the same endpoints and end tangents, minimizes the *bending energy*

$$\tfrac{1}{2} \int_0^\ell \kappa(s)^2 \, \mathrm{d}s, \tag{2.2}$$

where $\kappa = \dot{\theta}$ is the curvature[2]. We will determine the Euler-Lagrange equation for such minimizers through a variational argument.

Suppose $\boldsymbol{\gamma}$ is an elastica from (x_0, y_0) to (x_ℓ, y_ℓ) with angle function θ. Let $\boldsymbol{\gamma}_t$ be the curve with angle function $\theta_t(s) = \theta(s) + t\varphi(s)$ and the same endpoints as $\boldsymbol{\gamma}$, where φ is a differentiable function with $\varphi(0) = \varphi(\ell) = 0$. Then the endpoints, length and end tangent angles of $\boldsymbol{\gamma}_t$ do not depend on t, and we have $\boldsymbol{\gamma} = \boldsymbol{\gamma}_0$.

[1]We shall often omit the word "segment" and simply say "curve".

[2]When we consider planar curves, κ will always denote the signed curvature.

Since γ minimizes (2.2), it must be a stationary curve for the bending energy

$$\mathscr{E}_{\text{bend}}(\gamma) = \tfrac{1}{2}\int_0^\ell \left(\frac{\mathrm{d}\theta}{\mathrm{d}s}\right)^2 \mathrm{d}s + \lambda_1\left(x_0 + \int_0^\ell \cos\theta\,\mathrm{d}s - x_\ell\right) + \lambda_2\left(y_0 + \int_0^\ell \sin\theta\,\mathrm{d}s - y_\ell\right).$$

We differentiate the energy of γ_t with respect to t

$$\begin{aligned}\frac{\mathrm{d}\mathscr{E}_{\text{bend}}(\gamma_t)}{\mathrm{d}t} &= \tfrac{1}{2}\int_0^\ell \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\mathrm{d}(\theta+t\varphi)}{\mathrm{d}s}\right)^2 \mathrm{d}s + \lambda_1\int_0^\ell \frac{\mathrm{d}}{\mathrm{d}t}\cos(\theta+t\varphi)\,\mathrm{d}s + \lambda_2\int_0^\ell \frac{\mathrm{d}}{\mathrm{d}t}\sin(\theta+t\varphi)\,\mathrm{d}s\\ &= \int_0^\ell \frac{\mathrm{d}(\theta+t\varphi)}{\mathrm{d}s}\frac{\mathrm{d}\varphi}{\mathrm{d}s}\,\mathrm{d}s - \lambda_1\int_0^\ell \varphi\sin(\theta+t\varphi)\,\mathrm{d}s + \lambda_2\int_0^\ell \varphi\cos(\theta+t\varphi)\,\mathrm{d}s,\end{aligned}$$

so for $t=0$ we get

$$\begin{aligned}\frac{\mathrm{d}\mathscr{E}_{\text{bend}}(\gamma_t)}{\mathrm{d}t}\bigg|_{t=0} &= \int_0^\ell \frac{\mathrm{d}\theta}{\mathrm{d}s}\frac{\mathrm{d}\varphi}{\mathrm{d}s}\,\mathrm{d}s - \lambda_1\int_0^\ell \varphi\sin\theta ds + \lambda_2\int_0^\ell \varphi\cos\theta\,\mathrm{d}s\\ &= \left[\varphi\frac{\mathrm{d}\theta}{\mathrm{d}s}\right]_0^\ell - \int_0^\ell \varphi\frac{\mathrm{d}^2\theta}{\mathrm{d}s^2}\,\mathrm{d}s - \lambda_1\int_0^\ell \varphi\sin\theta\,\mathrm{d}s + \lambda_2\int_0^\ell \varphi\cos\theta\,\mathrm{d}s \qquad (2.3)\\ &= -\int_0^\ell \varphi\left(\frac{\mathrm{d}^2\theta}{\mathrm{d}s^2} + \lambda_1\sin\theta - \lambda_2\cos\theta\right)\mathrm{d}s,\end{aligned}$$

where we have used that $\varphi(0) = \varphi(\ell) = 0$.

Since $t = 0$ is a stationary point for $t \mapsto \mathscr{E}_{\text{bend}}(\gamma_t)$ and since φ was arbitrary, we conclude that the angle function θ satisfies

$$\frac{\mathrm{d}^2\theta}{\mathrm{d}s^2} + \lambda_1\sin\theta - \lambda_2\cos\theta = 0, \qquad (2.4)$$

which is the Euler-Lagrange equation for the elastica.

Remark. The condition that $\varphi(0) = \varphi(\ell) = 0$ is what fixes the end tangents and ensures that $\varphi(\ell)\dot\theta(\ell) - \varphi(0)\dot\theta(0) = 0$ for all φ. If we allow the tangents to change, thus minimizing the energy for a curve with only fixed length and end points, we obtain the boundary conditions $\dot\theta(0) = \dot\theta(\ell) = 0$. In other words, the curve of minimal energy will satisfy (2.4) and have inflection points at its ends.

In order to find a simpler version of the Euler-Lagrange equation, we consider the role of the Lagrange multipliers λ_1 and λ_2. Setting $(\lambda_1, \lambda_2) = \lambda(\cos\phi, \sin\phi)$, with $\lambda \geq 0$, the previous equation becomes

$$\frac{\mathrm{d}^2\theta}{\mathrm{d}s^2} + \lambda\sin(\theta-\phi) = 0. \qquad (2.5)$$

We note that if $\lambda = 0$, the differential equation becomes $\ddot\theta(s) = 0$, so $\kappa = \dot\theta$ is constant, which means that γ is part of either a circle or a straight line.

Proposition 2.1. *Let $\gamma\colon [0,\ell] \to \mathbb{R}^2$ be an elastica, and let $\mu > 0$, $\alpha \in \,]-\pi,\pi]$. The scaled and rotated curve $\tilde\gamma\colon [0,\ell/\mu] \to \mathbb{R}^2$ defined by*

$$\tilde\gamma(s) = \tfrac{1}{\mu}R_\alpha\gamma(\mu s), \quad R_\alpha = \begin{pmatrix}\cos\alpha & -\sin\alpha\\ \sin\alpha & \cos\alpha\end{pmatrix},$$

is also an elastica.

Proof. It is easy to see that $\tilde\gamma$ is a rotated and scaled version of γ, since R_α is a rotation matrix and μs runs through $[0,\ell]$ as s runs through $[0,\ell/\mu]$. We find that $\tilde\gamma$ is parameterized by arc length, since for any $s \in [0, \ell/\mu]$ we have

$$L(\tilde{\gamma}, s) = \int_0^s \|\dot{\tilde{\gamma}}(u)\| \,\mathrm{d}u = \int_0^s \|\dot{\gamma}(\mu u)\| \,\mathrm{d}u = \tfrac{1}{\mu} \int_0^{\mu s} \|\dot{\gamma}(t)\| \,\mathrm{d}t = \tfrac{1}{\mu} L(\gamma, \mu s) = s.$$

The angle function θ for γ satisfies (2.5) for some λ, ϕ. The angle function for $\tilde{\gamma}$ is

$$\tilde{\theta}(s) = \theta(\mu s) + \alpha,$$

and thus

$$\ddot{\tilde{\theta}}(s) = \mu^2 \ddot{\theta}(\mu s) = -\mu^2 \lambda \sin\left(\theta(\mu s) - \phi\right) = -\tilde{\lambda} \sin\left(\tilde{\theta}(s) - \tilde{\phi}\right),$$

which shows that $\tilde{\theta}$ satisfies (2.5) with $\tilde{\lambda} = \mu^2 \lambda$ and $\tilde{\phi} = \phi + \alpha$. □

The above proposition shows that λ and ϕ in (2.5) may be considered as scaling and rotation parameters, and we can thus find all elastica, except for the circle, by solving

$$\ddot{\theta} = -\sin\theta. \tag{2.6}$$

2.2 Elliptic functions

In this section we introduce the elliptic functions defined by Jacobi and state some of their important properties. We also define the elliptic integrals of the first and second kind, which are essential in finding explicit parameterizations for the elastica. The elliptic functions and integrals have been covered in several works, e.g. [Lov20], [Gre59]. For the results in this section we mainly follow [Law89] and [Mey01].

Definition

Let $k \in [0, 1]$. The *elliptic functions* sn, cn and dn with *(elliptic) modulus* k are defined as the solutions to the system of differential equations

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}u} \operatorname{sn} u &= \operatorname{cn} u \operatorname{dn} u, & \operatorname{sn} 0 &= 0, \\ \frac{\mathrm{d}}{\mathrm{d}u} \operatorname{cn} u &= -\operatorname{sn} u \operatorname{dn} u, & \operatorname{cn} 0 &= 1, \\ \frac{\mathrm{d}}{\mathrm{d}u} \operatorname{dn} u &= -k^2 \operatorname{sn} u \operatorname{cn} u, & \operatorname{dn} 0 &= 1. \end{aligned} \tag{2.7}$$

We shall write $\operatorname{sn}(u, k)$, etc., when the dependence on the modulus is important, and otherwise simply use $\operatorname{sn} u$. The *complementary modulus* $k' \in [0, 1]$ is defined by

$$k^2 + k'^2 = 1. \tag{2.8}$$

Proposition 2.2. *For a fixed modulus k, the elliptic functions satisfy the following equations for all $u \in \mathbb{R}$:*

$$\begin{aligned} \operatorname{sn}^2 u + \operatorname{cn}^2 u &= 1, \\ \operatorname{dn}^2 u + k^2 \operatorname{sn}^2 u &= 1 \\ \operatorname{dn}^2 u - k^2 \operatorname{cn}^2 u &= k'^2. \end{aligned} \tag{2.9}$$

Proof. Define $A = \operatorname{sn}^2 + \operatorname{cn}^2$, $B = k^2 \operatorname{sn}^2 + \operatorname{dn}^2$, $C = \operatorname{dn}^2 - k^2 \operatorname{cn}^2$. From the system (2.7) we have

$$\frac{\mathrm{d}A}{\mathrm{d}u} = 2 \operatorname{sn} u \operatorname{sn}' u + 2 \operatorname{cn} u \operatorname{cn}' u = 2 \operatorname{sn} u \operatorname{cn} u \operatorname{dn} u - 2 \operatorname{cn} u \operatorname{sn} u \operatorname{dn} u = 0,$$

and likewise $\frac{\mathrm{d}B}{\mathrm{d}u} = \frac{\mathrm{d}C}{\mathrm{d}u} = 0$. Hence A, B and C are constant functions. We now apply the initial conditions and get for all $u \in \mathbb{R}$

$$\begin{aligned} \operatorname{sn}^2 u + \operatorname{cn}^2 u &= \operatorname{sn}^2 0 + \operatorname{cn}^2 0 = 1 \\ \operatorname{dn}^2 u + k^2 \operatorname{sn}^2 u &= \operatorname{dn}^2 0 + k^2 \operatorname{sn}^2 0 = 1 \\ \operatorname{dn}^2 u - k^2 \operatorname{cn}^2 u &= \operatorname{dn}^2 0 - k^2 \operatorname{cn}^2 0 = 1 - k^2 = k'^2. \end{aligned}$$ □

Define the *(elliptic) amplitude* am by

$$\operatorname{am}(u) = F^{-1}(u), \qquad F(u) = \int_0^u \frac{\mathrm{d}t}{\sqrt{1 - k^2 \sin^2 t}}.$$

We note that F is well-defined and strictly increasing, so the same holds for am.

Proposition 2.3. *The elliptic functions with modulus k can be expressed as*

$$\operatorname{sn}(u) = \sin(\operatorname{am} u), \quad \operatorname{cn}(u) = \cos(\operatorname{am} u), \quad \operatorname{dn}(u) = \sqrt{1 - k^2 \sin^2(\operatorname{am} u)}.$$

Moreover, the amplitude satisfies

$$\frac{\mathrm{d}}{\mathrm{d}u} \operatorname{am} u = \operatorname{dn} u, \quad \operatorname{am} 0 = 0.$$

Proof. Since (2.7) has a unique set of solutions it is enough to show that the functions above are such solutions. We have

$$\frac{\mathrm{d}}{\mathrm{d}u} \operatorname{am} u = \frac{1}{\frac{\mathrm{d}F}{\mathrm{d}u}(F^{-1}(u))} = \frac{1}{\frac{\mathrm{d}F}{\mathrm{d}u}(\operatorname{am} u)} = \sqrt{1 - k^2 \sin^2(\operatorname{am} u)}. \tag{2.10}$$

Hence

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}u} \sin(\operatorname{am} u) &= \cos(\operatorname{am} u)\sqrt{1 - k^2 \sin^2(\operatorname{am} u)}, \\ \frac{\mathrm{d}}{\mathrm{d}u} \cos(\operatorname{am} u) &= -\sin(\operatorname{am} u)\sqrt{1 - k^2 \sin^2(\operatorname{am} u)}, \\ \frac{\mathrm{d}}{\mathrm{d}u}\sqrt{1 - k^2 \sin^2(\operatorname{am} u)} &= \frac{-k^2 \sin(\operatorname{am} u) \frac{\mathrm{d}}{\mathrm{d}u} \sin(\operatorname{am} u)}{\sqrt{1 - k^2 \sin^2(\operatorname{am} u)}} = -k^2 \sin(\operatorname{am} u) \cos(\operatorname{am} u), \end{aligned}$$

and we have

$$\sin(\operatorname{am} 0) = 0, \quad \cos(\operatorname{am} 0) = 1, \quad \sqrt{1 - k^2 \sin^2(\operatorname{am} 0)} = 1.$$

From the above it is evident that $\frac{\mathrm{d}}{\mathrm{d}u} \operatorname{am} u = \operatorname{dn} u$ and since, obviously, $F(0) = 0$, we have $\operatorname{am} 0 = 0$. □

Proposition 2.4. *For the special moduli $k = 0$ and $k = 1$ we have*

$$\begin{aligned} \operatorname{sn}(u, 0) &= \sin u, & \operatorname{sn}(u, 1) &= \tanh u, \\ \operatorname{cn}(u, 0) &= \cos u, & \operatorname{cn}(u, 1) &= \operatorname{sech} u, \\ \operatorname{dn}(u, 0) &= 1, & \operatorname{dn}(u, 1) &= \operatorname{sech} u. \end{aligned}$$

Moreover, on compact intervals, the elliptic functions converge uniformly to these functions as $k \to 0^+$ and as $k \to 1^-$, respectively.

Proof. For $k = 0$, we have $F(u) = u$, so $\operatorname{am}(u, 0) = u$, and the above identities follow from Proposition 2.3. For the case $k = 1$, we observe that

$$\frac{\mathrm{d}}{\mathrm{d}u}\tanh u = \operatorname{sech}^2 u, \qquad \tanh(0) = 0,$$
$$\frac{\mathrm{d}}{\mathrm{d}u}\operatorname{sech} u = -\tanh u \operatorname{sech} u, \qquad \operatorname{sech}(0) = 1,$$

so the tuple $(\tanh, \operatorname{sech}, \operatorname{sech})$ satisfy (2.7) with $k = 1$.

The uniform convergence on compact sets follows from the ODE theorem on continuous dependence of solutions on the parameters, (see e.g. [Die60], p. 291). □

Elliptic integrals

The integral $F(u, k)$ given by the function F above is called *the incomplete elliptic integral of the first kind.* We define *the incomplete elliptic integral of the second kind* by

$$E(u, k) = \int_0^u \operatorname{dn}^2(t, k)\,\mathrm{d}t. \tag{2.11}$$

Proposition 2.5. *For any fixed modulus k, the functions* sn*,* am*, F and E are odd, while* cn *and* dn *are even functions.*

Proof. Using that $\sin^2$ is an even function, we have

$$F(-u) = \int_0^{-u} \frac{\mathrm{d}t}{\sqrt{1 - k^2 \sin^2(t)}} = -\int_0^{u} \frac{\mathrm{d}t}{\sqrt{1 - k^2 \sin^2(-t)}} = -F(u),$$

so F is odd, and, being the inverse of an odd function, so is am. The rest follows from sin and cos being respectively odd and even. □

We observe that for $k = 0$, E is the identity, while $E(u, 1) = \int_0^u \operatorname{sech}^2 t\,\mathrm{d}t = \tanh u$. Moreover, by Propositions 2.3–2.4 we must have

$$\operatorname{am}(u, 1) = \arcsin(\tanh u) = \int_0^u \operatorname{sech} t\,\mathrm{d}t.$$

This function is called the Gudermannian function[3] $\operatorname{gd}\colon \mathbb{R} \to \left]-\pi/2, \pi/2\right[$ and may also be expressed as

$$\operatorname{gd}(u) = \arctan(\sinh u) = \operatorname{sgn}(x) \arccos(\operatorname{sech} u) = 2\arctan(e^u) - \tfrac{\pi}{2}.$$

For $k = 1$, F is the inverse of the Gudermannian function, which is defined on the interval $\left]-\pi/2, \pi/2\right[$ by

$$F(u, 1) = \operatorname{gd}^{-1}(u) = \int_0^u \sec t\,\mathrm{d}t = \ln(\sec t + \tan t).$$

We have considered the elliptic functions and integrals for specific values of k, but for our application, we will need to know their dependence on k. In particular, we need their derivatives with respect to k. Deriving these is a technical exercise, which is included in Appendix A.

[3]The Gudermannian function was named so by Cayley [Cay62].

Addition formulas and periodicity

Proposition 2.6. *The elliptic functions satisfy the addition formulas*

$$\begin{aligned}
\operatorname{sn}(u+v) &= \frac{\operatorname{sn}u\operatorname{cn}v\operatorname{dn}v+\operatorname{sn}v\operatorname{cn}u\operatorname{dn}u}{1-k^2\operatorname{sn}^2u\operatorname{sn}^2v}\\
\operatorname{cn}(u+v) &= \frac{\operatorname{cn}u\operatorname{cn}v-\operatorname{sn}u\operatorname{sn}v\operatorname{dn}u\operatorname{dn}v}{1-k^2\operatorname{sn}^2u\operatorname{sn}^2v}\\
\operatorname{dn}(u+v) &= \frac{\operatorname{dn}u\operatorname{dn}v-k^2\operatorname{sn}u\operatorname{sn}v\operatorname{cn}u\operatorname{cn}v}{1-k^2\operatorname{sn}^2u\operatorname{sn}^2v}.
\end{aligned}$$

The proof requires some calculations, and we postpone it to Appendix B. Alternate proofs can be found e.g. in [WW62] and [Bow61].

We define the *quarter period* $K=K(k)$ by

$$\operatorname{am}(K)=\frac{\pi}{2},\quad \text{i.e.}\quad K=F\left(\frac{\pi}{2}\right),$$

so that $\operatorname{sn}K=1$, $\operatorname{cn}K=0$, and $\operatorname{dn}K=k'$.[4]

From the addition formulas it follows that

$$\begin{aligned}
\operatorname{sn}(u+K) &= \frac{\operatorname{sn}u\operatorname{cn}K\operatorname{dn}K+\operatorname{sn}K\operatorname{cn}u\operatorname{dn}u}{1-k^2\operatorname{sn}^2u\operatorname{sn}^2K} = \frac{\operatorname{cn}u\operatorname{dn}u}{1-k^2\operatorname{sn}^2u} = \frac{\operatorname{cn}u}{\operatorname{dn}u}\\
\operatorname{cn}(u+K) &= \frac{\operatorname{cn}u\operatorname{cn}K-\operatorname{sn}u\operatorname{sn}K\operatorname{dn}u\operatorname{dn}K}{1-k^2\operatorname{sn}^2u\operatorname{sn}^2K} = \frac{-k'\operatorname{sn}u\operatorname{dn}u}{1-k^2\operatorname{sn}^2u} = -k'\frac{\operatorname{sn}u}{\operatorname{dn}u}\\
\operatorname{dn}(u+K) &= \frac{\operatorname{dn}u\operatorname{dn}K-k^2\operatorname{sn}u\operatorname{sn}K\operatorname{cn}u\operatorname{cn}K}{1-k^2\operatorname{sn}^2u\operatorname{sn}^2K} = \frac{k'\operatorname{dn}u}{1-k^2\operatorname{sn}^2u} = \frac{k'}{\operatorname{dn}u}
\end{aligned}$$

and by using the above equations twice we have

$$\begin{aligned}
\operatorname{sn}(u+2K) &= \frac{\operatorname{cn}(u+K)}{\operatorname{dn}(u+K)} = -\operatorname{sn}u\\
\operatorname{cn}(u+2K) &= -k'\frac{\operatorname{sn}(u+K)}{\operatorname{dn}(u+K)} = -\operatorname{cn}u\\
\operatorname{dn}(u+2K) &= \frac{k'}{\operatorname{dn}(u+K)} = \operatorname{dn}u
\end{aligned}$$

From this we observe that dn is $2K$-periodic and by using the above twice it is evident that sn and cn are $4K$-periodic.

The elliptic integrals F and E do not have addition formulas, but we can find some useful relations. For $u\in\mathbb{R}$, we have

$$\begin{aligned}
F(u+\pi) &= \int_0^{u+\pi}\frac{\mathrm{d}t}{\sqrt{1-k^2\sin^2t}} = \int_{-\pi}^{u}\frac{\mathrm{d}v}{\sqrt{1-k^2\sin^2(v+\pi)}} = \int_{-\pi}^{u}\frac{\mathrm{d}v}{\sqrt{1-k^2\sin^2v}}\\
&= \int_{-\pi}^{0}\frac{\mathrm{d}v}{\sqrt{1-k^2\sin^2(v+\pi)}} + \int_{0}^{u}\frac{\mathrm{d}v}{\sqrt{1-k^2\sin^2(v+\pi)}} = -F(-\pi)+F(u)\\
&= F(u)+F(\pi).
\end{aligned}$$

For $u=-\pi/2$, we get $F(\pi/2)=F(-\pi/2)+F(\pi)$, so we have

$$F(\pi)=2F(\pi/2)=2K.$$

[4]The function $K(k)$ is also called the *complete elliptic integral of the first kind.* The *complete elliptic integral of the second kind* is the function $k\mapsto E(K(k),k)$.

Moreover, for $n \in \mathbb{Z}$ we get $F(u+n\pi) = F(u) + nF(\pi)$. In particular, we have

$$F(\operatorname{am} u + n\pi) = u + nF(\pi) = u + 2nK,$$

and by taking the amplitude on both sides, we obtain $\operatorname{am}(u+2nK) = \operatorname{am} u + n\pi$.

Turning our focus to E, we have for any $u \in \mathbb{R}$,

$$\begin{aligned} E(u+K) &= \int_0^{u+K} \operatorname{dn}^2 t \, \mathrm{d}t = \int_0^K \operatorname{dn}^2 t \, \mathrm{d}t + \int_K^{u+K} \operatorname{dn}^2 t \, \mathrm{d}t = E(K) + \int_0^u \operatorname{dn}^2(s+K) \, \mathrm{d}s \\ &= E(K) + \int_0^u \frac{k'^2}{\operatorname{dn}^2 s} \, \mathrm{d}s = E(K) + \left[E(s) - k^2 \frac{\operatorname{sn} s \operatorname{cn} s}{\operatorname{dn} s} \right]_0^u \\ &= E(K) + E(u) - k^2 \frac{\operatorname{sn} u \operatorname{cn} u}{\operatorname{dn} u} \end{aligned}$$

Using this and the fact that E is odd, it follows that for all $n \in \mathbb{Z}$ we have

$$E(nK) = nE(K), \qquad E(u+2nK) = E(u) + 2nE(K).$$

Extension of k-domain

The elliptic functions, as we have defined them, are only valid for $k \in [0,1]$. However, by analytic continuation (see e.g. Lawden [Law89]), the domain of k may be extended. For $k > 1$, the following identities hold for all $u \in \mathbb{R}$:

$$\begin{aligned} \operatorname{sn}(u,k) &= \tfrac{1}{k} \operatorname{sn}(ku, \tfrac{1}{k}), \\ \operatorname{cn}(u,k) &= \operatorname{dn}(ku, \tfrac{1}{k}), \\ \operatorname{dn}(u,k) &= \operatorname{cn}(ku, \tfrac{1}{k}), \\ E(u,k) &= kE(ku, \tfrac{1}{k}) + u(1-k^2). \end{aligned}$$

Observe that $K(k) \to \infty$ as $k \to 1$, so we cannot extend K continuously. We choose the extension

$$K(k) = \tfrac{1}{k} K\left(\tfrac{1}{k}\right), \quad k > 1,$$

which ensures that the period of sn is always $4K$, while the periods for cn and dn are $2K$ and $4K$, respectively, when $k > 1$. We stress that this is *not* the analytic continuation of K, which in fact takes non-real values for $k > 1$. With this choice of K-extension the above results for E are still valid for $k > 1$. Moreover, it ensures that the derivative of K has the same expression for all $k \neq 1$, see Appendix A.

Remark. When $k > 1$ we cannot define the complimentary modulus k' as we have done above, so we will let $k' \in [0,1] \cup i\,[0,\infty[$ such that $k^2 + k'^2 = 1$. However, we will only need the number k'^2, which is real for all k.

Bugs in Python functions

For the implementation, we have used the elliptic functions included in the SciPy library. Unfortunately, these do not always provide correct values. Specifically the value for $\operatorname{dn}(K(k),k)$ is erroneous for most k. We filed a bug report explaining this, which is included in Appendix C. Since then, bugs have also been discovered in the elliptic integral of the second kind $E(u,k)$.

We have not found a better Python library, but we have tried to circumvent this problem, when necessary. For example one can use $\operatorname{dn} = \sqrt{1 - k^2 \operatorname{sn}^2}$, since there are apparently no bugs in SciPy's sn.

2.3 Explicit parameterization

The solutions to (2.6) can be expressed in closed form via the elliptic functions, and we shall derive them in the following. There are two classes of elastica: curves with inflection points (i.e. where $\dot{\theta} = 0$) and curves without inflections.

Inflectional elastica

We consider an elastica with angle function θ. Let us assume that $\theta(0) = 0$ and that $\dot{\theta}(0) > 0$; in particular $s = 0$ is not an inflection point. Moreover, we assume that $s_0 > 0$ is the first positive inflection point, i.e. $\dot{\theta}(s_0) = 0$. We now know that $\theta(s_0) > 0$.

For all $s \in \mathbb{R}$ we have

$$\begin{aligned}\dot{\theta}(s)^2 = \dot{\theta}(s_0)^2 + \int_{s_0}^{s} \frac{\mathrm{d}}{\mathrm{d}t}\left(\dot{\theta}(t)^2\right) \mathrm{d}t &= \dot{\theta}(s_0)^2 + 2\int_{s_0}^{s} \dot{\theta}(t)\ddot{\theta}(t)\,\mathrm{d}t = -2\int_{s_0}^{s} \dot{\theta}(t)\sin\theta(t)\,\mathrm{d}t \\ &= -2\int_{\theta(s_0)}^{\theta(s)} \sin u\,\mathrm{d}u = 2\left(\cos\theta(s) - \cos\theta(s_0)\right) = 4\left(\sin^2 \tfrac{1}{2}\theta(s_0) - \sin^2 \tfrac{1}{2}\theta(s)\right),\end{aligned}$$

hence

$$\dot{\theta}(s) = \pm 2\sqrt{\sin^2 \tfrac{1}{2}\theta(s_0) - \sin^2 \tfrac{1}{2}\theta(s)}.$$

For $s \in [0, s_0]$ we may take the positive square root because of our assumption that $\dot{\theta}(0) > 0$. Moreover, it follows that $\theta(s_0) \leq \pi$, since otherwise we could find an $s \in [0, s_0]$ for which the square root would not be well-defined.

For $s \in [0, s_0]$ we have

$$s_0 - s = \int_s^{s_0} \mathrm{d}t = \frac{1}{2}\int_s^{s_0} \frac{\dot{\theta}(t)}{\sqrt{\sin^2 \frac{1}{2}\theta(s_0) - \sin^2 \frac{1}{2}\theta(t)}}\,\mathrm{d}t.$$

Setting $k = \sin \frac{1}{2}\theta(s_0)$ and substituting $\varphi = \arcsin\left(\frac{1}{k}\sin\frac{1}{2}\theta(t)\right)$ we find

$$\frac{\mathrm{d}\varphi}{\mathrm{d}t} = \frac{\dot{\theta}(t)\cos\frac{1}{2}\theta(t)}{2k\sqrt{1 - k^{-2}\sin^2\frac{1}{2}\theta(t)}} = \frac{\dot{\theta}(t)\sqrt{1 - \sin^2\frac{1}{2}\theta(t)}}{2\sqrt{k^2 - \sin^2\frac{1}{2}\theta(t)}} = \frac{\dot{\theta}(t)\sqrt{1 - k^2\sin^2\varphi}}{2\sqrt{k^2 - \sin^2\frac{1}{2}\theta(t)}},$$

(where we have used that $\cos\frac{1}{2}\theta(t) > 0$) and thus

$$s_0 - s = \int_{\arcsin\left(\frac{1}{k}\sin\frac{1}{2}\theta(s)\right)}^{\arcsin\left(\frac{1}{k}\sin\frac{1}{2}\theta(s_0)\right)} \frac{\mathrm{d}\varphi}{\sqrt{1 - k^2\sin^2\varphi}} = \int_{\psi(s)}^{\frac{\pi}{2}} \frac{\mathrm{d}\varphi}{\sqrt{1 - k^2\sin^2\varphi}},$$

with $\psi(s) = \arcsin\left(\frac{1}{k}\sin\frac{1}{2}\theta(s)\right)$. Recalling the definition of F and using the above we have

$$F(\psi(s), k) = \int_0^{\psi(s)} \frac{\mathrm{d}\varphi}{\sqrt{1 - k^2\sin^2\varphi}} = \int_0^{\frac{\pi}{2}} \frac{\mathrm{d}\varphi}{\sqrt{1 - k^2\sin^2\varphi}} + \int_{\frac{\pi}{2}}^{\psi(s)} \frac{\mathrm{d}\varphi}{\sqrt{1 - k^2\sin^2\varphi}} = K + s - s_0,$$

where K is the quarter period corresponding to modulus k.

It follows that $\psi(s) = \mathrm{am}(s - s_0 + K, k)$, so

$$\sin\tfrac{1}{2}\theta(s) = k\sin\psi(s) = k\,\mathrm{sn}(s - s_0 + K).$$

Since $\theta(0) = 0$, we conclude that $s_0 = K$, so we get $\sin\frac{1}{2}\theta(s) = k\,\mathrm{sn}(s)$. Differentiating we find

$$\tfrac{1}{2}\dot{\theta}(s)\cos\tfrac{1}{2}\theta(s) = k\,\mathrm{cn}(s)\,\mathrm{dn}(s)$$

and since

$$\cos \tfrac{1}{2}\theta(s) = \sqrt{1-\sin^2 \tfrac{1}{2}\theta(s)} = \sqrt{1-k^2\,\mathrm{sn}^2(s)} = \mathrm{dn}(s)$$

we get $\dot{\theta}(s) = 2k\,\mathrm{cn}(s)$. Integrating this we find

$$\theta(s) = \theta(0) + \int_0^s \dot{\theta}(t)\,\mathrm{d}t = 2k\int_0^s \mathrm{cn}(t)\,\mathrm{d}t = 2k\left[\tfrac{1}{k}\arcsin(k\,\mathrm{sn}\,t)\right]_0^s = 2\arcsin(k\,\mathrm{sn}(s)).$$

We have found the angle function, and since any curve satisfies $\dot{\boldsymbol{\gamma}} = (\cos\theta, \sin\theta)$, we have

$$\begin{aligned}\dot{x}(s) &= \cos\theta(s) = 1 - 2\sin^2\tfrac{1}{2}\theta(s) = 1-2k^2\,\mathrm{sn}^2(s) = 2\,\mathrm{dn}^2\,s - 1,\\ \dot{y}(s) &= \sin\theta(s) = 2\sin\tfrac{1}{2}\theta(s)\cos\tfrac{1}{2}\theta(s) = 2k\,\mathrm{sn}(s)\,\mathrm{dn}(s).\end{aligned}$$

Hence an inflectional elastica that begins at $(0,0)$ has the parameterization

$$\begin{aligned}x(s) &= \int_0^s \left(2\,\mathrm{dn}^2\,t - 1\right)\,\mathrm{d}t = 2E(s,k) - s,\\ y(s) &= 2k\int_0^s \mathrm{sn}\,t\,\mathrm{dn}\,t\,\mathrm{d}t = 2k\left(1-\mathrm{cn}(s,k)\right).\end{aligned} \tag{2.12}$$

We see that we get a one-parameter family of curves, specified by the elliptic modulus $k = \frac{1}{2}\dot{\theta}(0)$. We note that for $k=0$ we get the straight line $(x,y) = (s,0)$. For $k=1$ we get

$$(x(s)\,,\,y(s)) = (2\tanh(s) - s\,,\,2(1-\mathrm{sech}(s))\,.$$

Non-inflectional elastica

For the elastica without inflections, one can derive the parameterization similarly to the argument above. A non-inflectional elastica starting at $(0,0)$ and with initial tangent angle $\theta(0)=0$ and initial curvature $\dot{\theta}(0) = 2/k$ has the parameterization

$$\begin{pmatrix}x(s)\\ y(s)\end{pmatrix} = \begin{pmatrix}\frac{2}{k}E(\frac{s}{k},k) + \left(1-\frac{2}{k^2}\right)s\\ \frac{2}{k}\left(1-\mathrm{dn}\left(\frac{s}{k},k\right)\right)\end{pmatrix}, \tag{2.13}$$

so again we have a family parameterized by the elliptic modulus $k\in]0,1]$.

We observe that for $k=1$ we get $(2\tanh(s)-s, 2(1-\mathrm{sech}(s))$ as in the inflectional case. We will investigate the limit as $k\to 0$. First, note that the "height" of the curve is $y_{\max} = \frac{2}{k}(1-\sqrt{1-k^2})$. By l'Hospital's rule we get

$$\lim_{k\to 0}\tfrac{2}{k}(1-\sqrt{1-k^2}) = \lim_{k\to 0}\frac{2k}{\sqrt{1-k^2}} = 0,$$

so curve height becomes arbitrarily small as $k\to\infty$.

By differentiating (2.13) twice, we find that the curvature is $\kappa_k(s) = \frac{2}{k}\,\mathrm{dn}\left(\frac{s}{k},k\right)$, which goes to infinity as $k\to 0$. However, if we scale the elastic curve by the factor $2/k$, while preserving unit speed, we get the curvature $\tilde{\kappa}_k(s) = \mathrm{dn}\left(\frac{s}{2},k\right)$. For s in a compact interval, $\tilde{\kappa}_k(s)\to 1$ uniformly, which means that any finite segment in the k-family of scaled elastica approaches a piece of a circle of radius 1 as $k\to 0$.

Combining the families

Most literature that involves the explicit parameterizations, consider the two families of elastica separately. It turns out, however, that we can describe them as one family using the analytic continuation of the elliptic functions. If we use the extensions, (2.12) becomes

$$\begin{aligned} x(s) &= 2kE(ks, \tfrac{1}{k}) + s(1-2k^2), \\ y(s) &= 2k\left(1 - \operatorname{dn}(ks, \tfrac{1}{k})\right), \end{aligned}$$

which is exactly (2.13) with k replaced by $1/k$. By letting $k \in [0, \infty[$ we can thus describe all elastica, except the circle, as one family, where $k = \frac{1}{2}\dot{\theta}(0)$ defines the initial curvature. We will reserve the name $\boldsymbol{\zeta}_k$ to denote these *basic elastica*, i.e. those unit speed curves that solve (2.6), start at $(0,0)$ and have $\dot{\theta}(0) \geq 0$. We thus define

$$\boldsymbol{\zeta}_k(s) = \boldsymbol{\zeta}(s,k) = \begin{pmatrix} 2E(s,k) - s \\ 2k\left(1 - \operatorname{cn}(s,k)\right) \end{pmatrix}, \quad s \in \mathbb{R},\ k \geq 0.$$

Figure 2.1 shows elastic curves for different values of k. All elastica are obtained by scaling and rotating these curves. The curves are clearly translation periodic, i.e. the y-coordinate is periodic in s, and the period must be that of cn, i.e. $4K$ if $k < 1$ and $2K$ if $k > 1$.

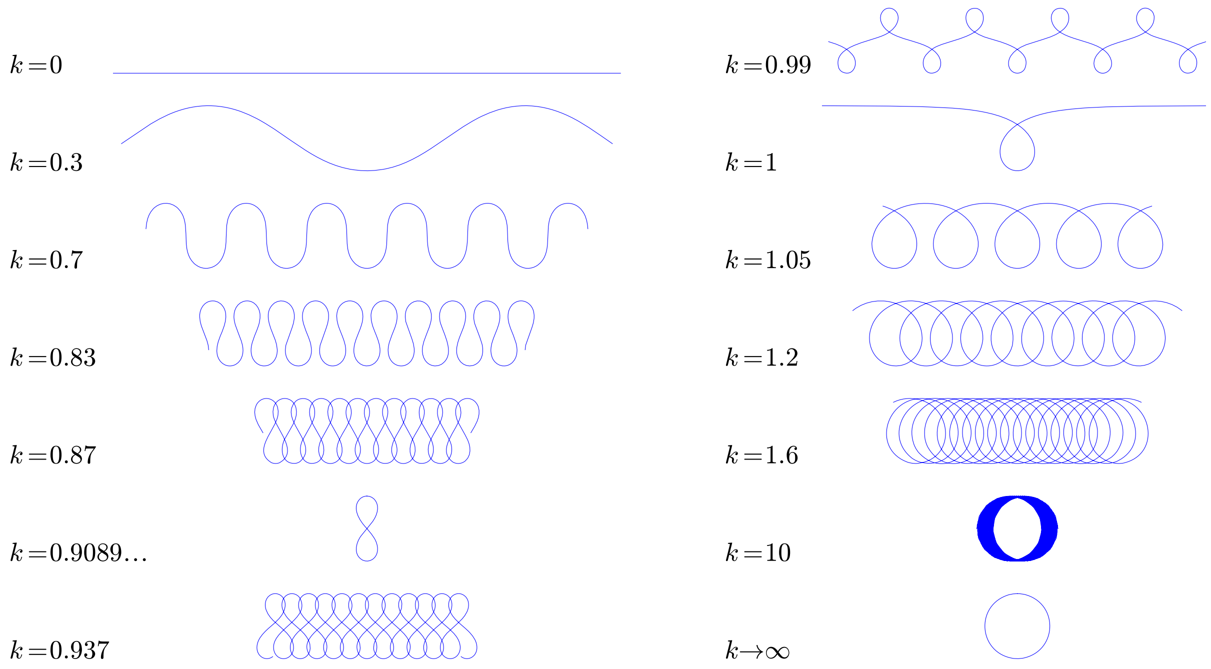


Figure 2.1: Elastica for different values of k. The curves are scaled to a uniform "height".

When we derived the parameterization for the inflectional elastica we found the angle function to be $\theta(s,k) = 2\arcsin\left(k \operatorname{sn}(s,k)\right)$, which shows that the maximal angle is $2\arcsin k$. For the non-inflectional elastica, this formula is not valid, since these all have increasing angle functions. Going back to the extension of sn we see that, when $k > 1$,

$$\theta(s,k) = 2\arcsin\left(k \operatorname{sn}(s,k)\right) = 2\arcsin\left(\operatorname{sn}(ks, \tfrac{1}{k})\right) = 2\operatorname{am}(ks, \tfrac{1}{k}).$$

The last equality is only valid for $\operatorname{am}(ks, \frac{1}{k}) \in [-\pi/2, \pi/2]$, but the calculation still leads us to the correct angle function for the non-inflectional elastica, namely $\theta(s,k) = 2\operatorname{am}(ks, \frac{1}{k})$.

Finally we note that the curvature function

$$\kappa(s) = \dot{\theta}(s) = 2k\operatorname{cn}(s,k) \tag{2.14}$$

is valid for all $k \geq 0$.

General elastica segments

We now want to define some suitable parameters to describe an arbitrary elastic curve segment. We can chose a segment of a basic elastica by choosing k, a starting point s_0 and a length $\ell > 0$ and taking $\boldsymbol{\zeta}_k|_{[s_0,s_0+l]}$. It will be convenient to remove the dependence on s_0 and ℓ from the domain and integrate them in the parameterization. We parameterize our elastica segments on the unit interval, setting $s = s_0 + \ell t$, with $t \in [0,1]$. The new curve parameter t is not unit speed. Finally, any elastica segment can be obtained by introducing a scaling factor $S > 0$, a rotation by an angle $\phi \in]-\pi,\pi]$ and translation by a vector $(\hat{x},\hat{y})$. We thus have a standard elastic segment parameterization

$$\boldsymbol{\gamma}_{(k,s_0,\ell,S,\phi,\hat{x},\hat{y})}(t) = SR_\phi\boldsymbol{\zeta}_k(s_0+\ell t) + (\hat{x},\,\hat{y}), \quad t \in [0,1].$$

It depends on seven *control parameters*, but we will often omit the subscript. Such a curve has constant speed $|\ell|S$ and length $L = |\ell|S$.

Remark. If $\ell < 0$, the orientation of the curve is changed. In the inflectional case, the elastica with opposite orientation can be obtained by a rotation by π. In this case we may therefore assume $\ell > 0$ without loss of generality. For elastica *without* inflections, however, a segment with $\ell < 0$ cannot be described as a segment with $\ell > 0$. One can instead reverse the direction of the parameterization for that case, and so all cases can be handled with the assumption $\ell > 0$.

Remark. The same elastic curve segment can be parameterized in different ways because of the periodicity. However if we demand that s_0 is nonnegative and less than a period and that the rotation $\phi \in]-\pi,\pi]$, we have a unique parameterization of any elastic curve segment in the plane.

For any elastic curve $\boldsymbol{\gamma}$ of the above type, the curve $\tilde{\boldsymbol{\gamma}}(t) = \boldsymbol{\gamma}(\frac{t}{\ell S})$ is unit speed. Letting θ and $\tilde{\theta}$ denote the angle functions of $\boldsymbol{\zeta}_k$ and $\tilde{\boldsymbol{\gamma}}$, respectively, we have $\tilde{\theta}(t) = \theta(s_0 + \frac{t}{S}) + \phi$, and thus

$$\tilde{\theta}''(t) = \tfrac{1}{S^2}\ddot{\theta}(s_0 + \tfrac{t}{S}) = -\tfrac{1}{S^2}\sin\theta\,(s_0+\tfrac{t}{S}) = -\tfrac{1}{S^2}\sin\left(\tilde{\theta}(t)-\phi\right),$$

so the angle function for $\tilde{\boldsymbol{\gamma}}$ satisfies (2.4) with

$$(\lambda_1,\,\lambda_2) = \tfrac{1}{S^2}(\cos\phi,\,\sin\phi). \tag{2.15}$$

2.4 Finding the control parameters of an elastic curve segment

We will now describe a way to calculate numerically the control parameters of a given planar elastic curve segment. In Chapter 3 the same recipe will be applied to an arbitrary planar curve to obtain a canonical first guess for an approximating elastic curve. The main idea is to exploit the fact that the curvature of an elastica is an affine function of the distance along a special direction.

Let $\mathbf{x}\colon [a,b] \to \mathbb{R}^2$ be an elastic curve parameterized by arc length. As for any planar curve, we can write the tangent and the normal as $\mathbf{t} = (\cos\theta,\sin\theta)$, $\mathbf{n} = (-\sin\theta,\cos\theta)$, and we have the Frenet-Serret equations

$$\frac{\mathrm{d}\mathbf{t}}{\mathrm{d}s} = \frac{\mathrm{d}\theta}{\mathrm{d}s}\mathbf{n} = \kappa\mathbf{n}, \quad \frac{\mathrm{d}^2\mathbf{t}}{\mathrm{d}s^2} = \frac{\mathrm{d}^2\theta}{\mathrm{d}s^2}\mathbf{n} - \kappa^2\mathbf{t}.$$

The tangent angle θ must satisfy the Euler-Lagrange equation (2.4) for some Lagrangian multipliers λ_1, λ_2 to be found.

Let u denote the projection of $\mathbf{x} = (x, y)$ onto the line spanned by $(\lambda_2, -\lambda_1)$, i.e.

$$u = \frac{1}{\lambda}(\lambda_2, -\lambda_1) \cdot (x, y) = \frac{\lambda_2 x - \lambda_1 y}{\lambda},$$

where $\lambda = \|(\lambda_1, \lambda_2)\| = S^{-2}$. Setting $\phi = 0$ in (2.15), we find that the vector $(\lambda_2, -\lambda_1)$ points in the downward direction in Figure 2.1. It follows that u is bounded and periodic in s. Moreover, we can write the Euler-Lagrange equation as $\ddot{\theta} = \lambda \dot{u}$, so we have

$$\kappa = \frac{\mathrm{d}\theta}{\mathrm{d}s} = \lambda u + \alpha = \lambda_2 x - \lambda_1 y + \alpha, \tag{2.16}$$

which is to say that the curvature is an affine function of u.

In order to find λ_1, λ_2 and α in a numerically stable manner, we solve the above equation in the least squares sense, i.e. we consider the quadratic minimization problem

$$\underset{\lambda_1, \lambda_2, \alpha}{\text{minimize}} \int_a^b (\kappa + \lambda_1 y - \lambda_2 x - \alpha)^2 \, \mathrm{d}s,$$

which leads to the following linear system

$$\begin{pmatrix} \int_a^b y^2 \,\mathrm{d}s & -\int_a^b xy \,\mathrm{d}s & -\int_a^b y \,\mathrm{d}s \\ -\int_a^b xy \,\mathrm{d}s & \int_a^b x^2 \,\mathrm{d}s & \int_a^b x \,\mathrm{d}s \\ -\int_a^b y \,\mathrm{d}s & \int_a^b x \,\mathrm{d}s & \int_a^b 1 \,\mathrm{d}s \end{pmatrix} \begin{pmatrix} \lambda_1 \\ \lambda_2 \\ \alpha \end{pmatrix} = \begin{pmatrix} -\int_a^b y\kappa \,\mathrm{d}s \\ \int_a^b x\kappa \,\mathrm{d}s \\ \int_a^b \kappa \,\mathrm{d}s \end{pmatrix}. \tag{2.17}$$

Let θ_u denote the angle between the tangent vector $\mathbf{t}$ and the u-axis (see Figure 2.2). We have

$$\begin{aligned} \cos\theta_u &= \frac{1}{\lambda}(\lambda_2, -\lambda_1) \cdot \mathbf{t} = \frac{1}{\lambda}(\lambda_2, -\lambda_1) \cdot \frac{\mathrm{d}\mathbf{x}}{\mathrm{d}s} = \frac{\mathrm{d}u}{\mathrm{d}s}, \\ \sin\theta_u &= \frac{1}{\lambda}(\lambda_1, \lambda_2) \cdot \mathbf{t}, \end{aligned} \tag{2.18}$$

and hence

$$\frac{\mathrm{d}\sin\theta_u}{\mathrm{d}u} = \frac{\mathrm{d}s}{\mathrm{d}u}\frac{\mathrm{d}\sin\theta_u}{\mathrm{d}s} = \frac{1}{\cos\theta_u}\cos\theta_u \frac{\mathrm{d}\theta}{\mathrm{d}s} = \kappa = \lambda u + \alpha,$$

or equivalently

$$P(u) := \sin\theta_u = \frac{1}{2}\lambda u^2 + \alpha u + \beta. \tag{2.19}$$

We solve this equation with respect to β in the least squares sense and obtain

$$\beta = \frac{1}{L}\int_a^b \left(\sin\theta_u - \frac{1}{2}\lambda u^2 - \alpha u \right) \mathrm{d}s, \tag{2.20}$$

where $L = b - a$ is the length of the curve $\mathbf{x}$.

For a unit speed elastica $\mathbf{x}(s) = S R_\phi \boldsymbol{\zeta}_k(s/S) + (\hat{x}, \hat{y})$ we have $(\lambda_1, \lambda_2) = S^{-2}(\cos\phi, \sin\phi)$ and $\kappa(s) = (2k/S)\,\mathrm{cn}(s/S)$. Substituting these into the definitions $u = (\lambda_2 x - \lambda_1 y)/\lambda$, $\alpha = \kappa - \lambda u$ and $\sin\theta_u = (\lambda_1, \lambda_2)/\lambda \cdot \mathbf{t}$ we have

$$\begin{aligned} u &= -2Sk(1 - \mathrm{cn}(s/S)) + \hat{x}\sin\phi - \hat{y}\cos\phi, \\ \sin\theta_u &= 2\,\mathrm{dn}^2(s/S) - 1, \\ \alpha &= 2kS^{-1}\,\mathrm{cn}(s/S) - \lambda u = 2k/S + (\hat{y}\cos\phi - \hat{x}\sin\phi)/S^2. \end{aligned} \tag{2.21}$$

The equation $\beta = \sin\theta_u - \lambda u^2/2 - \alpha u$ then becomes:

$$\beta = 1 + \frac{\hat{x}\sin\phi - \hat{y}\cos\phi}{2S^2}(\hat{x}\sin\phi - \hat{y}\cos\phi - 4kS). \tag{2.22}$$

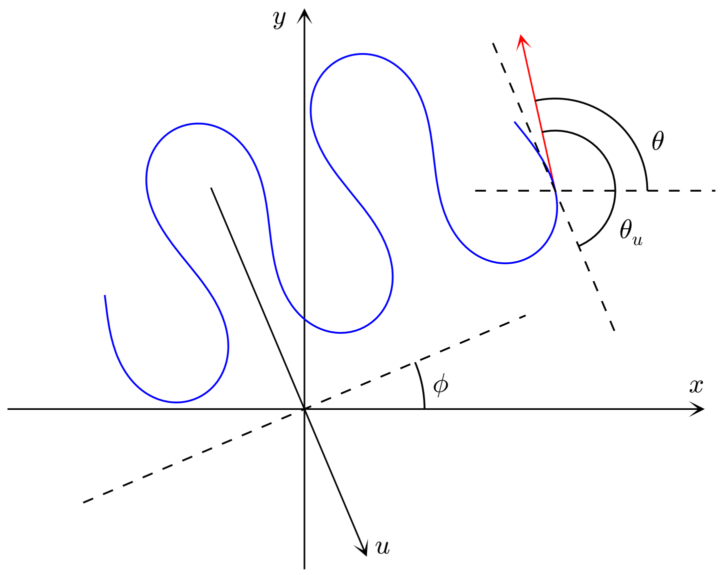


Figure 2.2: A segment of an elastica in the plane, showing ϕ, θ, the u-axis and the angle θ_u.

It follows from (2.19) that all points on the elastica correspond to u-values where the value of the polynomial P is between -1 and 1, and hence

$$u \in \left[\frac{-\alpha-\delta_-}{\lambda}, \frac{-\alpha+\delta_-}{\lambda}\right],$$

where $\delta_- = \sqrt{\alpha^2 - 2\lambda(\beta-1)}$.

If the elastica has an inflection, there must be some u_*, such that $\kappa = \lambda u_* + \alpha = 0$, but this means that u_* is a minimizer for $P(u)$, and thus its minimum must lie in $[-1, 1]$. Moreover, the inflectional elastica has points where $\sin\theta_u = 1$ (which happens twice per period), but no points where $\sin\theta_u = -1$ (see Figure 2.3). Hence u runs through all of the interval where $P(u)$ is less that 1; in other words, $u_{\min}$ and $u_{\max}$ are exactly the endpoints of the above interval.

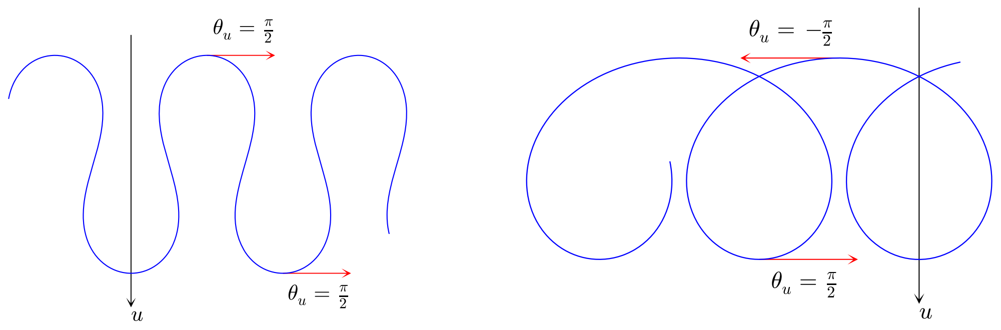

$\theta_u = \frac{\pi}{2}$
$\theta_u = \frac{\pi}{2}$
u
$\theta_u = -\frac{\pi}{2}$
$\theta_u = \frac{\pi}{2}$
u

Figure 2.3: The inflectional elastica have points where $\sin\theta_u = 1$, but not -1. For the non-inflectional elastica $\sin\theta_u$ takes both the values ± 1.

For the elastica without inflections, the tangent makes full rotations, so $\sin\theta_u$ takes both of the values ± 1. Hence, in this case, we must have

$$[u_{\min}, u_{\max}] = \left[\frac{-\alpha-\delta_-}{\lambda}, \frac{-\alpha-\delta_+}{\lambda}\right] \quad \text{or} \quad [u_{\min}, u_{\max}] = \left[\frac{-\alpha+\delta_+}{\lambda}, \frac{-\alpha+\delta_-}{\lambda}\right],$$

where $\delta_+ = \sqrt{\alpha^2 - 2\lambda(\beta+1)}$; these are the two cases corresponding to $\ell < 0$ and $\ell > 0$, respectively. We can thus determine whether the elastica has inflection points based on whether the minimum for the polynomial $P(u)$ is smaller or greater than -1, see Figure 2.4. In fact, from (2.21) and (2.22) we have

$$\alpha^2 - 2\lambda(\beta - 1) = \frac{4k^2}{S^2},$$

or equivalently

$$k = \frac{\sqrt{\alpha^2 - 2\lambda(\beta - 1)}}{2\sqrt{\lambda}}, \tag{2.23}$$

so we can find S, ϕ and k from λ_1, λ_2, α and β.

Remark. The above formula also holds if $\ell < 0$ and so does the expression for β in control parameters. The expressions for κ, λ_1, λ_2 and α simply change sign in this case.

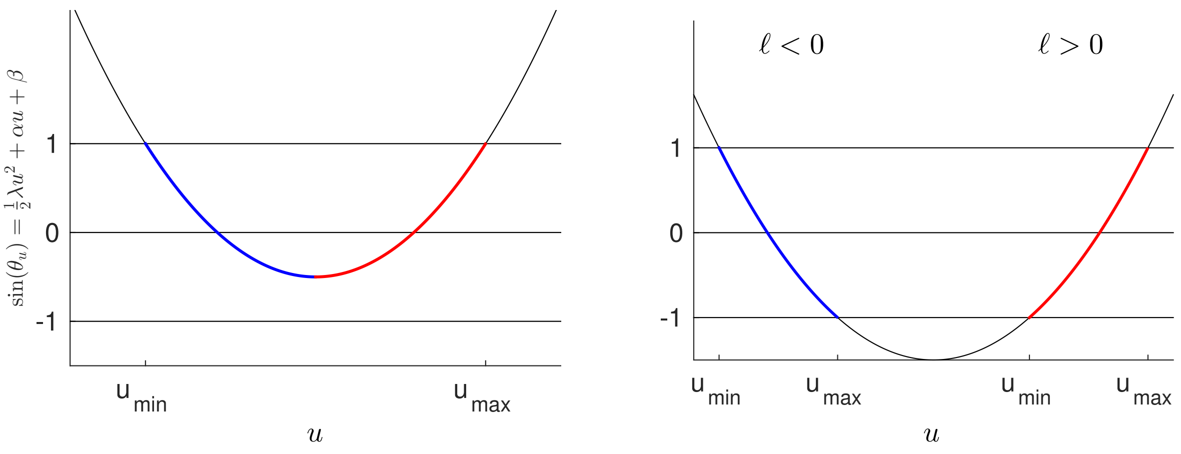


Figure 2.4: The parabola (2.19). To the left in the case of an elastica with inflection points, to the right without. The blue and red part corresponds to points with negative and positive curvature, respectively.

We still need to recover s_0 and ℓ. We have

$$u = -2kS\left(1 - \operatorname{cn}\left(s_0 + \tfrac{t}{S}\right)\right) + \hat{x}\sin\phi - \hat{y}\cos\phi,$$

and since

$$u_{\max} = \frac{-\alpha + \delta_-}{\lambda} = \hat{x}\sin(\phi) - \hat{y}\cos(\phi),$$

we get

$$\Delta(u) = u_{\max} - u = 2kS\left(1 - \operatorname{cn} s\right),$$

so

$$\operatorname{cn}(s, k) = 1 - \frac{\Delta(u)}{2kS}. \tag{2.24}$$

If we consider the unbounded complete elastica, then u oscillates between $u_{\min}$ and $u_{\max}$ and we can divide the elastica into segments where u is monotone, each with length equal to a half period $2KS$.

We first consider the case of an elastica with inflection points (i.e. $k < 1$). Here we have $\operatorname{cn}(s, k) = \cos(\operatorname{am}(s, k))$. If the start point $\mathbf{x}_0 = \mathbf{x}(a)$ is on segment number 1 and u is decreasing here, then

$$\operatorname{am}(s_0, k) = \arccos\left(1 - \frac{\Delta(u_0)}{2kS}\right),$$

and if the end point $\mathbf{x}_1 = \mathbf{x}(b)$ is on segment number n, then

$$\operatorname{am}(s_1, k) = \begin{cases} (n-1)\pi + \arccos\left(1 - \frac{\Delta(u_1)}{2kS}\right), & \text{if } n \text{ is odd,} \\ n\pi - \arccos\left(1 - \frac{\Delta(u_1)}{2kS}\right), & \text{if } n \text{ is even.} \end{cases}$$

If u is increasing on segment number 1, then

$$\operatorname{am}(s_0, k) = 2\pi - \arccos\left(1 - \frac{\Delta(u_0)}{2kS}\right),$$

and

$$\operatorname{am}(s_1, k) = \begin{cases} (n+1)\pi - \arccos\left(1 - \frac{\Delta(u_1)}{2kS}\right), & \text{if } n \text{ is odd,} \\ n\pi + \arccos\left(1 - \frac{\Delta(u_1)}{2kS}\right), & \text{if } n \text{ is even.} \end{cases}$$

In all cases we have

$$s_i = F(\operatorname{am}(s_i, k), k), \qquad i = 0, 1,$$

and $\ell = s_1 - s_0$.

In the case of an elastica without inflection points (i.e. $k \geq 1$) we need a little work to find am. We have

$$\operatorname{sn}(s, k) = \frac{1}{k} \operatorname{sn}\left(ks, \tfrac{1}{k}\right) = \frac{1}{k} \sin\left(\operatorname{am}\left(ks, \tfrac{1}{k}\right)\right)$$

and

$$\operatorname{sn}(s, k) = \sqrt{1 - \operatorname{cn}^2(s, k)} = \sqrt{\frac{\Delta(u)}{kS}\left(1 - \frac{\Delta(u)}{4kS}\right)}.$$

If u is decreasing on segment 1 then

$$\operatorname{am}\left(ks_0, \tfrac{1}{k}\right) = \arcsin\sqrt{\frac{\Delta(u_0)}{S}\left(k - \frac{\Delta(u_0)}{4S}\right)},$$

and if we have n segments

$$\operatorname{am}\left(ks_1, \tfrac{1}{k}\right) = \begin{cases} \frac{n-1}{2}\pi + \arcsin\sqrt{\frac{\Delta(u_1)}{S}\left(k - \frac{\Delta(u_1)}{4S}\right)}, & \text{if } n \text{ is odd,} \\ \frac{n}{2}\pi - \arcsin\sqrt{\frac{\Delta(u_1)}{S}\left(k - \frac{\Delta(u_1)}{4S}\right)}, & \text{if } n \text{ is even.} \end{cases}$$

If u is increasing on segment 1 then

$$\operatorname{am}\left(ks_0, \tfrac{1}{k}\right) = \pi - \arcsin\sqrt{\frac{\Delta(u_0)}{S}\left(k - \frac{\Delta(u_0)}{4S}\right)},$$

and if we have n segments

$$\operatorname{am}\left(ks_1, \tfrac{1}{k}\right) = \begin{cases} \frac{n+1}{2}\pi - \arcsin\sqrt{\frac{\Delta(u_1)}{S}\left(k - \frac{\Delta(u_1)}{4S}\right)}, & \text{if } n \text{ is odd,} \\ \frac{n}{2}\pi + \arcsin\sqrt{\frac{\Delta(u_1)}{S}\left(k - \frac{\Delta(u_1)}{4S}\right)}, & \text{if } n \text{ is even.} \end{cases}$$

Finally, we find the s-values using the incomplete elliptic integral

$$s_i = \tfrac{1}{k} F\left(\operatorname{am}\left(ks_i, \tfrac{1}{k}\right), \tfrac{1}{k}\right), \qquad i = 0, 1,$$

and $\ell = s_1 - s_0$.

Remark. If we have a negatively curved noninflectional elastica (i.e. $\ell < 0$), we can reverse the parameterization, find the elastica, and interchange (s_0, s_1).

We now have a scaled and rotated elastica segment, $\boldsymbol{\gamma}_0 = \boldsymbol{\gamma}_{(k,s_0,\ell,S,\phi,0,0)}$, and all that is left is to find the final translation $(\hat{x}, \hat{y})$. This is done by solving the equation

$$\mathbf{x}(s) = \boldsymbol{\gamma}_0(s) + (\hat{x}, \hat{y}),$$

in the least squares sense. The solution is

$$(\hat{x}, \hat{y}) = \frac{1}{L} \int_a^b (\mathbf{x}(s) - \boldsymbol{\gamma}_0(s))\,\mathrm{d}s. \tag{2.25}$$

2.5 Improved algorithm

In the above algorithm, the methods for finding α, β and λ (and thus k, S and ϕ) are based on integrals and thus numerically stable. The method for finding s_0 and ℓ however is based on the specific end tangents of the input curve $\mathbf{x}$. If the input is perturbed, the result will change. For this reason we will here describe an alternative method for finding s_0 and ℓ.

We set $\ell = L/S$, which guarantees the correct curve length. In order to find s_0, we minimize the difference between the tangent angles of the elastic curves $\mathbf{x}$ and $\boldsymbol{\gamma}_{s_0} = \boldsymbol{\gamma}_{(k,s_0,\ell,S,\phi,0,0,0)}$. The two tangents (which are unit vectors) are the same, if their dot product is 1, so we can find the optimal s_0 by maximizing the function

$$F(s_0) = \int_a^b \dot{\mathbf{x}}(s) \cdot \dot{\boldsymbol{\gamma}}_{s_0}(s)\,\mathrm{d}s,$$

over $s_0 \in [0, P(k)[$, where $P(k)$ is the period of the elastica with modulus k. It is easy to differentiate $\boldsymbol{\gamma}_{s_0}$ with respect to s_0, so we can find the first and second derivatives of F and use Newton's method to find the extremum points for F. We can then pick the one that gives the largest value.

Chapter 3

Curve approximation

In this chapter we present an algorithm for approximating an arbitrary planar curve segment by an elastic curve segment. The basic idea is to minimize some distance between the two curves with respect to the control parameters of the elastica. We first need to find a sensible distance function for planar curves, which we can input as the objective function in the optimization tool IPOPT. The optimization problem is non-convex and highly nonlinear, so it depends on a good initial guess. We shall describe a way of finding a canonical initial guess based on some geometric characteristics of the given curve.

Once we have a way of approximating arbitrary curves, we discuss additional constraints that we can include in the optimization to get better results. We also show how to approximate a curve by a G^1 piecewise elastic curve.

3.1 The objective function

Let $\mathbf{r}\colon [a,b] \to \mathbb{R}^2$ be a regular, C^2 curve in the plane and let $\boldsymbol{\gamma}$ be an elastic elastic curve segment. As our objective function we will use the squared L^2-distance $\|\boldsymbol{\gamma} - \mathbf{r}\|_2^2$, that is, the integral of the distances between the curves' points. For this to be a sensible distance function, we require that the two curves are parameterized in a compatible way. In Chapter 2 we chose always to parameterize our elastica with constant speed over the unit interval, but there is no guarantee that $\mathbf{r}$ has this parameterization, so we need to reparameterize one of the curves.

Let s denote the arc length function of $\mathbf{r}$, i.e.

$$s(t) = \int_a^t \|\mathbf{r}'(\tau)\| \,\mathrm{d}\tau, \quad t \in [a,b], \tag{3.1}$$

then the length of the curve is $L = s(b)$. Set $\varphi(t) = s(t)/L$, then $\varphi'(t) = \|\mathbf{r}'(t)\|/L$. In particular $\varphi' > 0$, so φ is a diffeomorphism $[a,b] \to [0,1]$.

Setting $\boldsymbol{\eta} = \mathbf{r} \circ \varphi^{-1}$, we have for all $t \in [0,1]$

$$\|\boldsymbol{\eta}'(t)\| = \|(\mathbf{r} \circ \varphi^{-1})'(t)\| = (\varphi^{-1})'(t)\|(\mathbf{r}' \circ \varphi^{-1})(t)\| = \frac{\|\mathbf{r}'\left(\varphi^{-1}(t)\right)\|}{\varphi'\left(\varphi^{-1}(t)\right)} = L,$$

so $\boldsymbol{\eta}$ is an (orientation preserving) reparameterization of $\mathbf{r}$ with constant speed L. Provided that the length of $\boldsymbol{\gamma}$ is close to L (which is true if the the elastica, approximates $\mathbf{r}$ well), $\boldsymbol{\eta}$ and $\boldsymbol{\gamma}$ are now parameterized in almost the same way.

We do not know the expression for $\boldsymbol{\eta}$ explicitly, but this is not important, because we have

$$\|\boldsymbol{\gamma} - \mathbf{r}\|_2^2 = \int_0^1 \|\boldsymbol{\gamma}(\tau) - \boldsymbol{\eta}(\tau)\|^2 \,\mathrm{d}\tau = \int_0^1 \|\boldsymbol{\gamma}(\tau) - (\mathbf{r} \circ \varphi^{-1})(\tau)\|^2 \,\mathrm{d}\tau = \int_a^b \|\boldsymbol{\gamma}(\varphi(t)) - \mathbf{r}(t)\|^2 \varphi'(t) \,\mathrm{d}t.$$

We therefore choose as our objective function

$$\mathscr{E}(\mathbf{p}) = \int_a^b \left\| \boldsymbol{\gamma}_{\mathbf{p}} \left(\frac{s(t)}{L} \right) - \mathbf{r}(t) \right\|^2 \frac{s'(t)}{L} \,\mathrm{d}t, \tag{3.2}$$

where $s(t)$ is given by (3.1). Then the objective function, $\mathscr{E}$, depends only on the control parameters $\mathbf{p} = (k, s_0, \ell, S, \phi, \hat{x}, \hat{y})$.

For the optimization, we have used the gradient driven tool IPOPT [WB05], so we need the first and second order partial derivatives of $\mathscr{E}$ with respect to the control parameters, which are

$$\begin{aligned} \frac{\partial \mathscr{E}}{\partial p_i} =& 2 \int_a^b \left(\boldsymbol{\gamma}_{\mathbf{p}} \left(\tfrac{s(t)}{L} \right) - \mathbf{r}(t) \right) \cdot \tfrac{\partial \boldsymbol{\gamma}_{\mathbf{p}}}{\partial p_i} \left(\tfrac{s(t)}{L} \right) \tfrac{s'(t)}{L} \,\mathrm{d}t, \\ \frac{\partial^2 \mathscr{E}}{\partial p_i \partial p_j} =& 2 \int_a^b \left(\tfrac{\partial \boldsymbol{\gamma}_{\mathbf{p}}}{\partial p_i} \left(\tfrac{s(t)}{L} \right) \cdot \tfrac{\partial \boldsymbol{\gamma}_{\mathbf{p}}}{\partial p_j} \left(\tfrac{s(t)}{L} \right) + \left(\boldsymbol{\gamma}_{\mathbf{p}} \left(\tfrac{s(t)}{L} \right) - \mathbf{r}(t) \right) \cdot \tfrac{\partial^2 \boldsymbol{\gamma}_{\mathbf{p}}}{\partial p_i \partial p_j} \left(\tfrac{s(t)}{L} \right) \right) \tfrac{s'(t)}{L} \,\mathrm{d}t. \end{aligned}$$

See Appendix A for a list of specific derivatives.

Implementation

For the implementation of this optimization problem, we have assumed that the given curve $\mathbf{r}$ is a spline. For the integrals we have used Gaussian quadrature, which is a way of approximating an integral by a weighted sum of specific integrand values. Gaussian quadrature with N points gives the exact value of the integral if the integrand is a polynomial of degree less than $2N - 1$.[1] Since a spline is a piecewise polynomial curve, it makes sense to do the integration on each knot interval and add up. Letting $a = \tau_0 < \ldots < \tau_M = b$ denote the distinct knots of $\mathbf{r}$, (3.2) becomes

$$\sum_{i=1}^{M} \left(\sum_{j=1}^{N} \left\| \boldsymbol{\gamma}_{\mathbf{p}} \left(\frac{s(t_{i,j})}{L} \right) - \mathbf{r}(t_{i,j}) \right\|^2 \frac{s'(t_{i,j})}{L} w_{i,j} \right),$$

where $t_{i,j}$ and $w_{i,j}$, $j = 1, \ldots, N$ denote the Gauss points and weights for the interval $[\tau_{i-1}, \tau_i]$, $i = 1, \ldots, M$.

It is easy to compute $s' = \|\mathbf{r}'\|$ in the Gauss points, but the arc length s is itself given by an integral. Fortunately the Gauss points do not depend on the optimization parameters, so we can, before starting optimization, once and for all calculate the values $s_{i,j} = s(t_{i,j})$. First we compute the arc length L_i of $\mathbf{r}$ for each knot interval

$$L_i = \sum_{j=1}^{N} \|\mathbf{r}'(t_{i,j})\| w_{i,j}, \quad i = 1, \ldots, M,$$

and we have the total curve length $L = \sum L_i$. Then, since $\tau_{i-1} < t_{i,j} < \tau_i$, we have

$$s_{i,j} = \sum_{n=1}^{i-1} L_n + \sum_{m=1}^{N} \|\mathbf{r}'(\hat{t}_m)\| \hat{w}_m,$$

where $\hat{t}_m$ and $\hat{w}_m$, $m = 1, \ldots, N$ are the Gauss points and weights for the interval $[\tau_{i-1}, t_{i,j}]$.

Remark. In the above, we assumed that the interval $[a, b]$ was the domain of the spline, but in some situations we may only want to approximate part of the curve. If for example $a \in [\tau_{\mu-1}, \tau_\mu[$ and $b \in]\tau_{\nu-1}, \tau_\nu]$, we will do as above, but only for the intervals $[a, \tau_\mu], [\tau_\mu, \tau_{\mu+1}], \ldots, [\tau_{\nu-2}, \tau_{\nu-1}]$, $[\tau_{\nu-1}, b]$.

[1] See [AS72] p. 887 for formulas and pp. 916–919 for Gauss points and weights.

3.2 The initial guess

The optimization problem defined in the previous section is non-convex and the result depends very much on the initial guess, see Figure 3.1. A canonical geometrically plausible guess is obtained from a generalization of the procedure of Section 2.4 to the case of an arbitrary input curve, which we will now describe. This description is identical to Section 4 of [BGN16] except for notation changes.

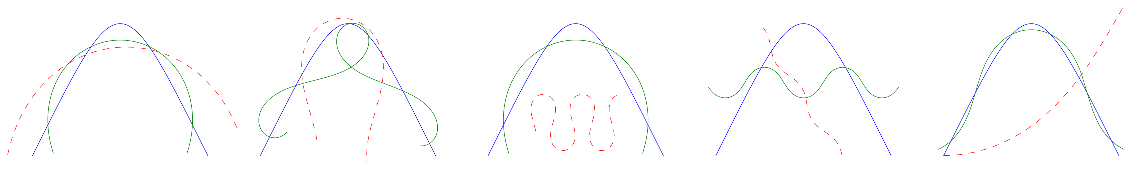


Figure 3.1: The blue curve is to be approximated by an elastica segment. The red dotted curves are different initial guesses for IPOPT optimization, and the green curves show the results. In the leftmost example the optimization terminated before an extremum was reached.

For the given curve, $\mathbf{r}\colon [a,b] \to \mathbb{R}^2$, we let (x,y) denote its coordinates and we can compute the curvature $\kappa(t) = \det(\mathbf{r}',\mathbf{r}'')/\|\mathbf{r}'\|^3$. We find $\lambda_1, \lambda_2, \alpha$ as before, by solving (2.17), noting that $\int_{s(a)}^{s(b)} f \,\mathrm{d}s = \int_a^b f(t)\,\frac{\mathrm{d}s}{\mathrm{d}t}\,\mathrm{d}t$. This gives us the scaling and rotation of the elastica (see (2.15)). We can judge the success by calculating the normalized residual

$$R_1 = \sqrt{\int_0^1 (\kappa(t) + \lambda_1\, y(t) - \lambda_2\, x(t) - \alpha)^2 \,\frac{\mathrm{d}s}{\mathrm{d}t}\,\mathrm{d}t} \Big/ \sqrt{\int_0^1 \kappa^2(t)\,\frac{\mathrm{d}s}{\mathrm{d}t}\,\mathrm{d}t}\,.$$

Similarly β can be found by (2.20), where $\sin\theta_u$ is given by (2.18), where $\mathbf{t}$ is the unit tangent for $\mathbf{r}$, and we can calculate the normalized residual

$$R_2 = \sqrt{\frac{1}{L}\int_0^1 \left(\sin\theta_u(t) - \tfrac{1}{2}\lambda\, u^2(t) - \alpha\, u(t) - \beta\right)^2 \frac{\mathrm{d}s}{\mathrm{d}t}\,\mathrm{d}t}\,.$$

Remark. Another possibility is to forget that we know λ and α and solve (2.19) with respect to λ, α, and β, but in the few cases we tried this, the results got worse.

We know that $\sin\theta_u$ takes values in $[-1,1]$, and since β is chosen to minimize the distance between $\sin\theta_u$ and the polynomial $P(u) = \frac{1}{2}\lambda\, u^2 + \alpha\, u + \beta$, the latter must be less than 1 for some u-values, so the number $\delta_- = \sqrt{\alpha^2 - 2\lambda(\beta - 1)}$ is well-defined. We can thus determine whether the elastica has inflection points and we can determine the parameter k from (2.23).

At this point we need to take into account the fact that the input curve is not necessarily an elastica. For an elastica, we could easily count the oscillations, but for an arbitrary curve there may be oscillations of different sizes. We find the curve segments where u is monotone, but we only count such a segment as an oscillation if it has some minimal height: we have used half of the difference $u_{\max} - u_{\min}$ (as defined in Section 2.4) as this minimum. Moreover, the right hand side of (2.24) need not be between -1 and 1, or, in the noninflectional case, between $\sqrt{1-1/k^2}$ and 1. We have circumvented this problem by replacing too small values by -1 (or $\sqrt{1-1/k^2}$) and too large values by 1. The two issues are illustrated in Figure 3.2.

We can thus find s_0 and ℓ. We can judge the validity by calculating

$$R_3 = \frac{1}{L}\int_{u(t)\notin[u_{\min},u_{\max}]} \frac{\mathrm{d}s}{\mathrm{d}t}\,\mathrm{d}t\,.$$

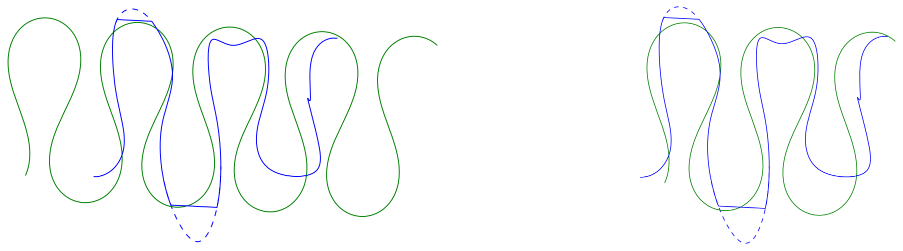


Figure 3.2: If the given curve moves outside the interval $[u_{\min}, u_{\max}]$ (dotted segment), it is simply cut off in these regions. The resulting elastica is shown in green. On the left all oscillations of the input curve are counted, on the right the two very small ones are ignored.

We finally determine the translation by (2.25) computed as

$$\begin{pmatrix}\hat{x}\\ \hat{y}\end{pmatrix} = \frac{1}{L}\int_{s(a)}^{s(b)} (\mathbf{r}(s) - \boldsymbol{\gamma}_0(s))\,\mathrm{d}s = \frac{1}{L}\int_a^b \left(\mathbf{r}(t) - \boldsymbol{\gamma}_0\left(\tfrac{s(t)}{L}\right)\right)\|\mathbf{r}'(t)\|\,\mathrm{d}t,$$

since we need the two curves to be parameterized in the same way when integrating over t. We define the residual as

$$R_4(\mathbf{p}_0) = \sqrt{\frac{\mathcal{E}(\mathbf{p}_0)}{L^2}}\,,$$

where $\mathbf{p}_0$ is the vector of control parameters found by the above procedure.

Remark. As above, all integrals are computed by Gaussian quadrature on each knot interval

$$\int_{s(a)}^{s(b)} f\,\mathrm{d}s = \int_a^b f(t)\frac{\mathrm{d}s}{\mathrm{d}t}\,\mathrm{d}t = \sum_{i=1}^{M}\int_{\tau_{i-1}}^{\tau_i} f(t)\|\mathbf{r}'(t)\|\,\mathrm{d}t \approx \sum_{i=1}^{M}\sum_{j=1}^{N} f(t_{i,j})\|\mathbf{r}'(t_{i,j})\|w_{i,j},$$

so what we need are the value, unit tangent, curvature and arc length of $\mathbf{r}$ sampled in the Gauss points. In the steps where we find the monotone segments and use these to find s_0 and ℓ, we add the end points to the set of sample points, since we are interested in the shape of the entire curve (and the first and last Gauss points are never the limits of the domain).

Remark. As described in Section 2.5, the method for finding s_0 and ℓ is not numerically stable, even less so here, where we need to find and count the segments where u is monotone. Instead we can again just set $\ell = L/S$, thus ensuring that the elastica has the same length as the given curve. The parameter s_0 can then be found by maximizing

$$F(s_0) = \int_{s(a)}^{s(b)} \dot{\mathbf{r}}(s)\cdot\dot{\boldsymbol{\gamma}}_{s_0}(s)\,\mathrm{d}s = \int_a^b \dot{\mathbf{r}}(t)\cdot\dot{\boldsymbol{\gamma}}_{s_0}\left(\tfrac{s(t)}{L}\right)\|\mathbf{r}'(t)\|\,\mathrm{d}t, \tag{3.3}$$

over a period of the elastica with modulus k. The success can be validated by computing the residual

$$R_3^* = 1 - \frac{F(s_0)}{L}$$

When ℓ and s_0 determined, we find the translation $(\hat{x}, \hat{y})$ as described above.

Remark. Instead of using the L^2-distance as the objective function, one could use the H^1-distance or H^2-distance. This would take into account the tangents and, if we choose the H^2-distance, the curvatures of the curves, and would help prevent results like the second and fourth examples in Figure 3.1. Since our method for finding the initial guess is based on curvature, the H^2-distance may be the natural choice, but it is more complicated, and in our experience the L^2-distance gives good results provided that we have a good initial guess.

Results

We have tested the procedure on a selection of cubic Bézier curves, displayed in Figures 3.3 and 3.4. The residuals of the method for finding the initial guess are reported in Table 3.1, and the results from the minimization of $\mathscr{E}$ from (3.2) can be found in Table 3.2.

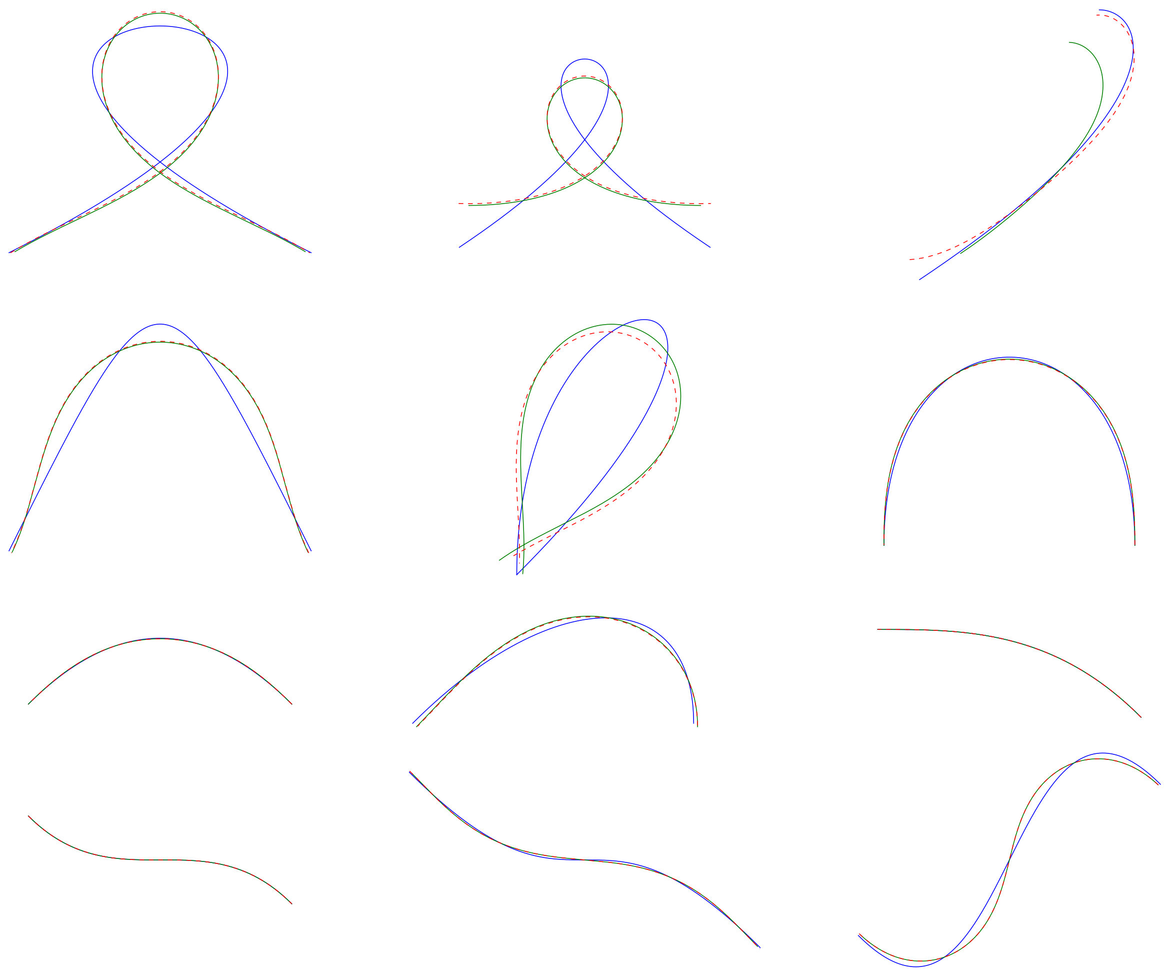


Figure 3.3: Examples of cubic Bézier curves (blue) and the canonical initial guess. The solid green curve is the initial guess where s_0 is found by counting segments where u is monotone. The dashed red curve is the initial guess where s_0 is found by maximizing (3.3).

	R_1	R_2	R_3	R_4	R_3^*	R_4^*
1	4.56e-01	1.39e-01	0.00e+00	9.71e-03	1.01e-02	8.65e-03
2	4.48e-01	2.29e-01	5.24e-01	4.80e-02	7.42e-02	4.38e-02
3	1.36e-01	2.05e-02	1.90e-01	7.68e-02	9.58e-03	1.58e-02
4	6.78e-01	1.74e-01	1.43e-01	2.21e-02	2.34e-02	2.23e-02
5	6.52e-01	2.71e-01	1.43e-01	3.12e-02	4.25e-02	3.40e-02
6	9.92e-02	2.46e-02	4.76e-02	3.62e-03	4.41e-04	3.61e-03
7	6.39e-02	4.42e-03	4.76e-02	1.03e-03	4.36e-05	1.01e-03
8	2.69e-01	6.92e-02	1.43e-01	1.14e-02	3.56e-03	1.10e-02
9	1.72e-02	3.34e-04	0.00e+00	1.34e-04	7.96e-07	1.31e-04
10	2.00e-02	6.22e-04	0.00e+00	1.16e-04	1.84e-06	1.16e-04
11	3.30e-01	1.39e-02	0.00e+00	4.96e-03	1.09e-03	3.28e-03
12	3.58e-01	9.97e-02	1.90e-01	1.52e-02	9.20e-03	1.43e-02

Table 3.1: The first column refers to the examples in Figure 3.3, the next four report the residuals R_1, R_2, R_3 and R_4 in the approximation process. Next, we report the residual R_3^* and the new value of R_4 obtained by using the improved algorithm. It makes no sense to compare R_3 and R_3^*, but a comparison of R_4 and R_4^* shows that the improved algorithm is indeed for most cases an improvement.

	$R_4(\mathbf{p}_{\text{opt}})$	$\|\nabla\mathscr{E}(\mathbf{p}_{\text{opt}})\|$	$\sharp$ iter	$R_4(\mathbf{p}_{\text{opt}})$	$\|\nabla\mathscr{E}(\mathbf{p}_{\text{opt}})\|$	$\sharp$ iter
1	7.98e-03	1.55e-10	12	7.98e-03	7.14e-12	9
2	3.81e-02	6.14e-02	1000†	3.81e-02	1.30e+00	1000†
3	1.80e-03	1.95e-12	26	1.80e-03	1.14e-12	12
4	1.20e-02	7.64e-10	11	1.20e-02	3.45e-11	21
5	1.76e-02	7.51e-10	10	1.76e-02	3.59e-10	9
6	1.15e-03	5.09e-12	24	1.15e-03	5.83e-10	17
7	3.15e-04	2.74e-12	20	3.15e-04	2.05e-13	16
8	3.24e-03	1.00e-12	204	3.24e-03	2.32e-11	127
9	1.23e-03	2.95e-04	1000†	3.96e-05	6.00e-11	75
10	9.92e-05	7.26e-12	292	9.92e-05	4.83e-13	243
11	1.67e-03	8.56e-11	40	1.67e-03	2.20e-12	59
12	4.10e-03	5.49e-11	185	4.10e-03	4.91e-11	122

Table 3.2: The first column refers to the examples in Figure 3.4, the next three report the normalized L^2-distance, the gradient norm $\|\nabla\mathscr{E}\|$ and the number of iterations for the optimized elastica, all based on the "unstable" initial guess. Next, we report the same three numbers for the "stable" initial guess. In case 9, we have an improvement in the L^2-distance, but in Figure 3.4 it cannot be perceived.
†IPOPT terminated because iteration count reached maximum (which was set to 1000).

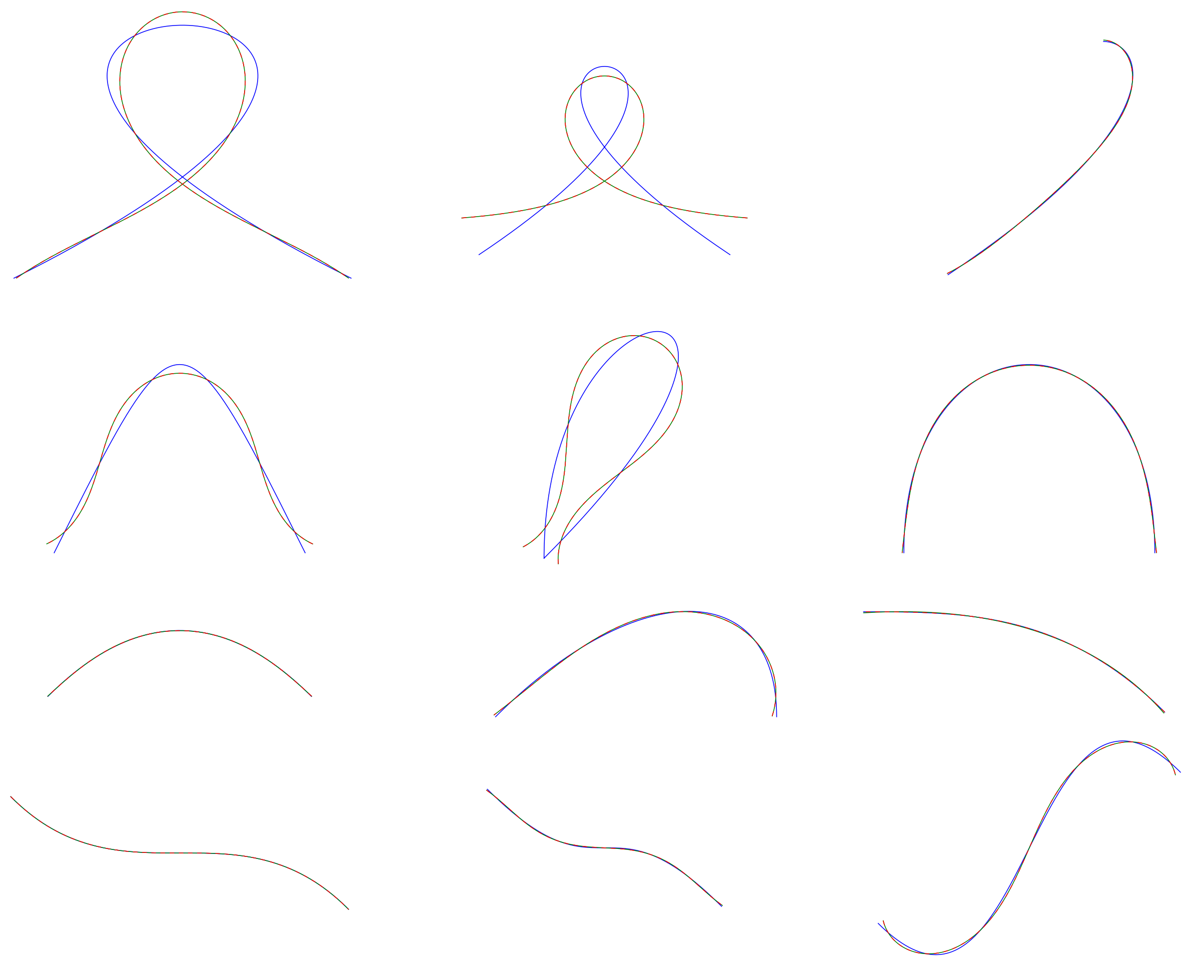


Figure 3.4: Examples of cubic Bézier curves (blue) and the optimized elastica approximation. The green and red curves are the solutions based on the green and red initial guesses in Figure 3.3.

3.3 Additional optimization constraints

If we want the approximating elastic curve to satisfy further conditions, we can introduce these as constraints in the optimization problem. We mention here some constraints that may be relevant in different settings.

Firstly, we could demand that the length of the elastica is close to the length L of the original curve. In terms of the control parameters, this is a very simple constraint, namely that $|L - |\ell|S| < \varepsilon$, where ε is some tolerance. In most cases this constraint is superfluous, because it will necessarily be satisfied by the best approximation. However, it may help during optimization to avoid moving in a bad direction within the parameter space. For example, without this constraint it can happen that the optimization converges to a point where ℓ or S is zero.

Remark. One may note that the expression $L - |\ell|S$ is not differentiable with respect to ℓ. In our implementation this does not pose a problem because it is determined whether ℓ is positive or negative during the step of finding the initial guess. During optimization ℓ is then only allowed to move through either the positive or the negative numbers, and we do not need the absolute value in the constraint.

We may want the solution curve to start and end at the same points as the original curve. This can be done by introducing the constraints

$$\boldsymbol{\gamma}(0) = \mathbf{r}(a), \quad \boldsymbol{\gamma}(1) = \mathbf{r}(b),$$

which is actually four constraints in the implementation, since each point has two coordinates. This constraint is relevant, for example, if we have several connected curves (in the sense that one curve starts where another one ends) that we want to approximate by elastic curves, and we want the elastic curves to be connected as well.

For the same reason, we may want the elastica to have the same end tangent directions as the original curve. The obvious way to implement this would be to make sure that the unit tangents are the same. The problem with this is that, if we want both endpoints and tangents to fit, this adds up to eight constraints; but we only have seven parameters to vary. IPOPT has no way of knowing that the unit tangents each correspond only to one degree of freedom, so it will simply state that there are not enough degrees of freedom. Instead we can look at the angles. We can easily determine the tangential angles α_0 and α_1 at the end points of the original curve, from its tangents, and we can choose the angles to lie in $]-\pi, \pi]$. In fact, most programming languages have a function for this purpose, often named atan2. For the elastica, we saw in Chapter 2 that the expression for the angle function depends on whether the curve has inflections. Moreover if the orientation is reversed, that is, if $\ell < 0$, the angle changes by π. The general angle function for $\boldsymbol{\gamma}_{(k,s_0,\ell,S,\phi,\hat{x},\hat{y})}$ is

$$\theta_{(k,s_0,\ell,\phi)}(t) = \begin{cases} \phi + 2\arcsin\left(k\,\mathrm{sn}(s_0 + \ell t)\right) + \pi \mathbf{1}_{\mathbb{R}_-}(\ell) & \text{if } k \le 1, \\ \phi + 2\,\mathrm{am}\left(k(s_0 + \ell t), \frac{1}{k}\right) + \pi \mathbf{1}_{\mathbb{R}_-}(\ell) & \text{if } k > 1. \end{cases} \tag{3.4}$$

With this, we can introduce the constraints

$$\theta(0) \equiv \alpha_0 \pmod{2\pi}, \quad \theta(1) \equiv \alpha_1 \pmod{2\pi}.$$

It is relevant to note that, even though the formula for θ changes depending on k and ℓ, the expressions for its partial derivatives do not.

The elastic curves represent the possible shapes of the blade described in Section 1.1. However, not all elastic curves can be realized by the blade. Depending on the material, thickness and

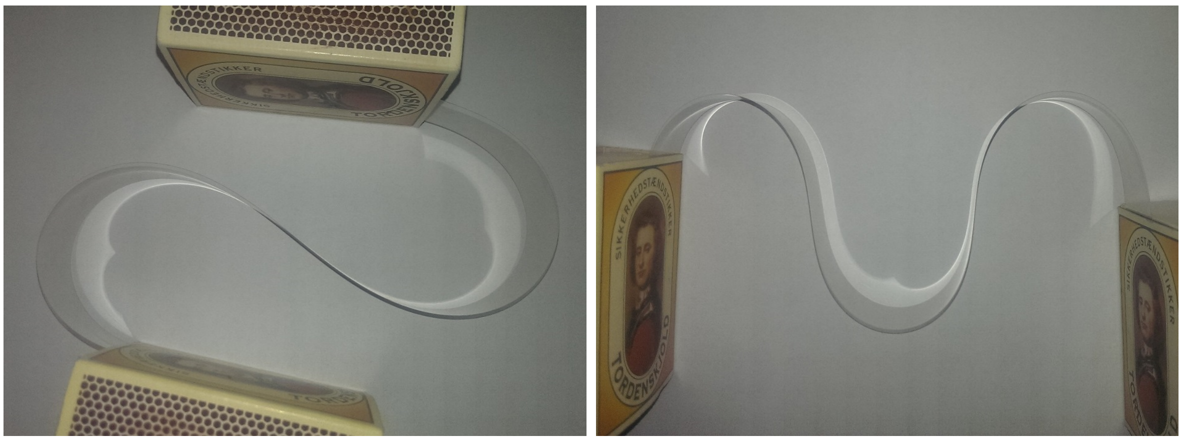


Figure 3.5: Thin transparent plastic strip fixed between two matchboxes.

cross section of the blade, there will be an upper limit to the amount of bending it can handle without being permanently deformed. The stress is proportional to the curvature so, to avoid too heavy bending, we can impose bounds on the curvature. While the maximal curvature for an unbounded elastica is $2k/S$, we may consider a segment that does not achieve this maximum, which makes it a bit more complicated to introduce a constraint. However, if the optimization results in a too curved elastica, it is most likely because the original curve is too curved. In this case introducing a curvature constraint will worsen the optimization result so it would be better to fix this in the design phase so that the original curve does not demand higher curvature than the blade can provide.

While some very flexible materials can produce quite complex elastic curves, see Figure 3.5, our cutting tool is metallic and thus pretty stiff. For this reason we cannot expect to obtain curves with more than two inflection points. We can impose this simply by constraining the parameter ℓ to be less than a period (and remember that the length of a period depends on k), since there are exactly two inflections per period. For the elastica without inflections, a constraint of this sort may also be useful, since we do not want self-intersections.

3.4 Approximation with several elastic segments

As we have seen, not all curves can be well approximated by an elastic curve. For example, if a curve has oscillations of different amplitudes, it will not resemble any elastic curve. A way to deal with this is to subdivide the curve into pieces and then approximate each piece.

Suppose that we are given a curve $\mathbf{r}\colon [a,b] \to \mathbb{R}^2$ and parameter values $a = t_0 < \ldots < t_N = b$. We then find N elastic curve segments such that $\boldsymbol{\gamma}_{\mathbf{p}_i}$ approximates $\mathbf{r}$ restricted to the interval $[t_{i-1}, t_i]$, for $i = 1, \ldots, N$, by minimizing the function

$$\hat{\mathscr{E}}(\mathbf{p}_1, \ldots, \mathbf{p}_N) = \sum_{i=1}^{N} \int_{t_{i-1}}^{t_i} \left\| \boldsymbol{\gamma}_{\mathbf{p}_i}\left(\frac{s_i(t)}{L_i}\right) - \mathbf{r}(t) \right\|^2 \frac{s_i'(t)}{L_i}\, \mathrm{d}t,$$

where

$$s_i(t) = \int_{t_{i-1}}^{t} \|\mathbf{r}'(\tau)\|\, \mathrm{d}\tau, \quad L_i = s_i(t_i), \qquad \text{for } i = 1, \ldots, N.$$

We observe that the i'th term in $\hat{\mathscr{E}}$ depends only on $\mathbf{p}_i$, so each segment is approximated independently. This means that the result will be N elastic curves with no connection whatsoever, but this is not what we want. We want a piecewise elastic curve that approximates $\mathbf{r}$ and is at least tangent continuous. We can of course obtain this by imposing constraints as described in the previous section, such that the endpoints of each of the elastic curves fit the original curve exactly. This, however, would let the parameters $t_1, \ldots, t_N$ dominate too much, see Figure 3.6. Instead we impose constraints that ensure geometric continuity of the piecewise elastic curve, but are not related to the original curve.

For continuity we want $\boldsymbol{\gamma}_{\mathbf{p}_i}(1) = \boldsymbol{\gamma}_{\mathbf{p}_{i+1}}(0)$, but for production, we do not need to be that exact, so we will introduce a tolerance $\varepsilon_0 \geq 0$. We impose the constraints

$$-\varepsilon_0 \leq \gamma^j_{\mathbf{p}_i}(1) - \gamma^j_{\mathbf{p}_{i+1}}(0) \leq \varepsilon_0, \;\; i = 1, \ldots, N-1, \;\; j = 1, 2,$$

where $j = 1, 2$ correspond to the coordinates. Production accuracy is usually in the order of millimetres, so if e.g. we set $\varepsilon_0 = 0.1\,\text{mm}$, we should be more than safe.

For tangent continuity (G^1-continuity), we need the unit tangents of adjacent curves to be equal. Again for production, we can be less exact; in fact in many cases an angle change of 1 degree cannot be visually discerned. We introduce the constraint in the form

$$\frac{\boldsymbol{\gamma}'_{\mathbf{p}_i}(1)}{\|\boldsymbol{\gamma}'_{\mathbf{p}_i}(1)\|} \cdot \frac{\boldsymbol{\gamma}'_{\mathbf{p}_{i+1}}(0)}{\|\boldsymbol{\gamma}'_{\mathbf{p}_{i+1}}(0)\|} \geq \cos\left(\frac{\pi}{180}\varepsilon_1\right), \;\; i = 1, \ldots, N-1,$$

where $\varepsilon_1 \geq 0$ is the tolerated angle difference in degrees. For implementation, since IPOPT needs the constraint bounds to be independent of the optimization parameters, we rewrite the condition as

$$\boldsymbol{\gamma}'_{\mathbf{p}_i}(1) \cdot \boldsymbol{\gamma}'_{\mathbf{p}_{i+1}}(0) - \|\boldsymbol{\gamma}'_{\mathbf{p}_i}(1)\| \, \|\boldsymbol{\gamma}'_{\mathbf{p}_{i+1}}(0)\| \cos\left(\frac{\pi}{180}\varepsilon_1\right) \geq 0, \;\; i = 1, \ldots, N-1.$$

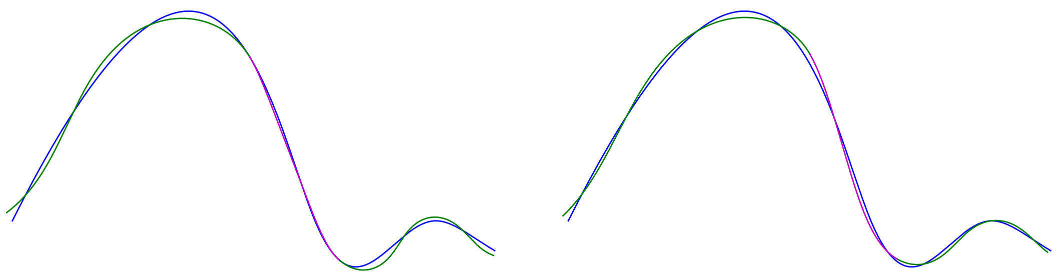


Figure 3.6: Spline curve approximated by a piecewise elastic curve. Left: The elastic segments' endpoints and tangents are fit to the original curve. Right: The optimization is constrained to tangent continuity.

We can of course impose further constraints in order to obtain a higher degree of continuity. In Chapter 7, we will construct surfaces by sweeping piecewise elastic curves obtained in the way described above. From a mathematical perspective it may be relevant to ensure curvature continuity (G^2-continuity) in order to get smooth reflection lines on the resulting surfaces. For production, however, this would require a very high precision, and we would of course have to set the tolerance ε_1 to zero.

When we approximate a curve by several elastic curve segments, the result depends on where we switch from one elastic segment to the next, see Figure 3.7. We will not consider how to choose the subdivision here, since the goal is to use this for surface approximation, and for surfaces there may be other considerations that must be taken into account when choosing where to subdivide.

Remark. We considered letting the parameters $t_0, \ldots, t_N \in [a, b]$ in which we split the given curve $\mathbf{r}$ be part of the optimization. However, since we calculate integrals by Gaussian curvature, we

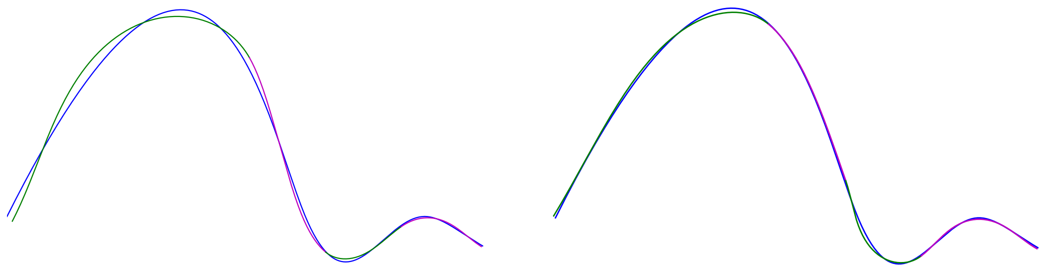


Figure 3.7: Spline curve approximated by four elastic curve segments with tangent continuity. Left: The split points are chosen by uniform subdivision of the spline curve's domain. Right: The split points are tweaked manually, to give a better result.

need to compute the value, tangent and arc length of $\mathbf{r}$ in specific points for each interval $[t_{i-1}, t_i]$. If these intervals can change, we have to recalculate all these in each step of the optimization, which is expensive. When we keep the parameters fixed, all computations on $\mathbf{r}$ can be done once and for all before starting optimization.

Chapter 4

The boundary value problem

In Chapter 2 we chose a parameterization of the space of elastic curve segments. The main advantage of this parameterization is that when the seven control parameters are known, the entire elastic curve is known accurately and all subsequent calculations are easy to perform. A more natural way to describe an elastic curve segment, from a geometric point of view, is by its length, endpoints, and the tangent angles at the endpoints (which are exactly the data we need to give the robots for hot blade cutting). The problem is that we do not have a closed form description in these parameters. Furthermore, the length, endpoints and tangents do not uniquely define an elastic curve segment; if, for example, we rotate one of the end tangents by 2π, moving through a continuous family of elastic curves, we will get a different curve segment with the same boundary conditions. (Such a tangent rotation is not always possible with a physical elastic rod, since the resulting elastica can have a self-intersection.) Even if we allow angles with arbitrary real values, to ensure a continuous angle function, we still do not have uniqueness, see Figure 4.1.

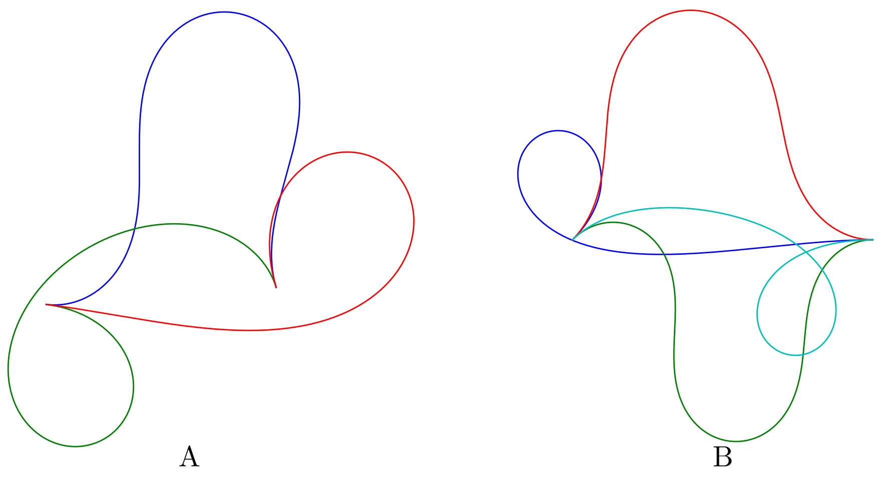


Figure 4.1: Examples of different elastic curves with the same length, endpoints, and end tangent directions. A: The blue curve can be continuously transformed into the green (or the red) curve by rotating the left (or the right) end tangent by 2π. B: If we choose continuous angle functions for the curves, going from the left point to the right, and starting with the same value, the values at the end will differ by 4π for the blue and the cyan curves. The end angle for the green and red curves, however, will be the same.

In this chapter we wish to solve the *boundary value problem* for the elastica, that is, we want to find the control parameters for an elastic curve segment with given length, end points and end tangents. As already mentioned, a unique solution does not exist, so we will describe a way to obtain a specific solution, the shape of which is guided by a particular input.

4.1 Discrete elastica

Before we try to find the control parameters, thus getting an exact elastica, we will first find a numerical solution to get a "sketch" of the curve that we are searching for. We do this by finding a *discrete elastica*, by which we mean a polyline that satisfies the boundary conditions and minimizes the bending energy in a discrete sense (see [BHN01]).

Let $\mathbf{x}_0, \mathbf{x}_1 \in \mathbb{R}^2$, $\alpha_0, \alpha_1 \in \mathbb{R}$ be the desired endpoints and end tangent angles of the elastic curve segment and let L be the desired length. We want to find a curve from $\mathbf{x}_0$ to $\mathbf{x}_1$ consisting of $N+1$ line segments of length $L/(N+1)$ and with the prescribed tangent angles at the endpoints, see Figure 4.2. For the bending energy $\int \kappa^2 \,\mathrm{d}s$, in this discrete setting, we use the sum of the squared turning angles $\theta_1, \ldots, \theta_N$, that is

$$\mathscr{E}_{\text{discrete}}(\theta_1, \ldots, \theta_N) = \sum_{i=1}^{N} \theta_i^2.$$

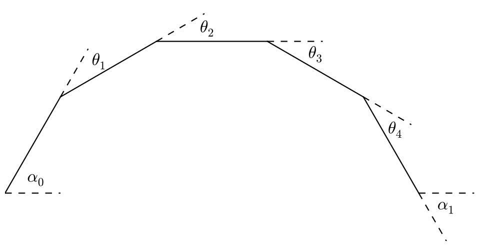


Figure 4.2: Polyline with prescribed boundary conditions. The turning angles $\theta_1, \ldots, \theta_N$ are shown, with $N = 4$.

To find the discrete elastica, we minimize $\mathscr{E}_{\text{discrete}}$ subject to the two constraints:

$$\alpha_0 + \sum_{i=1}^{N} \theta_i = \alpha_1, \qquad \mathbf{x}_0 + \frac{L}{N+1} \sum_{i=0}^{N} \begin{pmatrix} \cos\left(\alpha_0 + \sum_{j=1}^{i} \theta_j\right) \\ \sin\left(\alpha_0 + \sum_{j=1}^{i} \theta_j\right) \end{pmatrix} = \mathbf{x}_1.$$

The first constraint makes sure that the final angle is correct, and since we have not restricted α_0 and α_1 to an interval of length 2π, we are also certain that the total angle change is correct. The second constraint ensures that the curve beginning at $\mathbf{x}_0$ with angle α_0 after the N angle turns, will end at $\mathbf{x}_1$. In the implementation this is two constraints, one for each coordinate. In [BNR01] Bruckstein et al. show that if the solution curve to the discrete problem converges as $N \to \infty$, then the limit is an elastic curve.

It is easy to compute the first and second partial derivatives of the objective and constraint functions, and the problem has been implemented in IPOPT with good results. The optimization's performance and result depend on the initial guess for the turning angles. For example, if we set $\mathbf{x}_0 = (0, 0)$, $\mathbf{x}_1 = (1, 0)$, $\alpha_0 = \alpha_1 = 0$, $L = 1.5$ and let $\theta_i = 0$, $i = 1, \ldots, N$ be our initial guess, IPOPT cannot solve the problem[1]. This is no surprise since this corresponds to pressing the ends of a straight elastic rod together. At some point a buckling will take place and the rod will bend, but there is no way to know in which direction. However, if we guide the optimization in a certain direction, we can solve the problem. Setting $\theta_1 = 0.1$ in the initial guess, IPOPT finds the upwards bending curve seen in Figure 4.3.[2]

One way to choose an initial guess, is to extract it from a "sketch" curve that resembles the

[1]IPOPT terminates with the message: "Converged to a point of local infeasibility. Problem may be infeasible."

[2]For $N = 50$, IPOPT finds the downwards bending curve. This can of course be helped by providing a better initial guess.

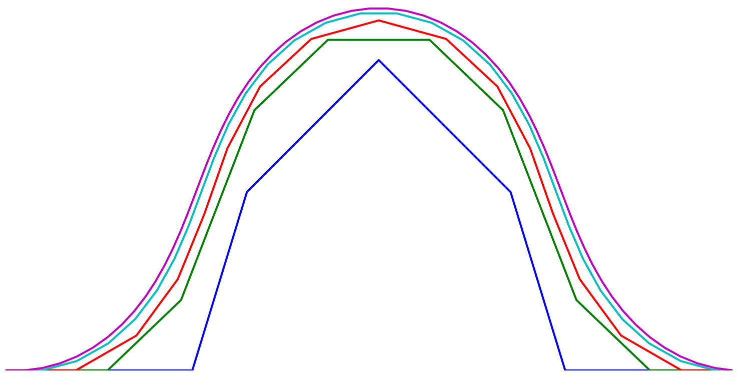


Figure 4.3: Discrete elastica with prescribed boundary conditions for $N = 5, 10, 15, 30, 60$.

elastic curve we are searching for. It is easy to construct a spline curve that meets the boundary conditions except for the length condition; we simply choose $\mathbf{x}_0$ and $\mathbf{x}_1$ as the first and last control points and let the second and the second-to-last control points lie on the half lines $\mathbf{x}_0 + s\mathbf{t}_0$ and $\mathbf{x}_1 - s\mathbf{t}_1$, respectively, with $s > 0$ and $\mathbf{t}_i = (\cos\alpha_i, \sin\alpha_i)$, $i = 0, 1$. From the spline $\mathbf{r}\colon [0,1] \to \mathbb{R}^2$, we can then compute the tangent angles at $t_i = i/(N+1)$, $i = 1, \ldots, N$ and use the angle differences as the initial guess for the turning angles in the optimization. For example, the three discrete elastica in Figure 4.1A, can be found in this way by using the splines in Figure 4.4 as sketch curves.

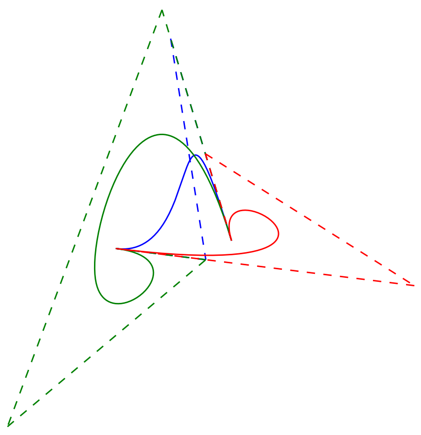


Figure 4.4: Three cubic spline curves and their control polygons. The blue and red curves are actually Bézier curves. Using the turning angles, endpoints and end angles from these curves, we find the elastica of Figure 4.1A after IPOPT optimization.

Since we are mainly interested in curves without self-intersections and with at most two inflection points, in many cases we can simply use a cubic Bézier curve with control points $\mathbf{x}_0$, $\mathbf{x}_0 + s_0\mathbf{t}_0$, $\mathbf{x}_1 - s_1\mathbf{t}_1$, and $\mathbf{x}_1$, for some $s_0, s_1 > 0$ as sketch curve.[3] However, this will not work if $\mathbf{t}_0$ and $\mathbf{t}_1$ are parallel, as in the case described above. Another question that comes up is how to choose the numbers s_0 and s_1. One possibility is choosing $s_0 = s_1$ such that the Bézier curve has exactly length L, but this is not always a good strategy, if we want to avoid self-intersection. In any case, the sketch curve should be chosen such that the total angle difference is correct, that is, the integral of the curvature must be equal to $\alpha_1 - \alpha_0$.

[3]A cubic Bézier curve has at most one self-intersection and at most two inflection points.

4.2 Analytical expression

We wish to find the control parameters for an elastic curve with given endpoints (x_0, y_0), (x_1, y_1), end angles α_0, α_1 and length L. In other words, we want a vector $\mathbf{p} = (k, s_0, \ell, S, \phi, \hat{x}, \hat{y})$ such that

$$\boldsymbol{\gamma}_{\mathbf{p}}(0) = \begin{pmatrix} x_0 \\ y_0 \end{pmatrix}, \quad \boldsymbol{\gamma}_{\mathbf{p}}(1) = \begin{pmatrix} x_1 \\ y_1 \end{pmatrix},$$
$$\theta_{(k,s_0,\ell,\phi)}(0) = \alpha_0, \quad \theta_{(k,s_0,\ell,\phi)}(1) = \alpha_1, \quad |\ell|S = L.$$

It turns out that we can eliminate all but three of the control parameters, the first being S which we set to $L/|\ell|$.

From the first angle condition, we can, recalling the angle function (3.4), express the rotation parameter as

$$\phi(k, s_0, \ell) = \begin{cases} \alpha_0 - 2 \arcsin\left(k \operatorname{sn}(s_0)\right) - \pi \mathbf{1}_{\mathbb{R}_-}(\ell) & \text{if } k \leq 1, \\ \alpha_0 - 2 \operatorname{am}\left(k(s_0), \frac{1}{k}\right) - \pi \mathbf{1}_{\mathbb{R}_-}(\ell) & \text{if } k > 1. \end{cases}$$

Moreover, we observe that $\Delta\boldsymbol{\gamma} = \boldsymbol{\gamma}(1) - \boldsymbol{\gamma}(0)$ does not depend on the translation $(\hat{x}, \hat{y})$ and that $\Delta\theta$ depends neither on ϕ nor on the possible π term.

To find the control parameters for the elastic curve we minimize the function

$$\mathscr{F}(k, s_0, \ell) = \left(\Delta\gamma^1_{(k,s_0,\ell)} - (x_1 - x_0)\right)^2 + \left(\Delta\gamma^2_{(k,s_0,\ell)} - (y_1 - y_0)\right)^2 + \left(\Delta\theta_{(k,s_0,\ell)} - (\alpha 1 - \alpha 0)\right)^2.$$

Again we use IPOPT for the optimization, and we refer to Appendix A for derivatives. When a minimizing set of parameters (k, s_0, ℓ) have been found, we use the vector

$$\mathbf{p} = (k, s_0, \ell, S/|\ell|, \phi(k, s_0, \ell), \hat{x}, \hat{y})$$

as control parameters for the elastica, where the translation $(\hat{x}, \hat{y})$ is computed from the condition that the curve starts at (x_0, y_0).

Since $\hat{x}$, $\hat{y}$ and ϕ are computed from x_0, x_1, and α_0, the resulting elastic curve will satisfy the conditions for the first point and angle exactly, while the final point and angle are only correct if $\mathscr{F} = 0$. For production purposes, however, we easily come close enough.

As before the optimization depends on a good initial guess. To obtain this, we first find a discrete elastica, that solves the boundary conditions, as described in the previous section (keeping in mind that different initializations will produce different results). Then, to get the control parameters, we will use the procedure described in Section 3.2. In that section, we restricted ourselves to C^2 curves, because we needed to compute curvature, but now we have a polyline. One way of handling this issue, is to approximate the discrete elastica by a spline curve and use this as input. However, since the algorithm only needs a discrete set of sample points, we can simply take the vertices of the discrete elastica. Since we know the turning angle at the i'th vertex, we immediately have an estimate for the curvature, namely $\kappa_i = \theta_i(N+1)/L$. The arc length at the i'th vertex is $s_i = iL/(N+1)$, and the unit tangent is given by

$$\mathbf{t}_i = \begin{pmatrix} \cos\left(\alpha_0 + \sum_1^{i-1} \theta_j + \frac{\theta_i}{2}\right) \\ \sin\left(\alpha_0 + \sum_1^{i-1} \theta_j + \frac{\theta_i}{2}\right) \end{pmatrix}.$$

The weight to assign each vertex (for quadrature) is $w_i = L/(N+1)$.

If the discrete elastica is close to the desired curve, the initial guess for the minimization of $\mathscr{F}$ should be good, so this optimization should not be able to jump to another solution to the given boundary values. In any case, we can still impose extra constraints (e.g. that ℓ is within a certain number of periods) to ensure that we get the desired solution.

4.3 Designing with elastic curves

The main objective of the BladeRunner project is to rationalize an arbitrary design and obtain a surface that can be cut by the hot blade technology. This will be the topic of Chapters 5–7. In this section we will see how the algorithms from this chapter can be used for creating pre-rationalized surfaces, i.e. surfaces that are, by design, ready for hot blade cutting.[4] The methods described in this section were implemented as a tool set in Rhino and tested at a workshop at RobArch 2016.[5] By following a specified work-flow, the workshop participants could design surfaces in Rhino and then directly export robot code for hot blade fabrication, which was done on site. The work done in preparation for the workshop was a collaboration between the BladeRunner partners at Odico, 3XN, DTU Compute and DTI, and the results are described in [BBC+16].

Single block designs

We will design our surfaces with the EPS blocks and cutting tool in mind. Let us for example consider a block with dimensions 600 mm × 800 mm × 500 mm and a blade of length 800 mm. We will orient our coordinate system according to the block, so that we cut in the y-direction, and the width is 600 mm, which is fitting for a 800 mm blade. We design a surface that intersects the block and represents the cut we want to perform. Since the block has to be supported, we do not allow the surface to intersect the bottom face of the block.

We want to design our surface so that it is close to an elastica-foliated surface that can be cut by the hot blade. This means that in the blade direction, i.e. the direction perpendicular to the cutting direction, the curvature should not be too high and there should be at most two inflection points. One can, for example, use tensor product spline surfaces with few control points in the blade direction. The test surfaces in this section are spline surfaces with 4 control points in the blade direction. Moreover, the control points are positioned in an (x, y)-grid with randomized z-values.

We sample a set of planar curves on the surface by taking the intersection with a set of planes along the cutting direction (or, in the case of a NURBS surface, we can design the surface such that its isocurves are planar), see Figure 4.5. For simplicity we use a set of parallel planes. These planar curves, which we will refer to as *sweep curves*, since they sweep out the surface, will serve as models for the blade shape. There are two issues we must address: the sweep curves do not have the same length as the blade, and they are not likely to be elastic curves. We therefore first extend the sweep curves (by the same amount at each end) to the blade length of 800 mm. We then take the endpoints and end tangents of the extended curves and find discrete elastic curves with the same boundary conditions using the method described in Section 4.1.[6] The sweep curves' planes are parallel to the (x, z)-plane, so we just need the x- and z-coordinates of the endpoints and tangents. The resulting elastic curves represent the blade shapes that will be

[4]A research project concerning design with elastic curves was initiated in 2015 by Odico, 3XN and DTU Compute with support from Innovation Fund Denmark.

[5]`http://www.robarch2016.org/workshops/` *Superform: Robotic Hot-Blade Cutting* workshop.

[6]For the RobArch-workshop, a Rhino-plugin created by J. Andreas Bærentzen and Kasper Steenstrup was used. It solves the minimization problem stated in Section 4.1 using an optimization algorithm written by J. Andreas Bærentzen. Details can be found in [BBC+16].

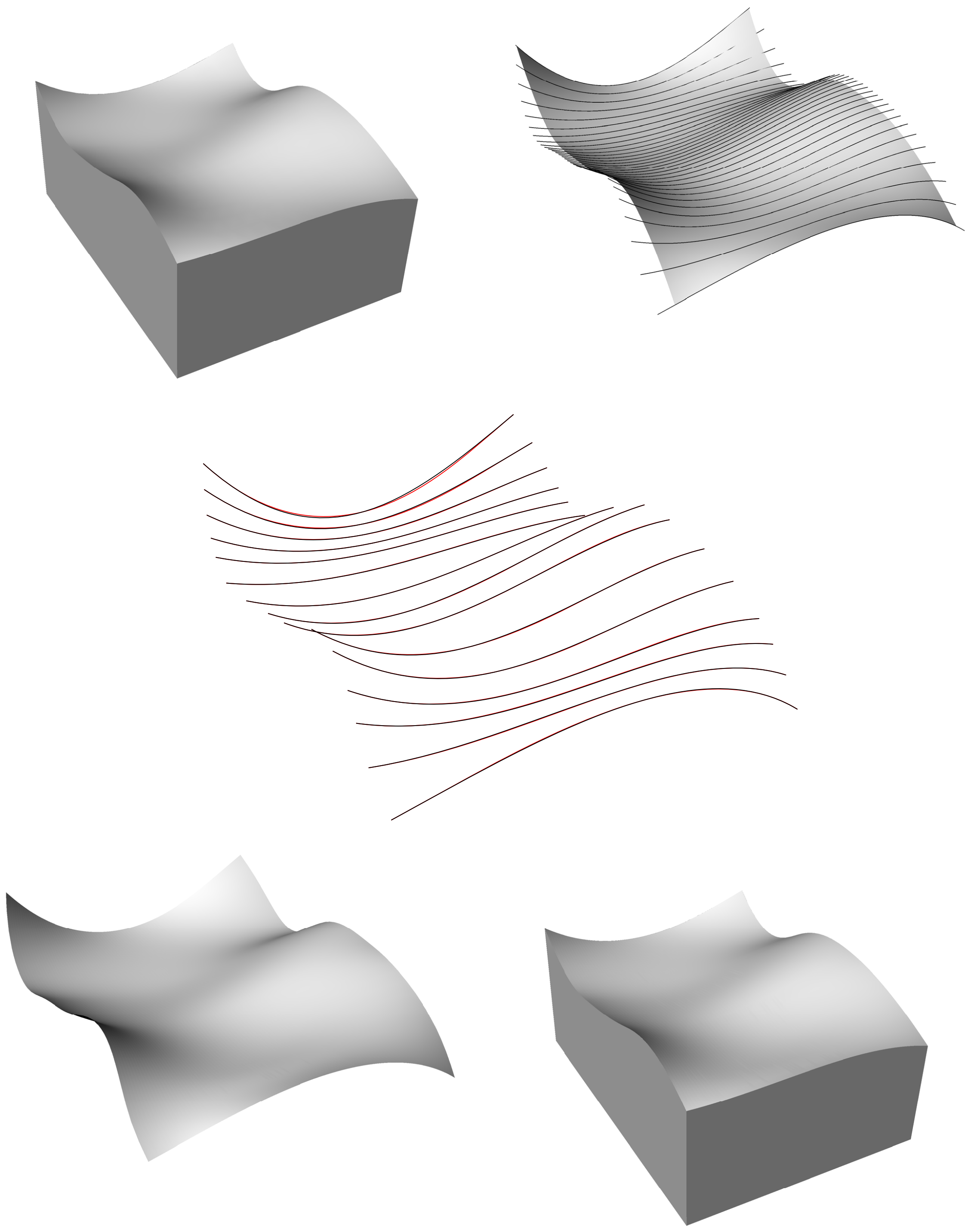


Figure 4.5: Top left: The block cut by a tensor product spline surface with 4×6 control points. Top right: The spline surface with 33 planar sweep curves (here isocurves) extended to blade length. Center: Selection of sweep curves (black) and discrete elastic curves (red) with the same boundary conditions. Bottom left: The surface obtained by lofting the elastic curves. Bottom right: The block cut by the elastica-foliated surface.

used for hot blade cutting and should be close to the sweep curves. We can then loft the elastic curves to get an image of the surface that will be cut, see Figures 4.5 and 4.6.

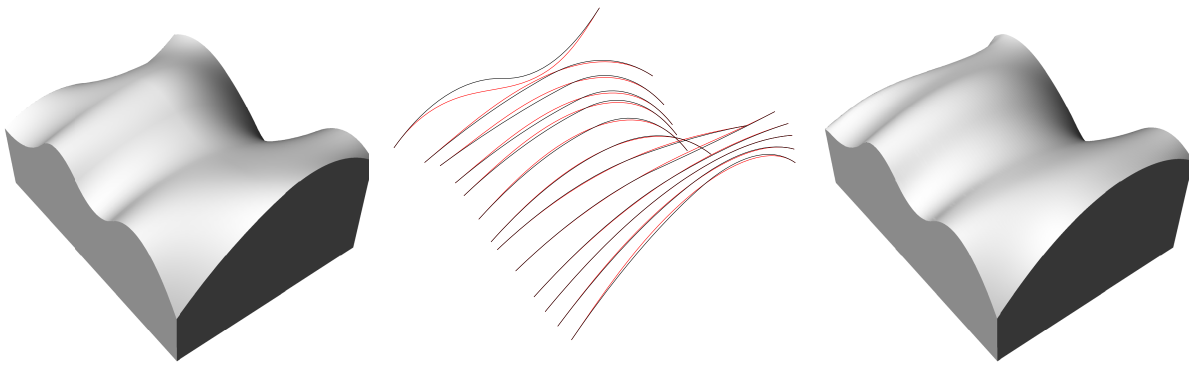


Figure 4.6: Left: Surface design with 4×6 control points. Center: Selection of sweep curves (black) and discrete elastic curves (red) with the same boundary conditions. Right: The block cut by the surface obtained from lofting the elastic curves.

The robot data, i.e. the end points and tangents, can be taken directly from the sweep curves. We find the discrete elastic curves and loft a new surface in order to visualize the result before cutting it. The new surface will be different from the original design, but that is not an issue here, since we are in a design process which aims at designing elastica-foliated surfaces. If the result does not fit the designer's idea, the original surface can be edited, and the elastica algorithm can be run again to get a new result. For this to work as a design process, the elastica algorithm has to be stable and fast. Using IPOPT, finding the discrete elastica for 33 sweep curves took around 20 seconds for the test surfaces in this section.

Remark. We do not know exactly how Rhino's plane/surface-intersection function works. Kasper Steenstrup asked about this on McNeel Forums[7] but the developers would not disclose this information. Our guess is, that the function first finds a lot of intersection points and then interpolates these by a spline (usually a cubic spline). Whatever the case, when working with complex geometries, we have observed the following disadvantages: The resulting spline curve can have a very high number of control points (compared to the surface) and it can have repeated interior knots, which may reduce its degree of differentiability. In such cases we approximate the found curve by a simpler spline (i.e. with fewer control points), which has simple knots except for the first and last knots, which have multiplicity equal to the spline degree plus one.

Remark. When initializing the optimization, instead of taking the initial guess for the turning angles from the sweep curves, we can (except for the first curve) use the result from the preceding discrete elastica.

A different type of surface can be obtained by doing multiple cuts. Instead of just designing one surface, the designer can create several intersecting surfaces and these can then be cut in order, with the possibility of rotating the block between cuts (if it is not much longer than it is wide). The resulting surface will typically be piecewise smooth with clear edge curves where the different cutting surfaces intersect, see Figure 4.7.

Designs with multiple blocks

For designs with multiple blocks we use basically the same procedure as above. We start with a prescribed set of blocks and design a surface that intersects the block volume. For each block

[7]`http://discourse.mcneel.com/`

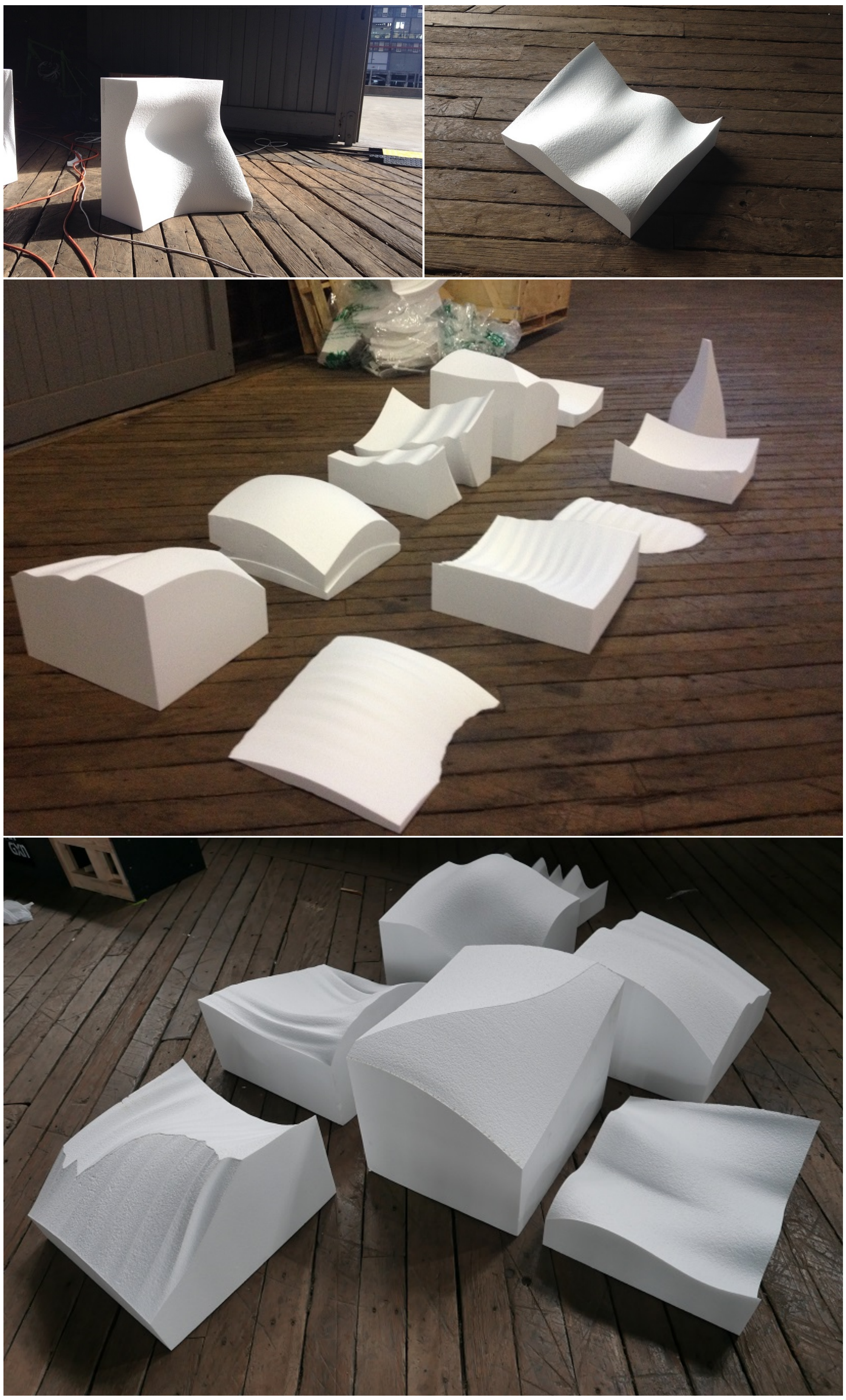


Figure 4.7: Single block designs cut at the RobArch 2016 *Superform* workshop. The bottom photo contains blocks that were cut twice.

we extract sweep curves and find discrete elastica with the same boundary conditions, but this time we do not extend the sweep curves before finding the elastica. If we first extend the curves to blade length and then find the discrete elastica, two adjacent curves will not connect at the block boundary, see Figure 4.8.

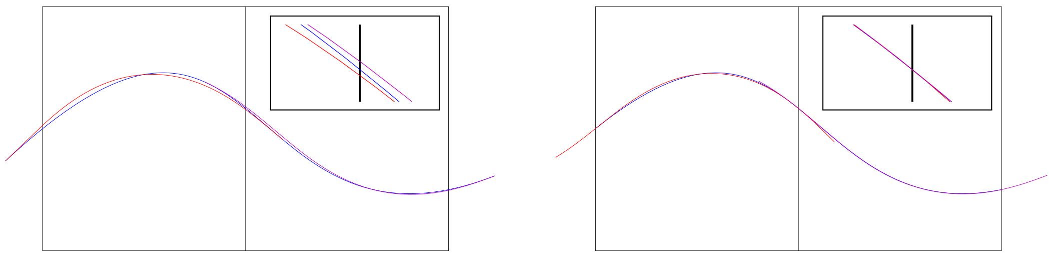


Figure 4.8: Left: Two segments of the blue spline curve are extended to blade length and then the elastic curves (red and magenta) with the same boundary conditions are found. Right: The segments of the blue curve that are inside the blocks are used for finding the elastic curves which are then extended to blade length. The close-ups show that we have tangent continuity at the block boundary in the latter case only.

We want the elastic curves to meet with tangent continuity at the block boundaries, which is guaranteed if we solve the boundary value problem without extending the sweep curves first. We can loft the elastic curves and get an image of the final design. The tangent continuity is of course only ensured at the sampled curves, while in between, we cannot even ensure continuity. However, if the curves are sampled closely, we will be within production tolerances.

The problem is now that the elastic curves do not have the same length as the blade, so we cannot use their endpoints and tangents as robot data. We must therefore extend the elastic curves to the blade length, for which we need to know their control parameters. We find these by solving the optimization problem defined in Section 4.2. When the control parameters are found, we can easily extend the curves; the elastic curve $\boldsymbol{\gamma}_{(k,s_0,\ell,S,\phi,\hat{x},\hat{y})}$ has constant speed ℓS, so if we want a segment of length L we must have $L = \ell S \Delta t$. If we want the t-values to be symmetric around the interval $[0,1]$ (so we extend by the same amount at each end), we can take the interval $[-a, 1+a]$ with $a = (L/(\ell S) - 1)/2$. In this case the endpoints and end tangents that must be given as robot data are $\boldsymbol{\gamma}(-a)$, $\boldsymbol{\gamma}(1+a)$, $\boldsymbol{\gamma}'(-a)$ and $\boldsymbol{\gamma}'(1+a)$, see Figure 4.9.

Remark. The algorithm for finding the control parameters was implemented using IPOPT, which is not directly compatible with Rhino. For multiple block designs at the RobArch workshop, when the discrete elastica had been found, the polylines had to be exported from Rhino. Then we would run a Python script to find control parameters and generate the robot code.

Figure 4.10 shows some designs, produced at the workshop.

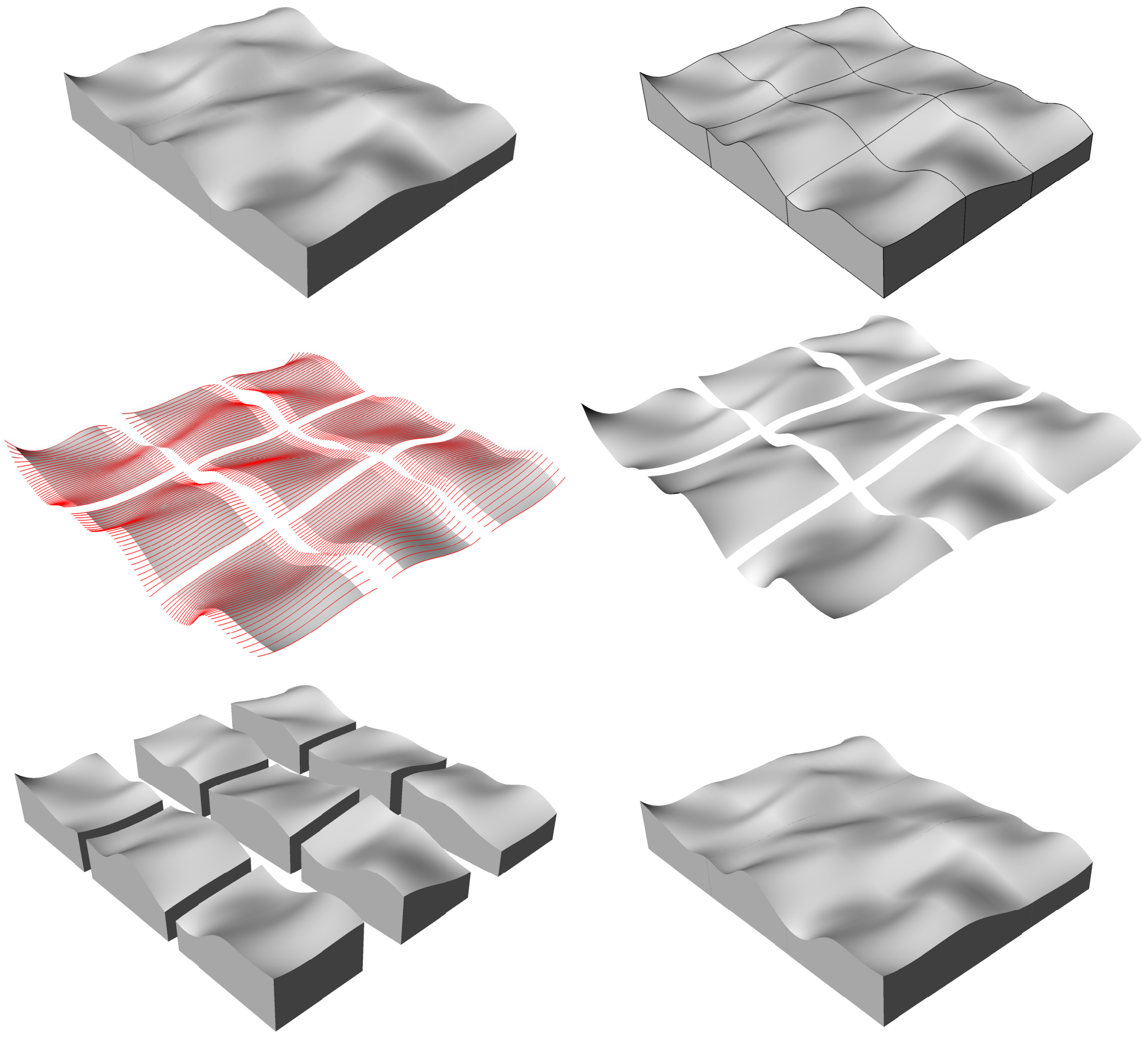


Figure 4.9: Top left: Surface design based on composite cubic Bézier curves. Top right: The design divided into block pieces. Center left: Each surface piece with (analytic) elastic curves extended to blade length. Center right: Surfaces obtained by lofting the elastic curves. Bottom left: The blocks cut by the elastica-foliated surfaces. Bottom right: The final result.

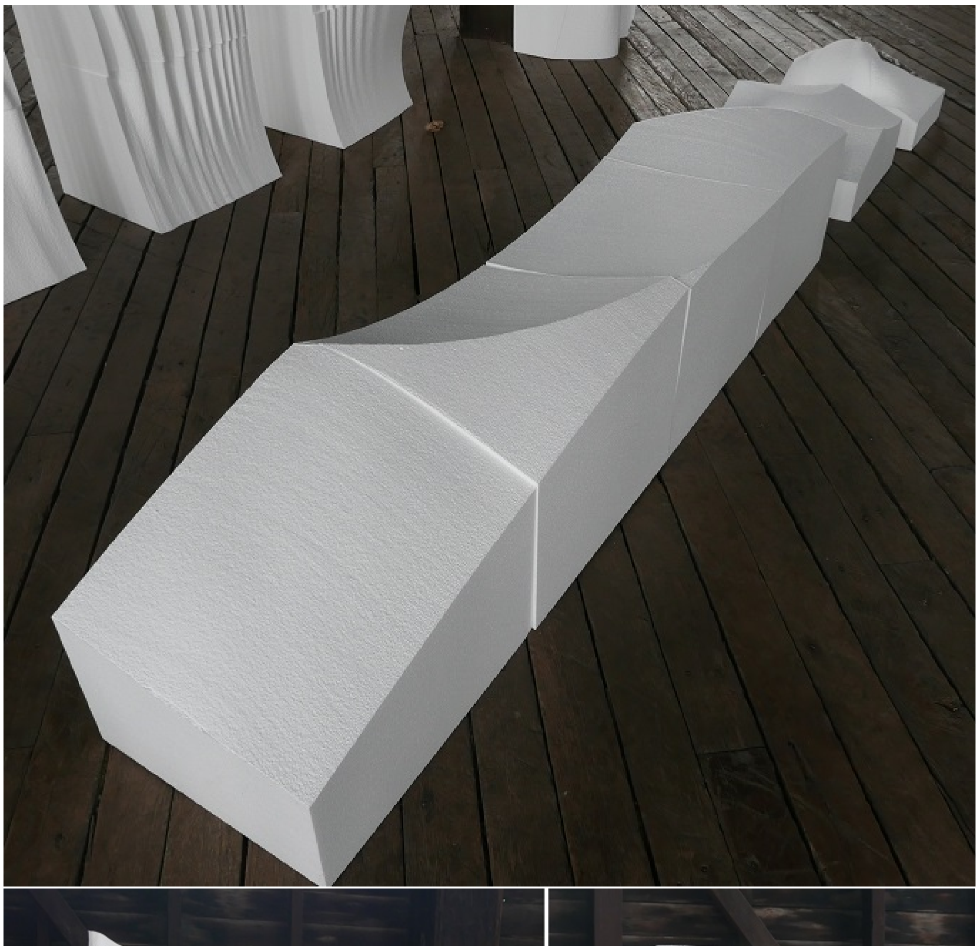


Figure 4.10: Designs with multiple blocks cut at the RobArch 2016 *Superform* workshop.

Chapter 5

Simple surface approximation

In this and the following two chapters we describe how to rationalize an arbitrary surface design to get a surface consisting of parts that can be produced by hot blade cutting. We first describe the basic approximation method for a surface that can be well approximated by an elastica-foliated surface. We then move on to more complicated examples that need to be subdivided, and we discuss different segmentation strategies. Finally, we describe how to make sure that adjacent pieces of a segmented surface meet with tangent continuity.

In this chapter we assume that we are given a surface which is well suited for elastica approximation, and for which an obvious cutting direction exist, see Figure 5.1 for an example. Our approximation method is similar to the one described in the previous chapter: find a set of planar sweep curves on the surface and approximate these by elastic curves. We then sweep a new surface using these curves. If the planar curves are sampled densely and they are approximated well, the resulting surface will be close to the original.

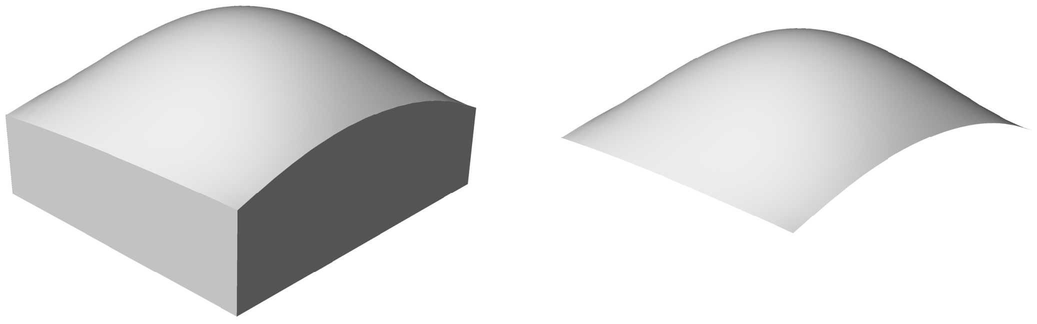


Figure 5.1: Design by Odico for hot blade cutting experiments, to be cut from a block with a 600 mm × 670 mm base. The surface has areas of positive Gaussian curvature and thus cannot be produced by hot wire cutting. On the other hand, it is no challenge for the hot blade.

5.1 Parallel intersecting planes

As before, we can use parallel planes spread out along the cutting direction to find sweep curves. If we orient the coordinate system, so the cutting direction is the y-direction as in the previous chapter, we again need the x- and z-coordinates of the sweep curves. We now approximate these curves by elastica using the algorithm from Chapter 3. To get the relevant data for the robot code, we extend the elastic curves to blade length, and we are ready to fabricate the surface, see Figure 5.2.

We cannot get an exact image of the resulting surface, since we are not sure how the robot interpolates (or approximates) the data it is given. Even if we did know the exact interpolation

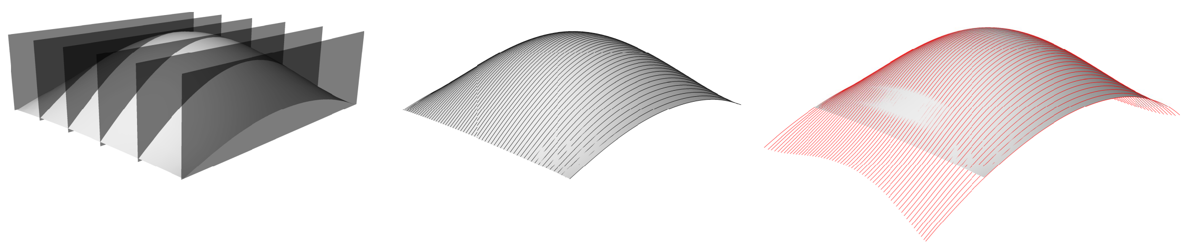


Figure 5.2: Left: The surface is intersected by parallel planes. Center: Sweep curves are found as intersections of the planes and the surface. Right. The sweep curves are approximated by elastic curves, which are extended to the length of the cutting tool (here 1 m).

scheme, finding the elastica for the points in between the sweep curves would require that we solve the boundary value problem, which we can only do by optimization. The best we can do is therefore to take a dense sample of sweep curves and loft the approximating elastic curves, see Figure 5.3.

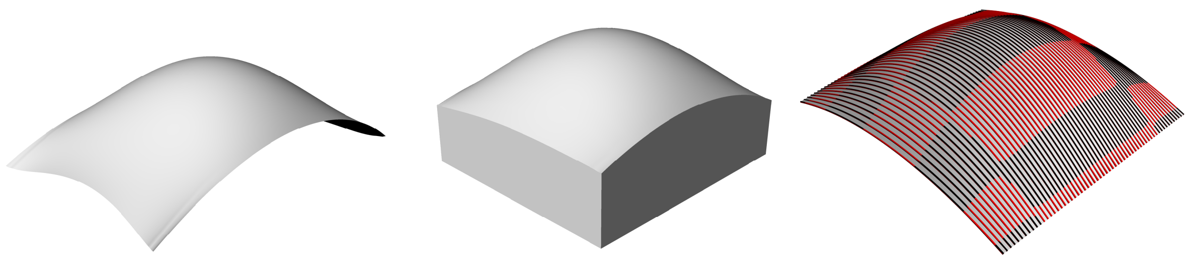


Figure 5.3: Left: The surface foliated by the (extended) elastic curves. Center: The elastica-foliated surface that will be cut from the block. Right: The original surface shown with the sweep curves (black) and the elastic curves (red).

The real surface produced by hot blade cutting can be seen in Figure 1.3. It was cut from a block with a 600 mm × 600 mm base, so it misses 35 mm on two sides, compared to the surface in Figure 5.1.

5.2 Non-parallel intersecting planes

Until now, we have used parallel planes to find sweep curves, which is fine if we are cutting with the tube for which cutting direction is not important. The flat blade, on the other hand, is designed to cut with its thin edge. When the intersecting planes are parallel and the blade shape changes, the velocity of the blade will be in a direction which is not normal to the blade plane. For cutting with the flat blade, it would be better to choose the planes such that their normals point in the cutting direction, which is equivalent to the sweep curves being geodesics.

Since we only want planar sweep curves, we cannot in general demand that they be geodesics. Instead, we take a curve on the surface which to some extent follows the cutting direction. It could for example be a planar curve through the center of the surface, or, in the case of a NURBS surface, the central isocurve in the cutting direction. By taking planes along this *trace curve* which are normal to the curve tangent, we obtain a new varying cutting direction, see Figure 5.4. This will not ensure that all points on the blade are moving in a direction normal to the blade plane, but it will in many cases be better than using parallel planes.

If the planes are based on a trace curve, it will often happen that the sweep curves do not cover the surface sufficiently at the beginning and end of the cut. NURBS surfaces can be extended smoothly, and we do this to find out how the blade should be shaped while moving into and

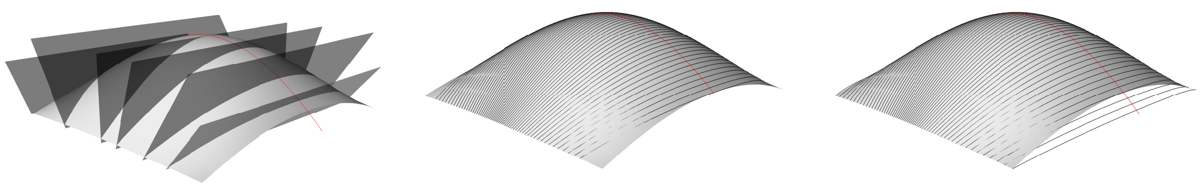


Figure 5.4: Left: Surface with intersecting planes based on a trace curve (red). Center: The sweep curves from trace curve planes at points on the surface. The sweep curves does not cover the surface sufficiently. Right: The trace curve is extended outside the surface, so the sweep curves cover the surface.

out of the EPS, see Figure 5.4. For surfaces that are not given by a parameterization, such as meshes or point clouds, this is not possible, but there are methods for extending such surfaces.

Remark. In fact we do not need to extend the surface (just the trace curve, in order to get intersecting planes), since we are only interested in the sweep curve segments that lie on the original surface. However, since we have not developed a method for finding one elastica based on several curve segments with space in between, we have chosen to extend the surface, so we get a continuous curve, which we can approximate.

Since the sweep curves' planes are not aligned with the coordinate system, we will need a transformation to get the curves in a two-dimensional format, which is needed for the curve approximation algorithm. It is trivial to get a frame for the curve planes, using the Gram-Schmidt process.

There are of course infinitely many ways to frame each plane, but one choice ensures minimal rotation from one frame to the next. This way to frame the curve, which was first described by Bishop [Bis75], is called the *frame of minimal rotation* and it is unique up to a single global rotation.[1] While the Frenet-Serret frame does not always exist, any regular curve has a frame of minimal rotation.

The frame of minimal rotation $(\mathbf{M}_1, \mathbf{M}_2, \mathbf{T})$ for a curve ($\mathbf{T}$ being the unit tangent for the curve) satisfies the equations

$$\begin{aligned} \dot{\mathbf{T}} &= k_1\mathbf{M}_1 + k_2\mathbf{M}_2, \\ \dot{\mathbf{M}}_1 &= -k_1\mathbf{T}, \\ \dot{\mathbf{M}}_2 &= -k_2\mathbf{T}, \end{aligned} \tag{5.1}$$

where k_1 and k_2 are functions of the arc length parameter, satisfying $k_1^2 + k_2^2 = \kappa^2$. We will not detail how to compute the frame here, but we note that for a planar curve, the Frenet-Serret frame has minimal rotation.

Let $\mathbf{r}\colon I \to \mathbb{R}^3$ denote the trace curve with I being its domain interval. For $\tau \in I$, let P_τ denote the intersecting plane at $\mathbf{r}(\tau)$. A point $\mathbf{p} \in P_\tau$ is assigned the two-dimensional coordinates (u, v) with

$$u = (\mathbf{p} - \mathbf{r}(\tau)) \cdot \mathbf{M}_1(\tau), \quad v = (\mathbf{p} - \mathbf{r}(\tau)) \cdot \mathbf{M}_2(\tau).$$

Since the sweep curve at $\mathbf{r}(\tau)$ is a planar spline, we only need to apply the above transformation to its control points, in order to describe it in P_τ-coordinates. In P_τ, we can now find an elastica $\boldsymbol{\eta}\colon \mathbb{R} \to \mathbb{R}^2$, such that the segment $\boldsymbol{\eta}|_{[0,1]}$ approximates the sweep curve, and we can move the

[1]Bishop calls this a relatively parallel adapted frame. It is also known as a zero-twisting frame, a perpendicular frame, and a Bishop frame.

elastic curve back to the three-dimensional space as the curve $\boldsymbol{\gamma}\colon \mathbb{R} \to \mathbb{R}^3$ given by

$$\boldsymbol{\gamma}(t) = \mathbf{r}(\tau) + \eta_1(t)\mathbf{M}_1(\tau) + \eta_2(t)\mathbf{M}_2(\tau), \quad t \in \mathbb{R}.$$

Finally, we can loft the elastic curves to get an image of the expected result, see Figure 5.5.

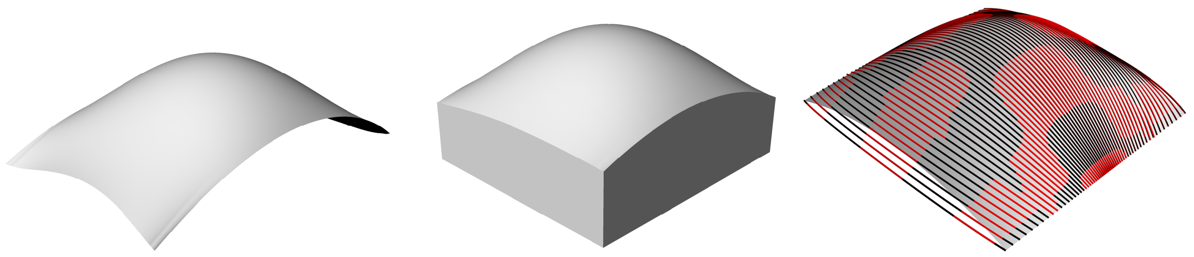


Figure 5.5: Result using intersecting planes based on the central isocurve. Left: The surface foliated by the (extended) elastic curves. Center: The elastica-foliated surface that will be cut from the block. Right: The original surface shown with the sweep curves (black) and the elastic curves (red).

5.3 Evaluation

We shall now describe how we evaluate our results. There are two things to evaluate: How close is the rationalized design to the original? Is the blade velocity close to the ideal cutting direction, and how much does it vary along the blade?

From the optimization we have, for each elastic curve, the value of the objective function $\mathscr{E}$, i.e. the squared L^2-distance between the sweep curve and the approximating elastic curve. As in section 3.2 we compute the normalized residual $R_4 = \sqrt{\mathscr{E}/L^2}$, where L is the length of the sweep curve, which gives us a scale invariant number. This number tells us how well the optimization algorithm performed, but we want a concrete way of evaluating the result; so we also compute the (one-sided) Hausdorff distance for each curve pair, that is,

$$d_H(\Gamma, \Lambda) = \max_{\mathbf{x}\in\Gamma} \min_{\mathbf{y}\in\Lambda} \|\mathbf{x} - \mathbf{y}\|$$

where Γ and Λ are sets of sampled points on the elastic curve and the sweep curve respectively. The errors are collected in Table 5.1, and we observe that sweep curves obtained from parallel planes are approximated best.

	PP	TCP
$\max_\tau R_4$	7.26e-4	1.86e-3
$\text{mean}_\tau R_4$	3.42e-4	6.15e-4
$\max_\tau d_H$ [mm]	1.61	2.36
$\text{mean}_\tau d_H$ [mm]	0.83	1.17

Table 5.1: Error data for surface approximation with parallel intersecting planes (PP) and planes based on a trace curve (TCP). In each case 60 sweep curves were approximated and the residual R_4 and the Hausdorff distance d_H were calculated. The Hausdorff distance is in millimetres and calculated from 1000 points sampled on each curve. We list the maximal value of R_4 and d_H and the mean over the 60 curve pairs.

We must also evaluate the cutting motion, since we do not want different parts of the blade to move with very different speeds. Moreover, we would like the blade to move in a direction close to the normal of its plane, so we will find a way to calculate the velocity of each point on the blade, to see if this is the case.

For $\tau \in I$, let $(u(\tau), v(\tau)) : \mathbb{R} \to \mathbb{R}^2$ denote the coordinates of the elastic curve in P_τ. The corresponding curve in $\mathbb{R}^3$ is

$$\boldsymbol{\gamma}(\tau) = \mathbf{r}(\tau) + u(\tau)\mathbf{M}_1(\tau) + v(\tau)\mathbf{M}_2(\tau)$$

We want to compute the velocity vector at each point $\boldsymbol{\gamma}(\tau)(t)$, $t \in [0,1]$, on the curve $\boldsymbol{\gamma}(\tau)$. This gives us a vector field $\boldsymbol{\gamma}'(\tau)$ along the curve, given by

$$\begin{aligned}\boldsymbol{\gamma}'(\tau) =& \mathbf{r}'(\tau) + u'(\tau)\mathbf{M}_1(\tau) + v'(\tau)\mathbf{M}_2(\tau) + u(\tau)\mathbf{M}_1'(\tau) + v(\tau)\mathbf{M}_2'(\tau) \\ =& \mathbf{r}'(\tau) + u'(\tau)\mathbf{M}_1(\tau) + v'(\tau)\mathbf{M}_2(\tau) - u(\tau)\left(\mathbf{T}'(\tau)\cdot\mathbf{M}_1(\tau)\right)\mathbf{T} - v(\tau)\left(\mathbf{T}'(\tau)\cdot\mathbf{M}_2(\tau)\right)\mathbf{T}(\tau) \\ =& \mathbf{r}'(\tau) + u'(\tau)\mathbf{M}_1(\tau) + v'(\tau)\mathbf{M}_2(\tau) - \left(\mathbf{T}'(\tau)\cdot\left(u(\tau)\mathbf{M}_1(\tau) + v(\tau)\mathbf{M}_2(\tau)\right)\right)\mathbf{T}(\tau)\end{aligned}$$

where we have used the properties of the frame of minimal rotation

$$\frac{\mathrm{d}\mathbf{M}_i}{\mathrm{d}\tau} = \frac{\mathrm{d}\mathbf{M}_i}{\mathrm{d}s}\frac{\mathrm{d}s}{\mathrm{d}\tau} = -\frac{\mathrm{d}s}{\mathrm{d}\tau}\left(\frac{\mathrm{d}\mathbf{T}}{\mathrm{d}s}\cdot\mathbf{M}_i\right)\mathbf{T} = -\left(\frac{\mathrm{d}\mathbf{T}}{\mathrm{d}\tau}\cdot\mathbf{M}_i\right)\mathbf{T}, \quad i = 1,2.$$

While we can easily compute $\mathbf{r}'$ and $\mathbf{T}' = \frac{\mathbf{r}''}{\|\mathbf{r}'\|} - \frac{\mathbf{r}'\cdot\mathbf{r}''}{\|\mathbf{r}'\|^3}\mathbf{r}'$, we cannot find (u', v'), which determines how the elastic curve changes inside the plane P_τ as τ changes, without knowing how the robot interpolates the data. In any case, we are only taking a discrete set of intersecting planes, corresponding to a set of parameters $\tau_i \in I$, so we will just use the two-sided difference quotient

$$u'(\tau_i) \approx \frac{u(\tau_{i+1}) - u(\tau_{i-1})}{\tau_{i+1} - \tau_{i-1}}$$

and likewise for v'. For the first and last τ-values we can, of course, only take the one-sided difference quotient.

For each $\tau \in I$ we can now compute the velocity $\boldsymbol{\gamma}'(\tau)(t)$ for different $t \in [0,1]$. We can compare the magnitudes of these vectors to see how much the speed of the blade varies, and we can compute the angle $\phi(\tau, t)$ between the velocity and the plane normal as

$$\phi(\tau, t) = \arccos\left(\frac{\boldsymbol{\gamma}'(\tau)(t)}{\|\boldsymbol{\gamma}'(\tau)(t)\|}\cdot\mathbf{T}(\tau)\right) = \arccos\left(\frac{\boldsymbol{\gamma}'(\tau)(t)}{\|\boldsymbol{\gamma}'(\tau)(t)\|}\cdot\frac{\mathbf{r}'(\tau)}{\|\mathbf{r}'(\tau)\|}\right)$$

See Table 5.2 and Figure 5.6 for results. We observe that, at least in this example, using non-parallel planes does indeed result in a cutting direction closer to the ideal one. On the other hand it results in a less uniform speed distribution along the blade.

	PP	TCP
$\max_{\tau,t}\phi(\tau,t)$ [°]	41.1	40.7
$\mathrm{mean}_{\tau,t}\phi(\tau,t)$ [°]	15.0	10.3
$\max_\tau \mathrm{mean}_t \phi(\tau,t)$ [°]	28.8	15.5
$\min_\tau \frac{\min_t \|\boldsymbol{\gamma}'(\tau)(t)\|}{\max_t \|\boldsymbol{\gamma}'(\tau)(t)\|}$	0.754	0.449

Table 5.2: For each of the 60 elastic curves we calculate the velocity in 100 points and find the angle deviation. We list here the maximal and mean angle deviation over all 600 velocities (in degrees). We also compute the mean of the angle deviation for each curve, and we list here the highest value. Finally for each curve we find the points with lowest and highest speed and we compute the quotient. We list here the value for the curve with the lowest value.

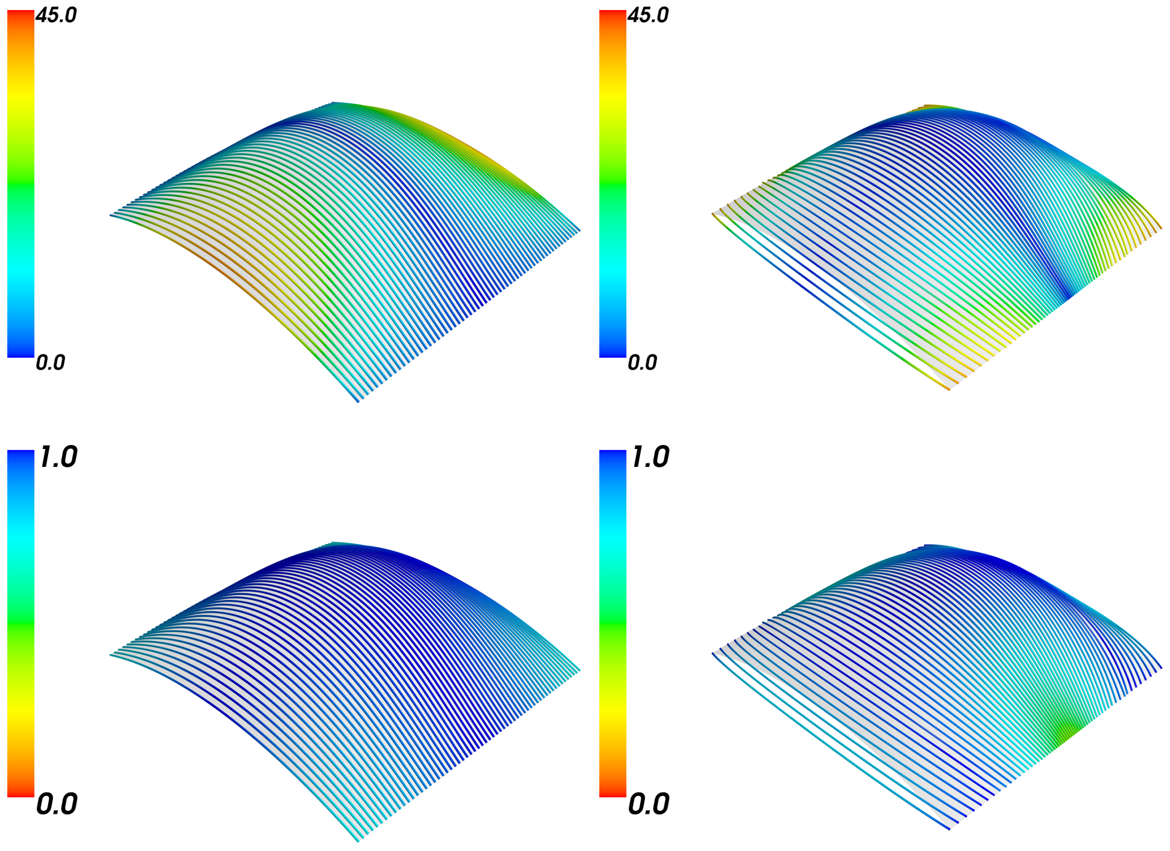


Figure 5.6: Visualization of the results. Left: Parallel intersecting planes. Right: Intersecting planes based on trace curve. Top: Colours show the angle deviation from the plane normal (in degrees). Bottom: The colour of each curve point is the speed at that point divided by the maximal speed along the elastic curve. As one might expect, if we look at angle deviation, the non-parallel planes are best, but if we consider speed variation, the parallel planes are best.

CHAPTER 6

Surface segmentation

When given an arbitrary design, we cannot be sure that it can be well approximated by a single elastica-foliated surface. If we simply find planar sweep curves as above, these may not resemble elastic curves at all. Moreover, the blade has a certain length, so if the surface is too big, we cannot produce it by hot blade cutting. Another limitation is that the set of possible blade shapes is much "smaller" than the set of elastic curve segments, as described in Section 3.3. For example, the surface in Figure 6.1 (left) is foliated by one elastica, but we could never hope to cut this in one go.

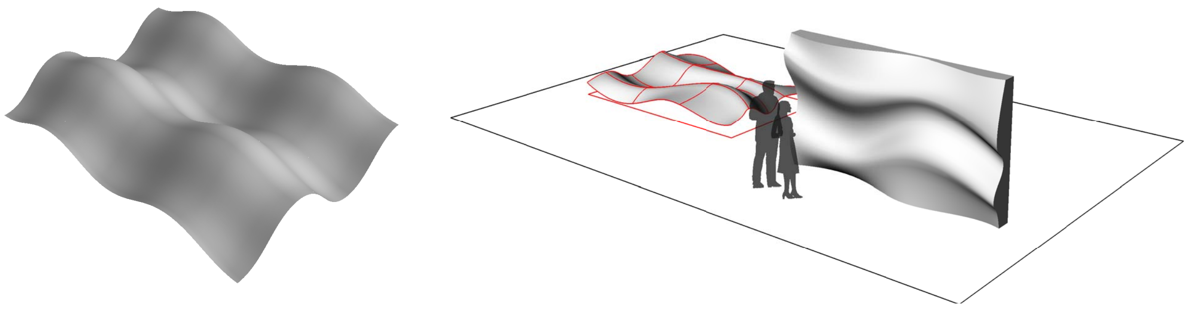


Figure 6.1: Left: Surface obtained by moving an elastic curve along another elastic curve. Right: Curved wall design by 3XN architects with dimensions 4200 mm × 2400 mm.

In this chapter we will discuss methods for subdividing a surface into pieces that can actually be cut. In doing the segmentation, we will only concern ourselves with limitations regarding possible blade shapes and the length of the blade. We will thus focus on segmenting a surface into strips that each, in principle, can be produced in one cut. In practice, we may need to subdivide the strips due to limitation in the EPS block size, but we will not take this into account here. In the next chapter we will approximate the patches of a segmented surface and discuss how to make sure these patches, when produced, fit together in a smooth way.

The work behind this chapter and the algorithms described in it were done in collaboration with Kasper Steenstrup. Most of our ideas were based on the specific case of the curved wall in Figure 6.1 (right), a design created by architects at 3XN with the goal of being a prototype showcase for the BladeRunner project. The implementation was for the most part done in RhinoPython and rely on many built-in Rhino functions. Since Rhino is NURBS based, all surfaces that we consider are NURBS surfaces.

6.1 Preparing for segmentation

In the previous chapter we assumed that the surface was ready for approximation. Here we address two issues that we need to consider before an arbitrary surface can be segmented and approximated: How should we orient the surface in relation to the EPS blocks it must be cut from? In which direction should we cut the surface.

Choosing an orientation in space

In most practical examples from architecture, e.g. wall panels, the surface that we want to produce can be considered as a height map, often over a rectangular domain. This means that we have a fixed direction that we can consider as upwards. Usually the design will be oriented with the z-axis pointing upwards.

The segmentation strategies we present in this chapter depend to some extent on knowing which direction is up. We will therefore describe a way of choosing a bounding box for an arbitrary surface. This box will then define a reference frame and, in particular, an "up"-direction given by the z-axis. Rhino has a built-in function for creating an oriented bounding box for a surface.

If the surface can be described as a height map over a rectangular domain, we of course want the bounding box to be aligned with the domain. A surface of this kind, such as the curved wall above, has four planar edge curves. Rhino has functions for finding these curves and the planes that they lie in. From these we can determine the orientation of the relevant bounding box, see Figure 6.2. If the surface is a graph over a rectangular domain, we choose a bounding box with this rectangular domain as the base.

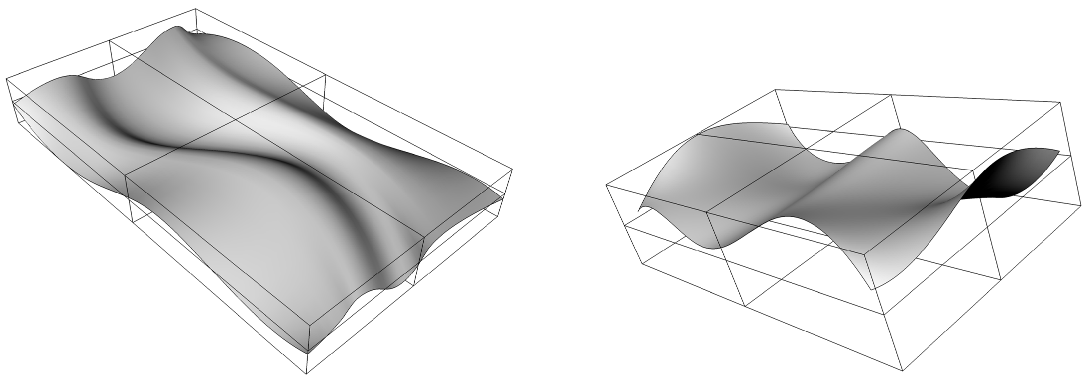


Figure 6.2: Bounding box for surfaces. The curved wall (left) has four planar edge curves and adjacent edge planes are perpendicular, so there is a natural choice for the bounding box.

If there is no direction from which the surface appears as a rectangle, e.g. if the edge curves are not planar or if the edge planes are not perpendicular, there may be no obvious choice for a bounding box. A NURBS surface is always defined on a rectangular parameter domain and it therefore has four "corners" along its boundary. We randomly pick three corners and from these we construct a plane to which we align the bounding box.[1] This often works well, but cases exist where there is a more obvious choice for the bounding box, see Figure 6.3

When we have chosen a bounding box we can can align our coordinate system to it so that we always have the z-direction pointing upwards. This also means that the plane that supports the EPS blocks will be parallel to the (x, y)-plane. The primary movement of the blade will be along vectors in this plane, i.e. the blade will not move so much up or down that it cuts through the tops or the bottoms of the EPS blocks.

[1]In Rhino a plane object has a coordinate system and thus defines a coordinate frame in space.

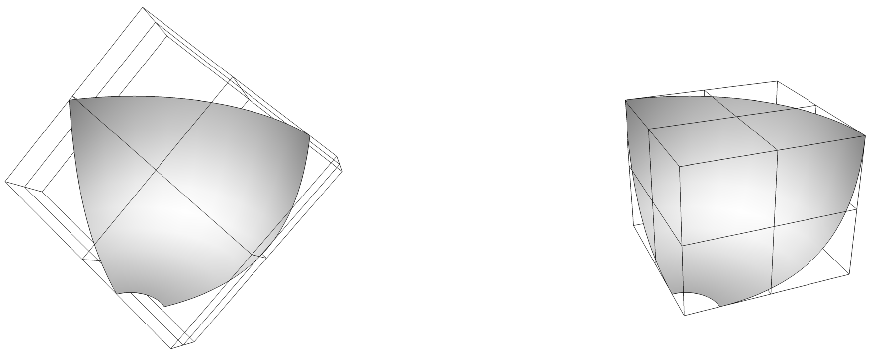


Figure 6.3: Design for the corner piece of a skater ramp. The surface does not have a rectangular boundary, but it still has a natural bounding box. Left: The bounding box based on a plane between three surface corners. Right: The natural bounding box based on the planar boundary curves.

Choosing a general cutting direction

We generally want to segment the surface in the direction perpendicular to the cutting direction, so we must choose this direction before we analyse the surface for where to subdivide. However, when we have a segmented surface, we want to choose the cutting direction for each segment based on its shape. Moreover, as we have seen above, the ideal cutting direction for a single segment need not be constant. For these reasons we will choose a "general direction" of cutting, given by a vector $\mathbf{v}$, and analyse the surface in the direction given by $\mathbf{v} \times \mathbf{z}$, where $\mathbf{z}$ points in the direction of the z-axis, i.e. upwards. Typically the general cutting direction will be along the x- or y-axis.

There may be geometric features that make a specific cutting direction the natural choice, or make the other directions bad choices, see Figure 6.4. Otherwise it makes sense to pick the longest dimension as the cutting direction in order to have as few segments as possible.

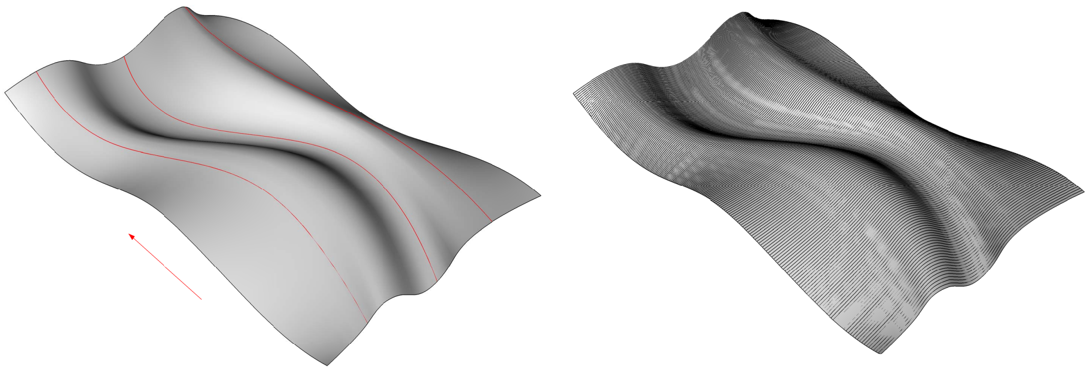


Figure 6.4: Left: For the curved wall, it seems natural to let the cutting direction follow the isocurves in the longest direction. The general cutting direction, which is indicated by the arrow, follows the edge of the bounding box. Right: The design with 200 sweep curves.

Once a general cutting direction $\mathbf{v}$ has been chosen, we will create a set of sweep curves from parallel intersecting planes along a line with direction $\mathbf{v}$. We will base our surface analysis on these sweep curves, our assumption being that the results will not change too much when the cutting direction is changed.

Remark. With the cutting direction chosen above, the sweep curves for the curved wall have points of too high curvature; the blade cannot bend that much without being permanently deformed. For this reason, we went for the opposite cutting direction, i.e. along the short direction, for the production, see Section 7.4. This direction, however, does not have as nice features to determine segmentation, so for describing our methods, we stick with the cutting direction chosen here. A larger version of the wall could be cut, since scaling up the design would reduce its curvature.

6.2 Surface analysis

We will now describe how we analyse a given surface in order to get guidelines for where to subdivide it. We consider three reasons to subdivide: size, inflection points, and whether a good approximation can be achieved.

Size

The most obvious reason for segmenting a surface is if it is too big. Clearly, each strip cannot be wider than the blade length. In fact since we want to bend the blade, there should be a gap, between the strip width and the blade length L_{blade}. Also, we do not want the robots holding the blade ends to collide with the EPS block while cutting, so we must demand that the cutting part of the blade, that is the part which is inside the EPS, should be shorter than the actual blade length.

If we, for example, say that 10% of the blade should be outside the EPS, we have

$$L_{\max} = 0.9\, L_{\text{blade}}.$$

We may assume that a segment of a sweep curve, will be approximated by an elastic curve of roughly the same length. Thus if any sweep curve on the surface is longer than $nL_{\max}$, we need at least n strips.

Inflection points

Another key feature of the surface relevant to segmentation is the number of inflection points on the sweep curves, since the blade can have at most two inflections. We have therefore implemented a method for finding the inflection points of a planar parameterized curve: We compute the signed curvature at a number of sample points on the curve. If the curvatures of two adjacent points have opposite signs, we use a bisection to find the curve parameter for the inflection, see Figure 6.5.

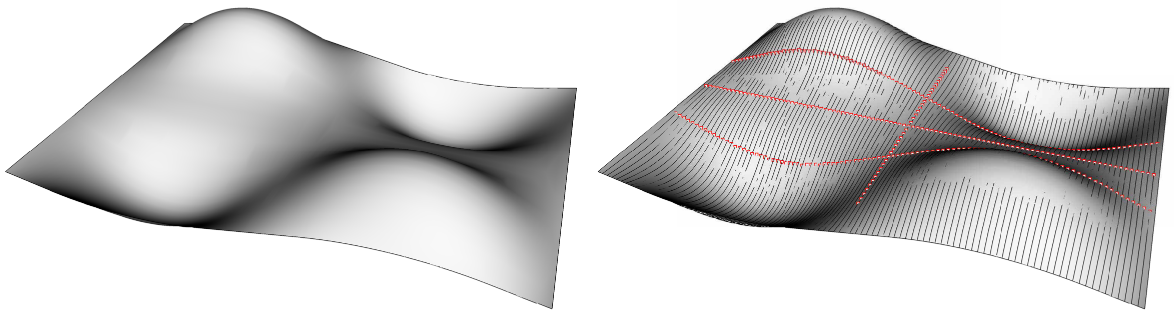


Figure 6.5: Left: Surface designed for rationalization tests. Right: The inflection points are shown for 101 sweep curves on the surface. The central curve is supposed to be a straight line, but the intersection algorithm produces a spline with many small oscillations.

Finding the inflection points will give a guide as to where the surface should be segmented. If a sweep curve has more than two inflections, we need to subdivide it into pieces that each have at most two inflection. In fact, because an elastic curve segment between two inflection points is always symmetric (which limits the possible shapes), it will often be desirable to have at most one inflection point in each sweep curve segment.

If a curve has many inflections close to each other, it may be relevant to consider whether this is indeed part of the designer's idea, or a "byproduct" of either the computer aided design process, or the way the sweep curve was found by intersecting the surface with a plane. If the latter is the case, we can either smoothen the surface in the relevant area before doing the approximation, or we can allow an approximating curve segment with fewer inflections points, than the original curve. In any case, because of the limited flexibility of the hot blade, we cannot produce curves with inflection points very close to each other. We choose to ignore close pairs of inflection points and inflections very close to the surface boundary. In this way we get rid of inflections on curve segments that are almost straight, and we reduce the number of inflection points we need to consider, when choosing where to subdivide.

In our implementation, for each sweep curve we first find the pair of inflections that are closest to each other (in arc length). If this distance is less than 200 mm and if the curve segment between them is almost a straight line, we will ignore the pair, i.e. not consider the points as inflections. We say that the curve segment is almost straight if its length is less than 1% longer than the distance between its ends. We repeat this process until there are no more close pairs of inflection points. If there are now inflections less than 100 mm from the surface boundary, with the curve segment from inflection to boundary being almost straight, we will ignore these as well, see Figure 6.6.

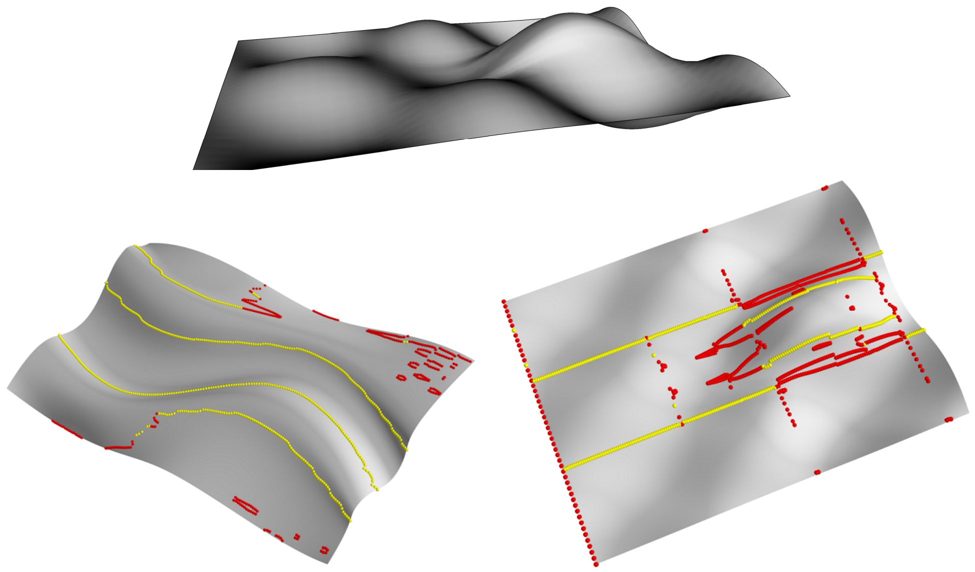


Figure 6.6: Left: The inflection points of 181 sweep curves on the the curved wall. The red inflections are ignored. Right: The inflection points of another surface design by 3XN. Top: The same surface seen from another angle.

Good elastica approximation

Instead of only considering inflection points, we may try to determine whether a given sweep curve can be approximated well by an elastic curve segment. Or rather, if we are going to subdivide the curve, how long segments can be approximated well? To answer this we first need to define what it means to be well approximated, and we need a (preferably fast) way to determine whether a curve satisfies this.

We will say that an approximation is good if the normalized L^2-distance $R_4 = \sqrt{\mathscr{E}/L^2}$ is smaller than some value $\varepsilon > 0$. It will be too time consuming to actually do the optimization, so we will compute this value for the canonical initial guess found by the algorithm described in Section 3.2. Letting $\mathbf{r}\colon [a,b] \to \mathbb{R}^2$ denote the sweep curve's 2D representation, we define $F_a\colon [a,b] \to [0,\infty[$ by

$$F_a(t) = R_4\left(\mathbf{p}_0(\mathbf{r}|_{[a,t]})\right) = \sqrt{\frac{\mathscr{E}\left(\mathbf{p}_0(\mathbf{r}|_{[a,t]})\right)}{L(a,t)^2}},$$

where $\mathbf{p}_0(\mathbf{r}|_{[a,t]})$ is the control point vector for the initial guess for an elastica approximating the curve segment $\mathbf{r}|_{[a,t]}$, and $L(a,t)$ is the length of that segment. We will say that the segment can be well approximated if $F_a(t) \le \varepsilon$.

This is a "better safe than sorry"-choice, meaning that we may discard segments because of a "bad" initial guess that might still have led to a good optimization result; on the other hand, if the initial guess is good, the optimization result should be even better. We also remark the subscript a, which denotes the dependence of F_a on the start point for the curve segment.

Clearly $F_a(a) = 0$. If $F_a(b) > \varepsilon$, we use the bisection method to find the largest $t_0 \in [a,b]$, such that $F_a(t_0) \le \varepsilon$. Once we have thus found the longest segment $\mathbf{r}_{[a,t_0]}$ that can be well approximated, we can repeat the procedure using F_{t_0} to find the next segment of $\mathbf{r}$ that can be well approximated. This will give us a suggestion for a subdivision of $\mathbf{r}$ into segments that can each be well approximated, but we want more freedom in choosing the segmentation. We therefore run the same procedure on the reversed curve, which will provide us with subintervals of $[a,b]$ in which subdivision is suggested, see Figure 6.7.

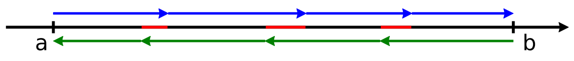


Figure 6.7: On the black axis, we have the parameter interval $[a,b]$ for the sweep curve. The blue arrows show the segments that can be well approximated by elastica, based on the bisection method. The green arrows show the same but with the method used on the reversed curve, i.e. from b to a. The red regions are suggested for subdivision, since any segment from one red region to the next can be well approximated.

Remark. Of course, there may be better ways of determining whether a curve segment can be well approximated. Instead of simply computing R_4, one could use a combination of the residuals we calculated to evaluate the initial guess algorithm. The residual R_1 actually measures how close the original curve segment is to satisfying the Euler-Lagrange equation for the elastica.

6.3 Segmentation

Above we have described the different reasons for segmentation and how we analyse the surface to find guidelines as to where we should segment it, but we have not actually segmented any surfaces. We will now describe different ways to segment based on the surface analysis.

Parallel strips

The simplest segmentation one can think of is parallel strips. This will also make production easy, since it will only require box-shaped EPS blocks. The strips should naturally be aligned with the chosen general cutting direction. Let us say that we cut in the y-direction, and thus want to segment in the x-direction. We can let all the strips have the same width, or we can base the width of each strip on the features of the surface.

As above, let $L_{\max}$ denote the maximal length of the part of the blade used for cutting. For each sweep curve $\mathbf{r}\colon [a,b] \to \mathbb{R}^2$, we now find the largest parameter $t \in [a,b]$, such that

- the length of the segment $\mathbf{r}|_{[a,t]}$ is less than $L_{\max}$,
- the segment $\mathbf{r}|_{[a,t]}$ contains at most two inflection points, (6.1)
- $F_a(t) \leq \varepsilon$.

Now we have a parameter t for each sweep curve. We compute the x-value at these curve points (in 3D-space) and choose the smallest, say x_0. Our first segmentation line is given by $x = x_0$, see Figure 6.8. We can now for each sweep curve find the segmentation parameter t_0 where it passes the line $x = x_0$. If this parameter is not unique, we pick the largest that is smaller than the t we found above. We repeat the procedure by for each sweep curve finding the largest $t \in [t_0, b]$ such that (6.1) is satisfied, with t_0 in place of a.

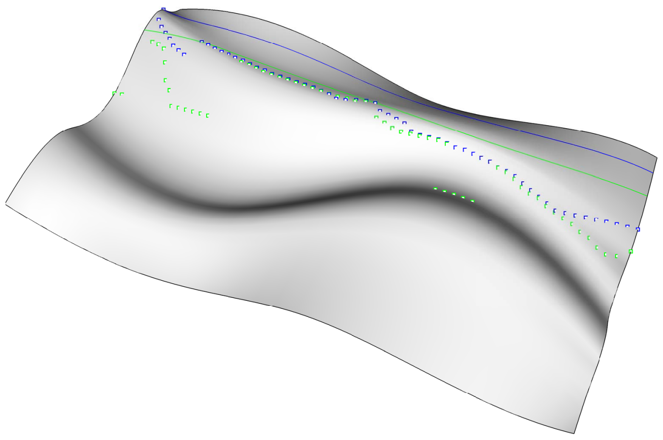


Figure 6.8: The blue points indicate the longest segments of each sweep curve that can be approximated well (tolerance $\varepsilon = 0.005$). The blue curve is the segmentation line defined by the point of smallest x-value. The green points indicate the longest segments, starting from the blue curve, that can be well approximated, and the green curve is the second segmentation line.

The above procedure will give us a set of x-values that determine where to segment the surface. If we want more freedom, we can run the procedure on the reversed sweep curves to get overlap regions, similar to those in Figure 6.7.

If we want strips of uniform width, we can simply divide the surface into n equally wide strips. If the sweep curve segments of each strip satisfies the conditions (6.1) (with a and t replaced by the end parameters for the relevant curve segment), we are done; otherwise we increase n by one and check again.

Freeform strips

If we do not restrict our subdivision to parallel strips, we can get more interesting segmentations. As above we can for each sweep curve $\mathbf{r}_i$ find the largest parameter t_i, such that (6.1) is satisfied. We can then create a curve on the surface that either interpolates or approximates the points

$\mathbf{r}_i(t_i)$, and use this a a segmentation curve. The lengths of the found sweep curve segments can vary a lot, see Figure 6.9, so we may want to reduce the distance between the points before interpolating or approximating. We pick a maximal distance δ between adjacent points and start with the shortest sweep curve segment, say $\mathbf{r}_i$. If one of the neighbouring segments is too long, that is, if e.g. $\|\mathbf{r}_{i+1}(t_{i+1}) - \mathbf{r}_i(t_i)\| > \delta$, we lower t_{i+1}.

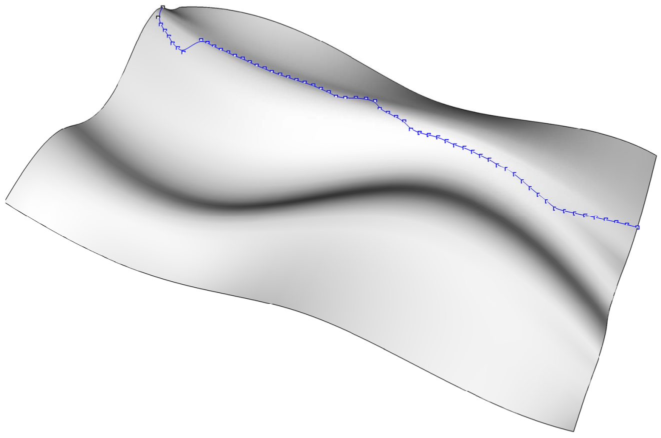


Figure 6.9: Blue points are as in Figure 6.8, but this time we approximate the points with a spline to get a segmentation curve.

We repeat the procedure to get more segmentation curves across the sweep curves. Again, this procedure can be applied to the reversed sweep curves, so we get overlap regions and thus more freedom to choose our segmentation curves.

Strips based on surface geometry

The above methods subdivide surfaces into parts that should be well suited for elastica approximation, but they do not take into account the aesthetics of the segmentation in relation to the surface shape. When the surface is cast in EPS-moulds, the segmentation curves will often be visible, so it is desirable to let these follow the surface geometry.

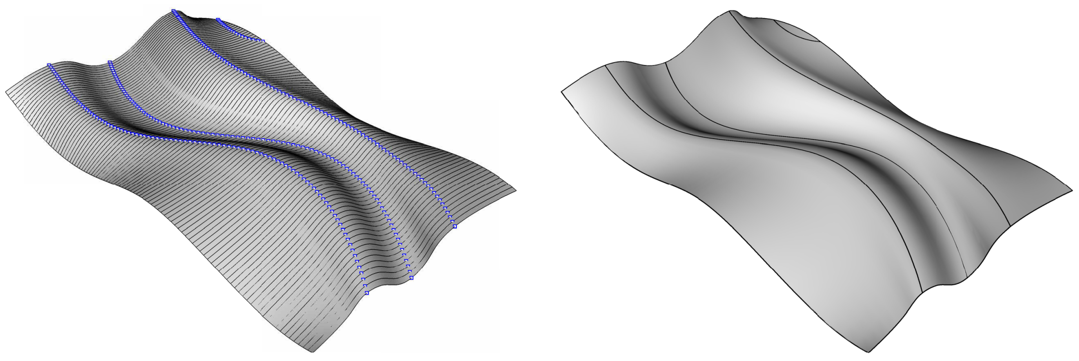


Figure 6.10: Left: The local minima and maxima in the z-direction are found for 100 sweep curves on the curved wall. Right: The surface is subdivided based on curves through the extrema.

Our idea is to find the valleys and ridges of the surface, and see if these give rise to nice segmentation curves. We use here that we have decided on an upwards direction (the z-axis) and we simply find the extrema of the sweep curves in this direction. Alternatively one can find the points of maximal and minimal (signed) curvature on the sweep curves. For the curved wall example, this gives us a very nice segmentation of the surface, see Figure 6.10.

There is no reason to expect that finding ridges and valleys will give segments that can be well approximated. To obtain this, we can do as we proposed for the parallel strips of uniform width: we check if each segment of the sweep curves satisfies (6.1). If this is not the case, we will subdivide the surface further by finding curves between those defined by the extrema. We can project the new curves onto the surface to get more segments and we can repeat this until each segment can be well approximated.

Chapter 7

Advanced surface approximation

In this chapter we assume that we are given a surface with a specified segmentation. We will also assume that EPS pieces have been precut (e.g. by hot wire) in the shapes of the segmentation strips. Our task is to find the best approximation of each segment by an elastica-foliated surface but with the global condition that the segments must meet in a tangent continuous way.

7.1 Planar sweep curves

Our first approach is, as always, to find sweep curves by intersecting the entire surface with parallel planes. We can then subdivide the sweep curves based on the surface segmentation. We now have planar curves, which are split into segments which we must approximate by elastic curves. The optimization scheme we defined in Section 3.4 solves this problem and can ensure tangent continuity where the curves are connected, see Figure 7.1.

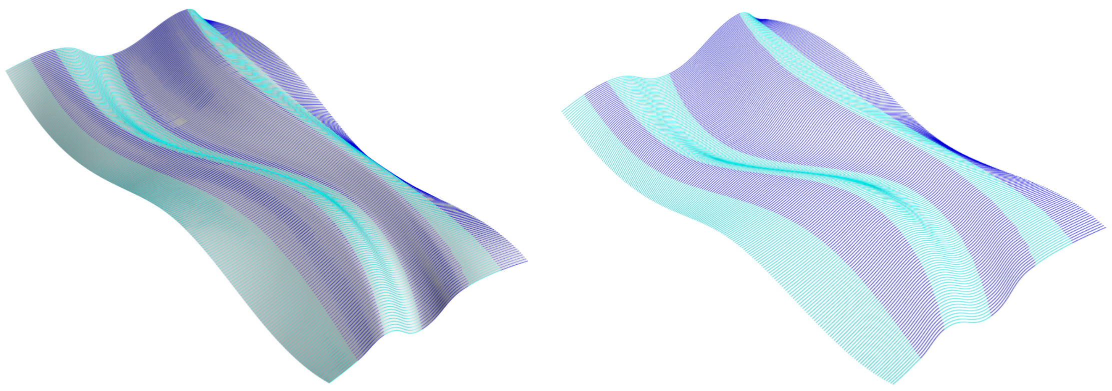


Figure 7.1: Approximation of 3XN's curved wall using 200 sweep curves based on parallel intersecting planes. The valleys and ridges of the surface give a segmentation into four strips; the outer two strips have been further subdivided to obtain a good approximation. Left: The piecewise elastic curves and the original surface. Right: Only the elastic curves.

Again, using parallel planes is not appropriate for the flat blade. Instead we can choose the intersecting planes' normals based on a trace curve on the surface and extend the surface to make sure the sweep curves cover it sufficiently, see Figure 7.2. We now meet a new challenge: if we only wanted to produce the two innermost segments, this method works fine, but on the outer segments we have areas where the blade rotates around a point inside the surface that we want to produce. When the blade moves back over an area which has already been cut, it will cause additional melting of the EPS and thus result in a wrong shape and possibly bad surface quality. Moreover, at the point around which the blade is rotating, far too much EPS will melt and evaporate. Even on the innermost pieces, where the blade path does not overlap, the fact that one end of the blade moves faster than the other can cause the surface quality to differ at the two ends.

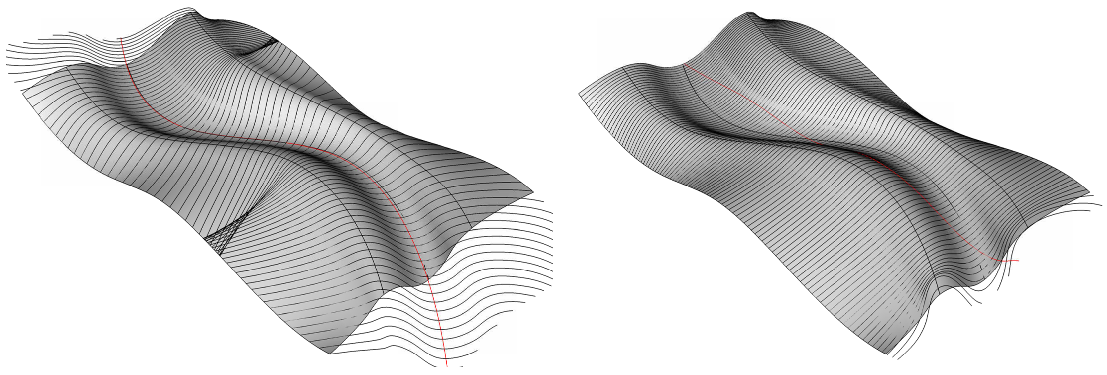


Figure 7.2: Sweepcurves based on trace curves (red). To the left we have used the central isocurve. To the right we have used the curve obtained by intersecting the surface with a vertical plane though its center.

7.2 Piecewise planar sweep curves

In order to avoid the problems described above, we will sacrifice the convenience of using global intersecting planes. Instead we will find different planes for each surface segment. However we still want the sweep curves for the segments to join at the boundaries. We propose two ways of doing this.

The first idea is to sample points along the boundary curves and draw line segments between them. For each of these lines we need a vector in order to define an intersecting plane. We want to orient the plane such that it contains the surface normal, but the surface normal direction will not be constant along the intersection curve. We therefore need to pick some vector as proxy for the surface normals to lie in the plane. We can, for example, find the surface normal at the endpoints of the line segment and compute a mean. If the line segment intersects the surface at more points than its ends, we can include these when computing the mean normal. Another possibility is to project (e.g. along the z-direction) the midpoint of the line segment onto the surface and use the normal at this point.

The second approach is to use a trace curve for each surface segment to obtain a set of intersecting planes. We can start from the trace curve of one surface segment and find sweep curves. For each of these we can, e.g. via bisection, find the parameter on the next trace curve, such that the sweep curves will meet at the boundary.

We now have a set of continuous sweep curves for the entire surface and the sweep curves are planar in each surface segment. Results of the two approaches can be seen in Figure 7.3.

7.3 An optimization using tangent planes

Since our new sweep curves are not planar, we can no longer use the optimization method from Section 3.4. We still want to approximate each curve segment separately, so instead of considering a piecewise planar sweep curve as one curve, we will consider each segment. A sweep curve thus consists of N planar curve segments $\mathbf{r}_i \colon [t_{i-1}, t_i] \to \mathbb{R}^3$, $i = 1, \dots, N$, which lie on the surface, such that

$$\mathbf{r}_i(t_i) = \mathbf{r}_{i+1}(t_{i+1}), \quad i = 1, \dots, N-1,$$

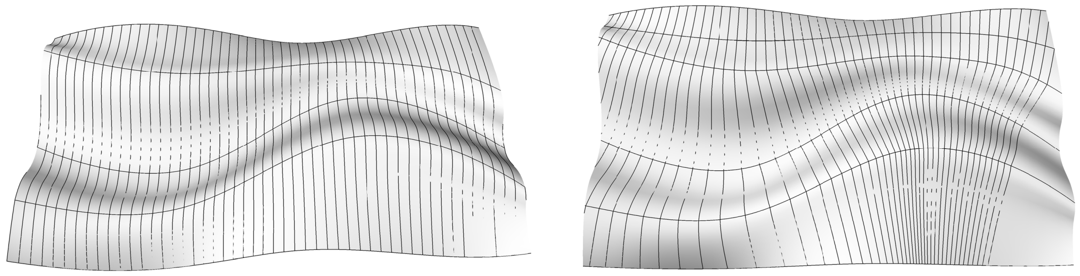


Figure 7.3: Left: Piecewise planar sweep curves obtained by drawing lines between the segmentation curves and using the mean of the surface normals at the lines' ends to define the intersecting planes. Right: Piecewise planar sweep curves based on a trace curve on each surface segment.

We want to find N planar elastic curve segments in $\mathbb{R}^3$ that approximate the sweep curve segments $\mathbf{r}_i$, which we can obtain by minimizing

$$\sum_{i=1}^{N} \int_{t_{i-1}}^{t_i} \left\| \boldsymbol{\gamma}_{\mathbf{p}_i} \left(\frac{s_i(t)}{L_i} \right) - \mathbf{r}_i(t) \right\|^2 s_i'(t)\, \mathrm{d}t$$

over the control parameter vectors $\mathbf{p}_i$, $i = 1, \ldots, N$, where s_i and L_i denote the arc length function and arc length for each sweep curve segment, i.e.

$$s_i(t) = \int_{t_{i-1}}^{t} \|\mathbf{r}_i'(\tau)\|\, \mathrm{d}\tau, \quad L_i = s_i(t_i), \quad \text{for } i = 1, \ldots, N.$$

Remark. The curves $\mathbf{r}_i$ and $\boldsymbol{\gamma}_{\mathbf{p}_i}$ are planar curves in $\mathbb{R}^3$, but we convert them into 2D-coordinates (in the relevant intersecting planes) before doing the optimization. The further constraints we impose on the problem, will be on the 3D curves, however.

We want the elastic curve segments to form a continuous curve in R^3, and when we approximate many sweep curves and use the obtained elastic curves to cut a surface, we want it to be tangent continuous between segments. We therefore impose constraints that the elastic curve segments should meet at the segment boundaries on the surface and that their end tangent should be in the surface's tangent plane at those points. Both of these constraints are very restrictive, so we will introduce some tolerances.

For the elastic curve segments to meet we must require $\boldsymbol{\gamma}_{\mathbf{p}_i}(1) = \boldsymbol{\gamma}_{\mathbf{p}_{i+1}}(0)$, for $i = 1, \ldots, N-1$. We also want the curve segments to meet close to the point on the surface, where the sweep curve segments $\mathbf{r}_i$ and $\mathbf{r}_{i+1}$ meet, that is at $\mathbf{r}_i(t_i)$.

For this purpose we introduce a unit vector $\mathbf{v}_i$ along which we allow the meeting point to vary. Unless the two curves segment $\mathbf{r}_i$ and $\mathbf{r}_{i+1}$ are coplanar, there is only one possible choice for this vector (up to a sign), namely the one that lies in the intersection between two planes. If the curves are coplanar, which is highly unlikely we let $\mathbf{v}_i$ be the cross product of the surface normal at $\mathbf{r}_i(t_i)$ and the normal to the plane of $\mathbf{r}_i$ and $\mathbf{r}_{i+1}$. The constraint we impose is thus

$$\boldsymbol{\gamma}_{\mathbf{p}_i}(1) = \mathbf{r}_i(t_i) + u_i \mathbf{v}_i = \boldsymbol{\gamma}_{\mathbf{p}_{i+1}}(0), \quad |u_i| < \varepsilon_0, \tag{7.1}$$

where u_i is a scalar, which is kept within a tolerance and determines how far the curves' meeting point is form the surface.

To ensure tangent continuity of the elastica-foliated surface, we want the end tangents of the elastic curve segments to lie in the tangent plane for the surface. The easiest way to ensure this is to demand that $\boldsymbol{\gamma}_{p_i}'(1) \cdot \mathbf{N}_i = \boldsymbol{\gamma}_{p_{i+1}}'(0) \cdot \mathbf{N}_i = 0$, where $\mathbf{N}_i$ is the surface normal at $\mathbf{r}_i(t_i)$.

Again we will allow some room for variation, so instead of using the actual surface normal, we will use an approximate normal vector $\mathbf{n}_i$ which will be kept close to the actual normal by a tolerance ε_1. We also introduce a tolerance ε_2 for how close the curves' tangents should be to being orthogonal to $\mathbf{n}_i$. Our constraint is thus

$$\cos(\frac{\pi}{180}(90+\varepsilon_2)) \leq \frac{\gamma'_{\mathbf{p}_i}(1)}{\|\gamma'_{\mathbf{p}_i}(1)\|} \cdot \mathbf{n}_i,\ \frac{\gamma'_{\mathbf{p}_{i+1}}(0)}{\|\gamma'_{\mathbf{p}_{i+1}}(0)\|} \cdot \mathbf{n}_i \ \leq \cos(\frac{\pi}{180}(90-\varepsilon_2)), \tag{7.2}$$

with the further demands that

$$\|\mathbf{n}_i\| = 1, \quad \mathbf{N}_i \cdot \mathbf{n}_i \geq \cos\left(\frac{\pi}{180}\varepsilon_1\right). \tag{7.3}$$

Summing up, by minimizing the function

$$\mathscr{G}(\mathbf{p}_1, \ldots, \mathbf{p}_N, u_1, \ldots, u_{N-1}, \mathbf{n}_1, \ldots, \mathbf{n}_{N-1}) = \sum_{i=1}^{N} \int_{t_{i-1}}^{t_i} \left\| \gamma_{\mathbf{p}_i}\left(\frac{s_i(t)}{L_i}\right) - \mathbf{r}_i(t) \right\|^2 s'_i(t)\,\mathrm{d}t$$

subject to the constraints (7.1)–(7.3), we obtain in each surface segment an approximating elastic curve segment with end points having a distance less than ε_0 to the surface and end tangents at most ε_2 degrees from an approximate tangent plane, which is itself at most ε_1 degrees off compared to the actual surface tangent plane.

Remark. From a mathematical perspective, the constraints defined above will in no way ensure tangent continuity of the surface foliated by the elastic curves. However, we only want precision within production tolerances. If we set $\varepsilon_0 = \varepsilon_1 = \varepsilon_2 = 0$ the elastic curve segments will form continuous, piecewise planar curves that have end points on the surface and end tangents in the surface's tangent planes at these points.

7.4 Results

We have tested the method described above on part of the curved wall; more precisely, on the four inner pieces of the six-piece segmentation used for the parallel planes approximation in Section 7.1. The result can be seen in Figure 7.4.

As we mentioned, some of the curves along the short dimension of the curved wall has areas of too high curvature. For our fabrication test, we therefore chose to use the short dimension as the cutting direction. We chose a simple segmentation in five parallel strips, using sweep curves based on parallel planes, see Figures 7.5 and 7.6.

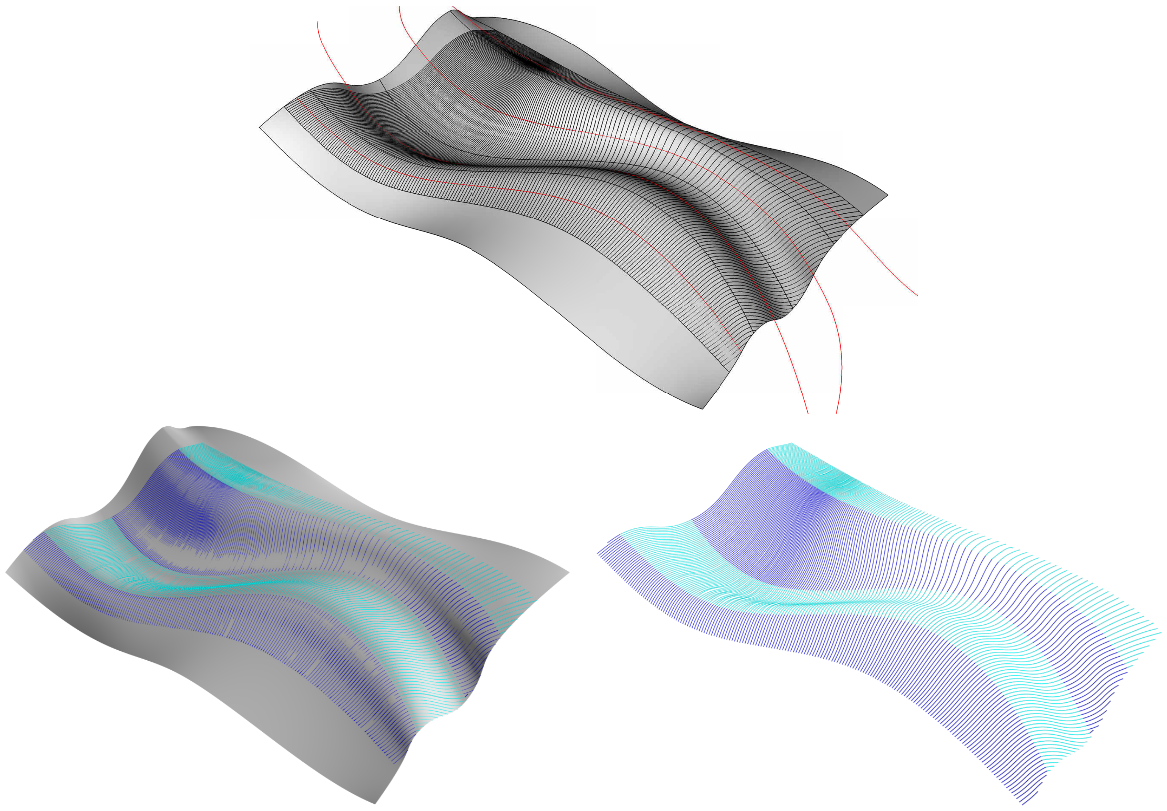


Figure 7.4: Top: 181 piecewise planar sweep curves based on a trace curve (red) in each of the four inner segments of the curved wall. Bottom left: The surface and the piecewise planar, piecewise elastic curves obtained from the optimization described in Section 7.3. Bottom right: Only the elastic curves.

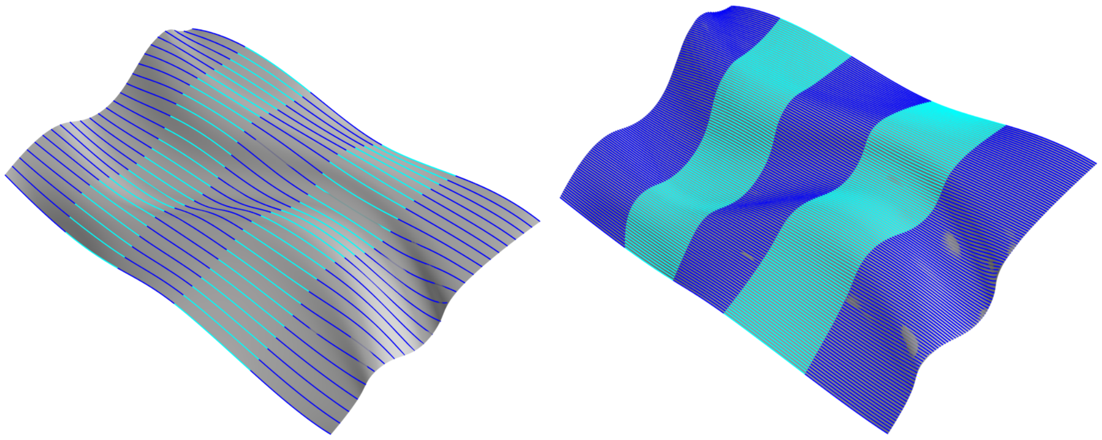


Figure 7.5: Rationalization of the curved wall. Left: 25 sweep curves in 5 segments each. The end points and end (unit) tangents of the elastic curves are constrained to be identical to those of the sweep curves. Right: 1201 sweep curves of 5 segments each. The elastic curves are constrained to meet with tangent continuity, but not at the points of the sweep curves.

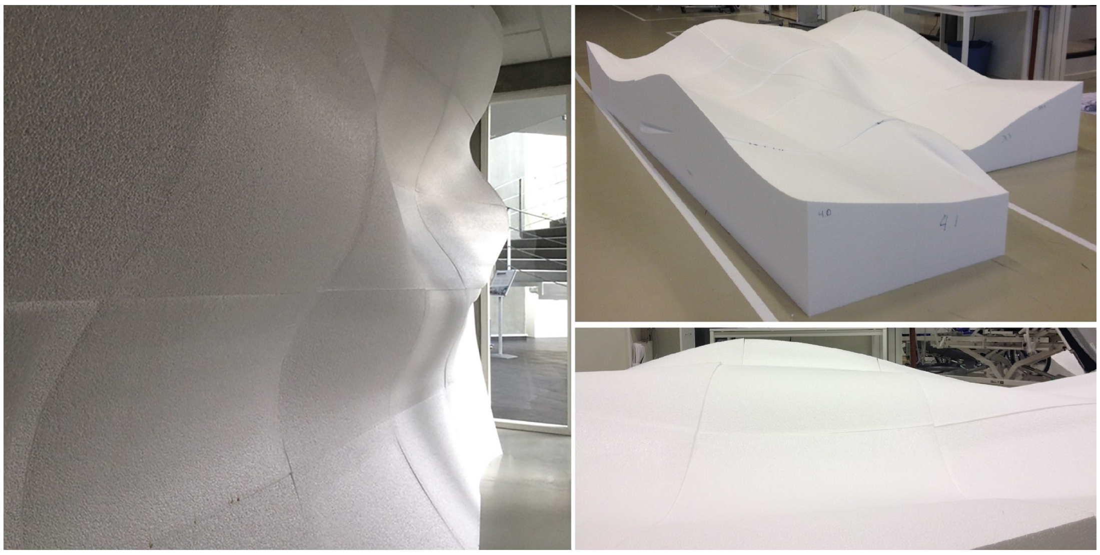


Figure 7.6: The produced result from cutting EPS blocks based on the rationalization in Figure 7.5 left.

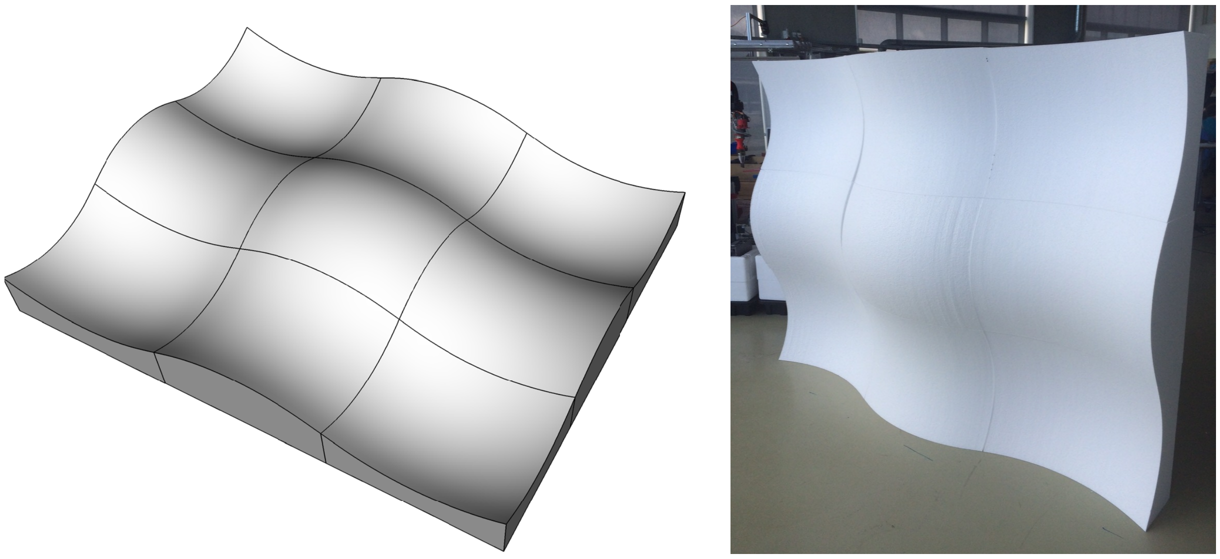


Figure 7.7: Surface design by 3XN and cut result of rationalized design. The surface was designed for hot blade cutting and segmented into blocks during the design phase. Tangent continuity for the cut result was obtained by constraining end points and tangents to be the same as those of the sweep curves.

Chapter 8

Finding the length of a symmetric elastic curve segment

In this chapter we deal with an interesting problem that came up during the course of the BladeRunner project. When we want the robots to bend the blade into a prescribed elastica shape, we provide the end points and tangents. These were computed based on the blade length, which means that if our estimate of this length is imprecise, the resulting elastica shape will not be identical to the one we have predicted. When we have a free blade, we can of course measure its length to a high precision, but when we mount it on the robots, part of the blade will be used for fixing it, and thus cannot deform elastically. Moreover, when the blade is heated, it will expand. We need to know the length of the part of the blade that can deform freely.

When the blade is mounted we can move its ends to given positions and turn them to given angles. Without loss of generality we may assume that the end points lie on a horizontal line. If we choose the same angle at both ends, we will get a symmetric shape and we can measure the curve height, see Figure 8.1. The symmetric curve will necessarily be a segment of a scaled version of the basic elastica $\boldsymbol{\zeta}_k$ for some k. The basic elastica has minimal y-value at $s = 0$, so we will here consider downward bending curves. Since we are considering a physical blade, we can assume that the obtained elastica segment constitutes less that a period of the entire (infinitely long) elastic curve. The question is now whether we can compute the blade length from measuring the height.

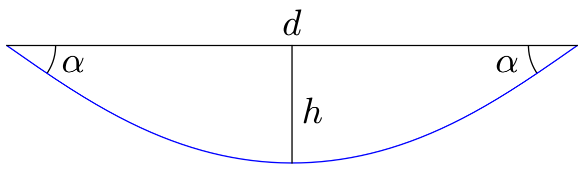


Figure 8.1: Downwards bending, symmetric elastic curve segment with end point distance d, end angles α and height h.

8.1 Free tangents

We will first consider the case where the blade is mounted in such a way that its ends can rotate freely. We saw in Section 2.1 that in this case the elastic curve segment will have inflection points at its ends.

Let $d > 0$ be the distance between the end points and let $h > 0$ be the measured height. The overall shape of the elastic curve is given by the unknown control parameter $k \in]0, 1[$, i.e. the elliptic modulus. What we do know is that the inflection points of this curve lie where the arc length parameter s is $nK(k)$, where n is odd. We may therefore take the relevant segment to be $s \in [-K, K]$, or equivalently, we may choose the control parameters $s_0 = -K$ and $\ell = 2K$. Letting (ζ_k^1, ζ_k^2) denote the coordinates of $\boldsymbol{\zeta}_k$, we can now compute in terms of k, the distance between the endpoints

$$d = S\left(\zeta_k^1(K) - \zeta_k^1(-K)\right) = S\left(2E(K,k) - K - (2E(-K,k) + K)\right) = 2S\left(2E(K,k) - K\right)$$

and the height

$$h = S\left(\zeta_k^2(K) - \zeta_k^2(0)\right) = 2kS(1 - \operatorname{cn}(K,k)) - 2kS(1 - \operatorname{cn}(0,k)) = 2kS.$$

We can determine the parameter k by solving

$$\frac{d}{h} = \frac{1}{k}\left(2E(K,k) - K\right) \tag{8.1}$$

Letting $f(k)$ denote the righthand side of the above equation, it can be shown that

$$f'(k) = -\frac{E(K,k)}{k^2 k'^2},$$

which is negative, so f is strictly decreasing and thus (8.1) has a unique solution, which can be found e.g. by bisection.

From k and h we can find the scaling parameter S, and the length is

$$L = 2SK = \frac{h}{k}K.$$

8.2 Fixed tangents

We now turn to the case where the tangents are fixed, which is the case for the robots used in our setup for hot blade cutting. We move the blade to a position with known end points and tangents, such that we get a symmetric downwards bending curve. Intuitively, if we increase the blade length, the height of the curve will also increase; if we shorten the blade, the height will decrease. We can thus assume that there is a one-to-one correspondence between the height and the length of curves in this specific configuration. It turns out, however, that its expression in terms of elliptic functions is quite complicated and hard to analyse mathematically.

Let $\alpha \in \,]0, \pi/2[$ be the angle formed by the end tangent and the line between the end points, and let $d > 0$ be the length of that line. Recall that the maximal tangential angle for the inflectional elastica with modulus k is $\theta_{\max}(k) = 2\arcsin(k)$. For the angle condition to be satisfied, our elastica must thus have $\theta_{\max}(k) > \alpha$, i.e. $k \geq \sin(\alpha/2)$. We will set $\beta = \sin(\alpha/2)$, so in the following β is a fixed value in the interval $\left]0, 1/\sqrt{2}\right[$. For any $k \geq \beta$ there are two segments of $\boldsymbol{\zeta}_k$ that satisfy the angle condition, see Figure 8.2.

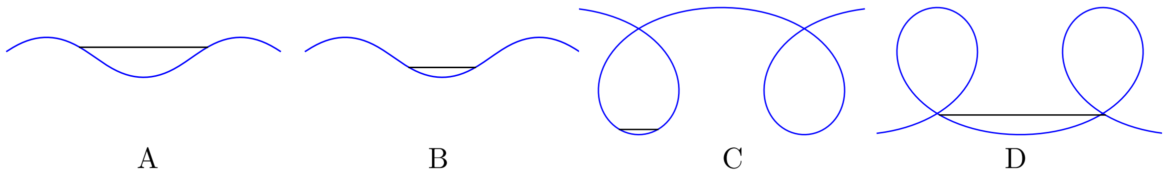


Figure 8.2: Elastica segments with interior angle $\alpha = \pi/6$ for $k = 0.3$ (cases A and B) and $k = 1.02$ (cases C and D).

We consider the following cases:

A: $k < 1$, $s \in [K, 2K]$, $\theta(s,k) = \alpha$.

B: $k \le 1$, $s \in [0, K]$, $\theta(s,k) = \alpha$.

C: $k \ge 1$, $s \in [0, K]$, $\theta(s,k) = \alpha$.

D: $k > 1$, $s \in [0, K]$, $\theta(s,k) = \pi - \alpha$.

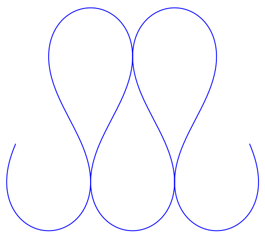


Figure 8.3: Elastica with $k = 0.8550924$.

Case A We first note that in order to avoid self-intersections, we must have k less than $k_{\text{crit}} \approx 0.8550924$, see Figure 8.3 and compare with Figure 2.1.

Let $k \in [\beta, k_{\text{crit}}[$ and $K = K(k)$. Letting $s_A \in [K, 2K]$ such that $\theta(s_A, k) = \alpha$, we consider the elastic curve segment with $s \in [-s_A, s_A]$ and distance d between its endpoints. The height of this segment is

$$h_A(k) = d\frac{\zeta_k^2(s_A) - \zeta_k^2(0)}{\zeta_k^1(s_A) - \zeta_k^1(-s_A)} = \frac{dk\,(1 - \operatorname{cn}(s_A, k))}{2E(s_A, k) - s_A}. \tag{8.2}$$

Case B We do not need to worry about self-intersections, so we let $k \in [\beta, 1]$, and take $s_B \in [0, K]$ such that $\theta(s_B, k) = \alpha$. The height h_B of the elastic curve segment with $s \in [-s_B, s_B]$ is given by (8.2) with B in place of A.

We note that for $k = \beta$ cases A and B coincide, since we get $s_A = s_B = \operatorname{sn}^{-1}(1, \beta) = K(\beta)$, and thus

$$h_A(\beta) = h_B(\beta) = \frac{d\beta}{2E(K(\beta), \beta) - K(\beta)}. \tag{8.3}$$

Case C Let $k \ge 1$ and $s_C \in [0, K]$ such that $\theta(s_C, k) = \alpha$. Again we recover (8.2) for the height with C in place of A. We note that for $k = 1$ cases B and C coincide, and we get $s_B = s_C = \operatorname{sn}^{-1}(\beta, 1) = \operatorname{arctanh}(\beta)$ and the height

$$h_B(1) = h_C(1) = \frac{d\left(1 - \sqrt{1 - \beta^2}\right)}{2\beta - \operatorname{arctanh}(\beta)}. \tag{8.4}$$

Case D Let $k > 1$ and $s_D \in [0, K]$ such that $\theta(s_D, k) = \pi - \alpha$ and consider the segment $s \in [s_D, 2K - s_D]$. This is an upwards curving elastica segment, but if we rotate it by π, we get a downwards curving segment that satisfies the desired angle condition. For simplicity we will do the calculations on the upwards curving elastica, and just make sure to get a positive height, which is thus

$$h_D(k) = d\frac{\zeta_k^2(K) - \zeta_k^2(s_D)}{\zeta_k^1(s_D) - \zeta_k^1(2K - s_D)} = \frac{dk\left(\operatorname{cn}(s_D, k) - \sqrt{1 - k^{-2}}\right)}{2E(s_D, k) - 2E(K, k) - s_D + K}$$

We observe that $h_D(k) \to 0$ as $k \to 1$, because as $k \to 1$, $K(k) \to \infty$, the numerator of $h_D(k)$ tends to $d\sqrt{1 - \beta^2}$ and the denominator minus K tends to $2\beta - 2 - \operatorname{arctanh}\beta$.

It is obvious that as $k \to \infty$, the segments of cases C and D will become identical circle segments, see Figure 8.4. The height will then be the sagitta of the circle with chord length d and angle α between the chord and the circle tangent. In other words we have

$$\lim_{k\to\infty} h_C(k) = \lim_{k\to\infty} h_D(k) = \tfrac{d}{2}\tan\tfrac{\alpha}{2} = \frac{d\beta}{2\sqrt{1 - \beta^2}}. \tag{8.5}$$

We have validated numerically (though we have not been able to prove analytically) that

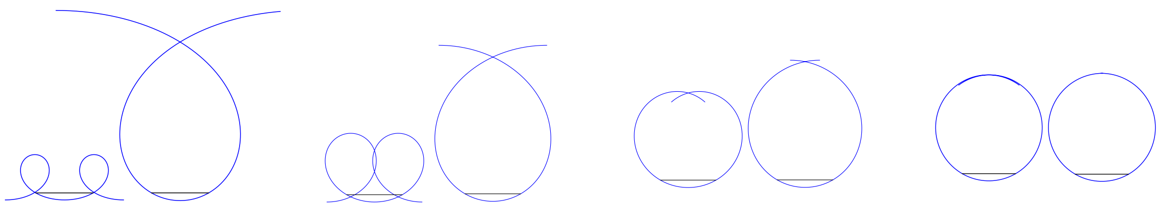


Figure 8.4: Cases C and D for $d = 1$, $\alpha = \pi/6$ and $k \in \{1.02, 1.1, 1.5, 5.0\}$. The cases coincide in the limit $k \to \infty$ when the elastica is a circle.

- h_A is strictly increasing on $[\beta, k_{\mathrm{crit}}[$,
- h_B is strictly decreasing on $[\beta, 1]$,
- h_C is strictly decreasing on $[1, \infty[$,
- h_D is strictly increasing on $]1, \infty[$.

In particular we have

$$h_D(k_1) < h_C(k_2) \leq h_B(k_3) \leq h_A(k_4)$$

for any choice of $k_1 \in]1, \infty[$, $k_2 \in [1, \infty[$, $k_3 \in [\beta, 1]$, and $k_4 \in [\beta, k_{\mathrm{crit}}[$, see Figures 8.5 and 8.6.

When α and d are given, we can compute values of (8.3), (8.4) and (8.5) corresponding to the heights of the elastica with free tangents, the one loop elastica (i.e. $k = 1$) and the circle segment, respectively. For any height h we can, by comparing with these values, determine whether h belongs to case A, B, C or D.

When we have determined the case, say A, we can find $k = h_A^{-1}(h)$. We do not have an analytic expression for h_A^{-1} but, since it is strictly monotone, we can find the k-value by the bisection method, for example. The length of the segment can now be found as

$$L = \frac{2ds_A}{\zeta_k^1(s_A) - \zeta_k^2(-s_A)} = \frac{ds_A}{2E(s_A, k) - s_A},$$

and similarly for cases B and C, while in case D we get

$$L = \frac{d(2K - 2s_D)}{\zeta_k^1(s_D) - \zeta_k^1(2K - s_D)} = \frac{d(K - s_D)}{2E(s_D, k) - 2E(K, k) - s_D + K}.$$

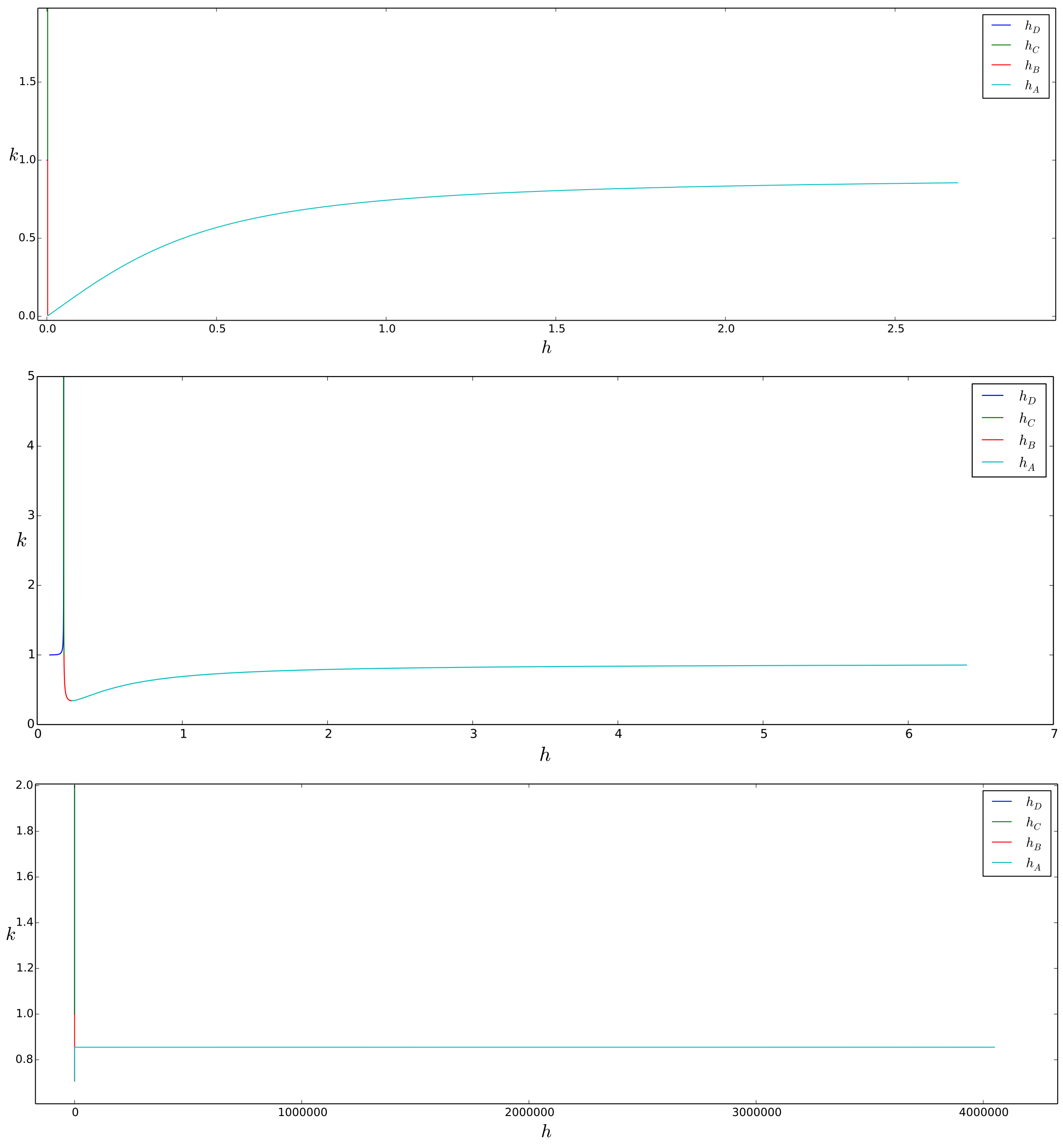


Figure 8.5: Graphs of the inverses of h_A, h_B, h_C and h_D. We have set $d = 1$ and from top to bottom we have $\alpha = 0.1$, $\alpha = 0.7$, $\alpha = 1.57$.

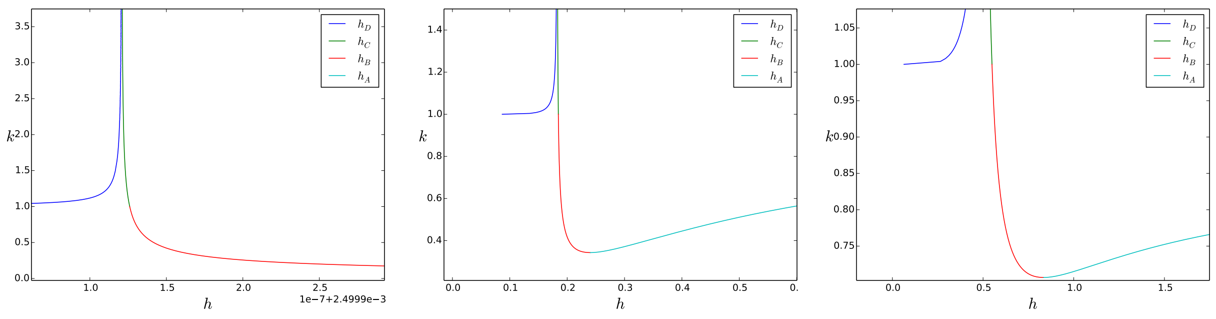


Figure 8.6: Close up of the areas in Figure 8.5 where h_B and h_C meet.

Chapter 9

Conclusion

We have described an optimization-based method for approximating arbitrary curves by elastic or piecewise elastic curves expressed in closed form using elliptic functions. We have developed a method for finding a canonical first guess for the optimization. In the case of piecewise elastic curves, we have imposed constraints on the optimization to ensure tangent continuity. We have described how to validate the success of the method by calculation of residuals.

For most examples, our curve approximation method performs very well, but an investigation into the effect of replacing the L^2 norm in the optimization by e.g. the H^1 or H^2 norm is in order. Using a Sobolev norm might make the algorithm more stable, but it might also increase computation time.

We have discussed reasons for subdividing a surface for approximation by elastica-foliated patches and proposed methods for choosing segmentation curves. Some of these methods are, however, computationally expensive. It would be desirable to find a fast method for determining whether a given surface can be approximated well by an elastica-foliated surface.

We have described how to approximate a set of planar or piecewise planar curves on a surface by piecewise elastic curves. For planar, piecewise elastic curves, we ensure tangent continuity within a given tolerance; for piecewise planar, piecewise elastic curves, we ensure that the end tangents lie in the tangent plane of the surface.

The methods that rely on curve approximation for planar curves perform well, while the method described in Section 7.3 is not completely stable (in its current implementation) and needs improvement. In both cases, doing optimization for many curves is time consuming, and the convergence rate naturally depends on the allowed tolerances. In order to determine the relevant tolerances, more hot blade cutting tests must be done with the physical production cell.

As we have mentioned, the continuity between segments is ensured only for the specific elastic curves that are obtained from optimization. The end point and tangent data is given to a set of robots which interpolate the data to perform the cutting motion. If we knew the robots' interpolation scheme (or, even better, if we could control it ourselves) we would be able to improve certain steps of the production.

Finally, we have described how to numerically determine the length of a symmetric elastic curve segment from its height and its end conditions. We can thus determine the length of a heated blade mounted in a robot cell, which is important when we wish to compute data for the hot blade cutting procedure. The one-to-one correspondence between length and height have only been established numerically. From a mathematical point of view, it would be more satisfying to prove that the length and the height depend monotonically on the elliptic parameter k.

Appendix A

Derivatives

In this appendix we shall find the derivatives with respect to k of the elliptic functions and the elliptic integrals. We follow Lawden [Law89], but we take specific care to obtain formulas for the derivatives at $k = 0$ and $k = 1$. We then list the first and second derivatives of the basic elastica as they appear in [BGN16], but here we have added the specific expressions for the derivatives at $k = 1$. These are important, since we want to be able to move from one family (inflectional or non-inflectional) to the other, when doing the optimizations described in Chapters 3 and 4. Finally, we list the derivatives of the angle function for the basic elastica.

Derivatives w.r.t k of the elliptic functions and elliptic integrals

Using the shorthand notation

$$\begin{aligned} S &= \operatorname{sn}(u,k), & C &= \operatorname{cn}(u,k), & D &= \operatorname{dn}(u,k), & E &= E(u,k) \\ S_k &= \frac{\partial}{\partial k}\operatorname{sn}(u,k), & C_k &= \frac{\partial}{\partial k}\operatorname{cn}(u,k), & D_k &= \frac{\partial}{\partial k}\operatorname{dn}(u,k), & E_k &= \frac{\partial}{\partial k}E(u,k) \end{aligned}$$

we differentiate (2.9) with respect to k and find

$$S_kS + C_kC = 0, \quad D_kD + kS^2 + k^2S_kS = 0. \tag{A.1}$$

Differentiating S_k with respect to u, and using the above we get

$$\frac{\partial}{\partial u}S_k = C_kD + D_kC = -\frac{S_kS}{C}D - \frac{kS^2 + k^2S_kS}{D}C = -\frac{S_kS}{CD}(D^2 + k^2C^2) - \frac{kS^2C}{D}, \tag{A.2}$$

and by rearranging and multiplying by $1/(CD)$, we obtain

$$\frac{\frac{\partial S_k}{\partial u}}{CD} + \frac{S_kS}{(CD)^2}(D^2 + k^2C^2) = -\frac{kS^2}{D^2}.$$

We note that the left-hand side is equal to $\frac{\partial}{\partial u}(S_k/(CD))$, so we integrate and get

$$\frac{S_k}{CD} = \frac{u}{k} + \frac{k}{k'^2}\frac{SC}{D} - \frac{1}{kk'^2}E + f(k)$$

which may be verified by differentiation and use of (2.9). Here the function f is the “constant” of integration, but this turns out to be identically zero: for $u = 0$ we get

$$f(k) = S_k\big|_{u=0} = \tfrac{\partial}{\partial k}\operatorname{sn}(0,k) = 0,$$

since $\operatorname{sn}(0,k) = 0$ for all k.

Finally, we arrive at

$$S_k = \frac{u}{k}CD + \frac{k}{k'^2}SC^2 - \frac{1}{kk'^2}ECD,$$

and we can find C_k and D_k from (A.1)

$$\begin{aligned}C_k &= -\frac{u}{k}SD - \frac{k}{k'^2}S^2C + \frac{1}{kk'^2}ESD,\\ D_k &= \frac{k}{k'^2}ESC - \frac{k}{k'^2}S^2D - ukSC\end{aligned}$$

We observe that the expressions we have found for the k-derivatives are not valid for $k \in \{0,1\}$. Setting $k=1$ in (A.2) we get

$$\tfrac{\partial S_k}{\partial u}\big|_{k=1} = -2\tanh u\, S_k|_{k=1} - \tanh^2 u,$$

which is a linear first order equation with the solution

$$S_k|_{k=1} = -\tfrac{1}{2}\left(\sinh u\cosh u - u\right)\operatorname{sech}^2 u,$$

where, as before, the constant of integration is zero. For $k=0$ we get the homogeneous equation $\partial S_k/\partial u = -\tan(u)S_k$ and since, again, the constant of integration is zero, we find $S_k|_{k=0} = 0$.

As above we can use (A.1) to find C_k and D_k, which are

$$\begin{aligned}C_k|_{k=0} &= 0, & C_k|_{k=1} &= \tfrac{1}{2}\left(\sinh u\cosh u - u\right)\tanh u \operatorname{sech} u,\\ D_k|_{k=0} &= 0, & D_k|_{k=1} &= -\tfrac{1}{2}\left(\sinh u\cosh u + u\right)\tanh u \operatorname{sech} u.\end{aligned}$$

The k-derivative of E and K can be found by differentiating under the integral

$$\begin{aligned}E_k &= \int_0^u \frac{\partial}{\partial k}\left(D^2\right)\mathrm{d}t = 2\int_0^u D_k D\,\mathrm{d}t\\ \frac{\mathrm{d}K}{\mathrm{d}k} &= \int_0^{\frac{\pi}{2}} \frac{\partial}{\partial k}\left(\frac{1}{\sqrt{1-k^2\sin^2 t}}\right)\mathrm{d}t = \int_0^{\frac{\pi}{2}} \frac{k\sin^2 t}{\left(1-k^2\sin^2 t\right)^{3/2}}\,\mathrm{d}t\end{aligned}$$

These are easy to compute for $k \in \{0,1\}$ and we refer to [Law89] for the details of deriving the general formulas

$$\begin{aligned}E_k|_{k=0} &= 2\int_0^u 0\,\mathrm{d}t = 0,\\ E_k|_{k=1} &= -\int_0^u \left(\sinh t\cosh t + t\right)\tanh t \operatorname{sech}^2 t\,\mathrm{d}t = -\tfrac{1}{2}\left(u + u\tanh^2 u - \tanh u\right),\\ E_k &= \frac{k}{k'^2}SCD - ukS^2 - \frac{k}{k'^2}EC^2, \qquad k \notin \{0,1\},\\ \tfrac{\mathrm{d}K}{\mathrm{d}k}\big|_{k=0} &= \int_0^{\frac{\pi}{2}} 0\,\mathrm{d}t = 0,\\ \tfrac{\mathrm{d}K}{\mathrm{d}k}\big|_{k=1} &= \int_0^{\frac{\pi}{2}} \frac{\sin^2 t}{|\cos^3 t|}\,\mathrm{d}t = \infty,\\ \tfrac{\mathrm{d}K}{\mathrm{d}k} &= \frac{E(K(k),k)}{kk'^2} - \frac{K(k)}{k}, \qquad k \notin \{0,1\}.\end{aligned}$$

It is no surprise that the last integral diverges, since $K \to \infty$ as $k \to 1$. We remark that the general expressions found in this section, are indeed valid for all $k \in \mathbb{R}^+ \setminus \{0,1\}$.

The second derivatives with respect to k can be found from straightforward (though tedious) calculations using the above formulas. For $k \in \{0,1\}$, analogously to the above, one can differentiate $\partial S_k/\partial u$ and (A.1) to obtain a linear equation for $\partial S_k/\partial k$.

Derivatives of the parameterized elastica

We now consider the basic elastica $\boldsymbol{\zeta}$ defined in Section 2.3 and we will use the shorthand notation

$$S = \operatorname{sn}(s,k),\ C = \operatorname{cn}(s,k),\ D = \operatorname{dn}(s,k),\ E = E(s,k).$$

We have

$$\boldsymbol{\zeta}(s,k) = \begin{pmatrix} 2E - s \\ 2k(1-C) \end{pmatrix}.$$

The derivatives with respect to s follow from the definitions of the elliptic functions.

$$\begin{aligned}
\frac{\partial}{\partial s}\boldsymbol{\zeta}(s,k) &= \begin{pmatrix} 2D^2 - 1 \\ 2kSD \end{pmatrix}, \\
\frac{\partial^2}{\partial s^2}\boldsymbol{\zeta}(s,k) &= 2kC \begin{pmatrix} -2kSD \\ D^2 - k^2S^2 \end{pmatrix} = 2kC \begin{pmatrix} -2kSD \\ 2D^2 - 1 \end{pmatrix}.
\end{aligned}$$

We have found the k-derivatives of S, C, D and E above and from these, with repeated use of (2.9), one can find

$$\begin{aligned}
\frac{\partial}{\partial k}\boldsymbol{\zeta}(s,k) =& \frac{2}{k'^2} \begin{pmatrix} k\left(SCD - EC^2 - sk'^2S^2\right) \\ k'^2 + C(k^2 - D^2) - SD(E - sk'^2) \end{pmatrix}, \\
\frac{\partial^2}{\partial s \partial k}\boldsymbol{\zeta}(s,k) =& \frac{2}{k'^2}\left(SD - C(E - sk'^2)\right) \begin{pmatrix} -2kSD \\ 2D^2 - 1 \end{pmatrix}, \\
\frac{\partial^2}{\partial k^2}\boldsymbol{\zeta}(s,k) =& \frac{2}{k'^4} \begin{pmatrix} 2SDC\left(D^2 - k^2E^2 + k'^2\left(s^2k^2 - (E-s)^2 - \frac{1}{2}\right)\right) \\ \frac{1}{k}\left((1 - 2k^2S^2)(E-s)(2sk^2 + E - s)C + DS(sk'^2 - E)(4k^2C^2 + k'^2)\right) \end{pmatrix} \\
&+ \frac{2}{k'^4} \begin{pmatrix} (E-s)(C^2 + D^2 - 4C^2D^2) + 2sk^2(2S^2 - 1)D^2 - sk'^2 \\ -skk'^2DS + s^2k^3C + kCS^2(2 - 2s^2k^4 - 2k^2S^2 + k^2) \end{pmatrix}
\end{aligned}$$

Again, we find expressions that are not valid for $k = 1$. However, the relevant values can easily be found from those for the elliptic functions. We have

$$\begin{aligned}
\frac{\partial}{\partial k}\boldsymbol{\zeta}(s,1) &= \left.\begin{pmatrix} 2\frac{\partial E}{\partial k} \\ 2(1-C) - 2k\frac{\partial C}{\partial k} \end{pmatrix}\right|_{k=1} = \begin{pmatrix} \tanh s - s\tanh^2 s - s \\ 2\left(1 - \operatorname{sech} s\right) - \left(\sinh s \cosh s - s\right)\tanh s \operatorname{sech} s \end{pmatrix} \\
\frac{\partial^2}{\partial s \partial k}\boldsymbol{\zeta}(s,1) &= \begin{pmatrix} 4D\frac{\partial D}{\partial k} \\ 2SD + 2kD\frac{\partial S}{\partial k} + 2kS\frac{\partial D}{\partial k} \end{pmatrix} = \begin{pmatrix} -2\left(\sinh s \cosh s + s\right)\tanh s \operatorname{sech}^2 s \\ 2s\operatorname{sech}^3 s + 2\tanh s \operatorname{sech} s - \sinh s - s\operatorname{sech} s \end{pmatrix} \\
\frac{\partial^2}{\partial k^2}\boldsymbol{\zeta}(s,1) &= \left.\begin{pmatrix} 2\frac{\partial^2 E}{\partial k^2} \\ -4\frac{\partial C}{\partial k} - 2k\frac{\partial^2 C}{\partial k^2} \end{pmatrix}\right|_{k=1} \\
&= \frac{1}{4}\begin{pmatrix} 2\sinh s\cosh s - 4s - \tanh s - 4s^2\tanh s \operatorname{sech}^2 s + 3s\operatorname{sech} s \\ -4s\sinh s - \cosh s + (1 - 2s^2)\operatorname{sech} s + 3s\tanh s \operatorname{sech} s + 4s^2\operatorname{sech}^3 s \end{pmatrix}
\end{aligned}$$

The derivatives of $\boldsymbol{\gamma}_{(k,s_0,\ell,S,\phi,\hat{x},\hat{y})}$ with respect to the control parameters can be found by straightforward calculations using the above.

Derivatives of the angle function

Recall that the angle function for the basic elastica is

$$\theta(s,k) = \begin{cases} 2\arcsin\left(k\,\mathrm{sn}(s,k)\right), & k \le 1, \\ 2\,\mathrm{am}\left(ks, \frac{1}{k}\right), & k > 1. \end{cases}$$

We have not defined an extension of am for $k > 1$, and we will not need this, but we will need the k-derivative of am.

Proposition A.1. *The k-derivative of* am *is*

$$\frac{\partial\,\mathrm{am}}{\partial k}(s,k) = \frac{s}{k}\,\mathrm{dn}(s,k) + \frac{k}{k'^2}\,\mathrm{sn}(s,k)\,\mathrm{cn}(s,k) - \frac{1}{kk'^2}E(s,k)\,\mathrm{dn}(s,k).$$

We postpone the proof since our main objective is finding the derivatives of θ.

For $k < 1$ we immediately find the derivatives

$$\begin{aligned} \frac{\partial}{\partial s}\theta(s,k) &= 2k\,\mathrm{cn}(s,k) \\ \frac{\partial}{\partial k}\theta(s,k) &= 2\frac{\mathrm{sn}(s,k) + k\frac{\partial\,\mathrm{sn}}{\partial k}(s,k)}{\mathrm{dn}(s,k)} = \frac{2}{k'^2}\left(\mathrm{sn}(s,k)\,\mathrm{dn}(s,k) + \mathrm{cn}(s,k)\left(sk'^2 - E(s,k)\right)\right), \end{aligned}$$

from which we also get

$$\begin{aligned} \frac{\partial}{\partial s}\theta(s,1) &= 2\,\mathrm{sech}\,s \\ \frac{\partial}{\partial k}\theta(s,1) &= 2\frac{\mathrm{sn}(s,1) + \frac{\partial\,\mathrm{sn}}{\partial k}(s,1)}{\mathrm{dn}(s,1)} = \sinh s + s\,\mathrm{sech}\,s \end{aligned}$$

For $k > 1$ we have

$$\frac{\partial}{\partial s}\theta(s,k) = 2k\,\mathrm{dn}\left(ks, \tfrac{1}{k}\right) = 2k\,\mathrm{cn}(s,k),$$

where we have used the analytic extension of dn, and we observe that this is the same expression as for $k < 1$. For the k-derivative, we have

$$\begin{aligned} \frac{\partial\theta}{\partial k}(s,k) &= 2s\,\mathrm{dn}\left(ks,\tfrac{1}{k}\right) - \tfrac{2}{k^2}\tfrac{\partial\,\mathrm{am}}{\partial k}\left(ks,\tfrac{1}{k}\right) \\ &= 2s\,\mathrm{dn}\left(ks,\tfrac{1}{k}\right) - \tfrac{2}{k^2}\left(k^2 s\,\mathrm{dn}\left(ks,\tfrac{1}{k}\right) - \frac{\mathrm{sn}\left(ks,\tfrac{1}{k}\right)\mathrm{cn}\left(ks,\tfrac{1}{k}\right)}{k(1-k^{-2})} + \frac{kE\left(ks,\tfrac{1}{k}\right)\mathrm{dn}\left(ks,\tfrac{1}{k}\right)}{1-k^{-2}}\right) \\ &= \frac{2}{k(k^2-1)}\,\mathrm{sn}\left(ks,\tfrac{1}{k}\right)\mathrm{cn}\left(ks,\tfrac{1}{k}\right) - \frac{2k}{k^2-1}E\left(ks,\tfrac{1}{k}\right)\mathrm{dn}\left(ks,\tfrac{1}{k}\right) \\ &= \frac{2}{k'^2}\left(\mathrm{sn}(s,k)\,\mathrm{dn}(s,k) - (E(s,k) - sk'^2)\,\mathrm{cn}(s,k)\right), \end{aligned}$$

and again we get the same expression as for $k < 1$.

The derivatives of the general angle function $\theta_{(k,s_0,\ell,\phi)}$ with respect to the control parameters can be found using the above.

Proof of Proposition A.1. For $k_0 \in]0,1[$ set $K_0 = K(k_0)$ and let $s \in [-K_0, K_0]$. We have

$$\mathrm{am}(s,k_0) = \arcsin\left(\mathrm{sn}(s,k_0)\right),$$

so

$$\begin{aligned}\frac{\partial\,\mathrm{am}}{\partial k}(s,k_0) &= \frac{\frac{\partial\,\mathrm{sn}}{\partial k}(s,k_0)}{\sqrt{1-\mathrm{sn}^2(s,k_0)}} = \frac{\frac{\partial\,\mathrm{sn}}{\partial k}(s,k_0)}{\mathrm{cn}(s,k_0)} \\ &= \frac{s}{k_0}\,\mathrm{dn}(s,k_0) + \frac{k_0}{k_0'^2}\,\mathrm{sn}(s,k_0)\,\mathrm{cn}(s,k_0) - \frac{1}{k_0k_0'^2}E(s,k_0)\,\mathrm{dn}(s,k_0),\end{aligned}$$

and we shall argue that this formula also holds if $s \notin [-K_0, K_0]$.

Let $s \in \mathbb{R}$ and pick $t \in [-K_0, K_0]$ and $n \in \mathbb{Z}$ such that $s = t + 2nK_0$. Let A denote the function $k \mapsto \mathrm{am}(t + 2nK(k), k)$. On the one hand we have

$$\begin{aligned}\frac{\mathrm{d}A}{\mathrm{d}k} &= \frac{\partial\,\mathrm{am}}{\partial u}(t+2nK(k),k)\frac{\mathrm{d}}{\mathrm{d}k}\,(t+2nK(k)) + \frac{\partial\,\mathrm{am}}{\partial k}(t+2nK(k),k) \\ &= 2n\,\mathrm{dn}(t+2nK(k),k)\frac{\mathrm{d}K}{\mathrm{d}k} + \frac{\partial\,\mathrm{am}}{\partial k}(t+2nK(k),k).\end{aligned}$$

Recalling from Section 2.2 that $\mathrm{am}(u + 2nK(k), k) = \mathrm{am}(u, k) + n\pi$, we find on the other hand that

$$\frac{\mathrm{d}A}{\mathrm{d}k} = \frac{\partial\,\mathrm{am}}{\partial k}(t,k),$$

so

$$\frac{\partial\,\mathrm{am}}{\partial k}(t+2nK(k),k) = \frac{\partial\,\mathrm{am}}{\partial k}(t,k) - 2n\,\mathrm{dn}(t+2nK(k),k)\frac{\mathrm{d}K}{\mathrm{d}k}.$$

In particular, for $k = k_0$ we get

$$\frac{\partial\,\mathrm{am}}{\partial k}(s,k_0) = \frac{\partial\,\mathrm{am}}{\partial k}(t,k_0) - 2n\,\mathrm{dn}(s,k_0)\frac{\mathrm{d}K}{\mathrm{d}k}(k_0)$$

Now, since $t \in [-K_0, K_0]$, we have

$$\begin{aligned}\frac{\partial\,\mathrm{am}}{\partial k}(t,k_0) =& \frac{t}{k_0}\,\mathrm{dn}(t,k_0) + \frac{k_0}{k_0'^2}\,\mathrm{sn}(t,k_0)\,\mathrm{cn}(t,k_0) - \frac{1}{k_0k_0'^2}E(t,k_0)\,\mathrm{dn}(t,k_0) \\ =& \frac{s-2nK_0}{k_0}\,\mathrm{dn}(s,k_0) + \frac{k_0}{k_0'^2}\,\mathrm{sn}(s,k_0)\,\mathrm{cn}(s,k_0) - \frac{\mathrm{dn}(s,k_0)}{k_0k_0'^2}E(s-2nK_0,k_0) \\ =& \frac{s}{k_0}\,\mathrm{dn}(s,k_0) + \frac{k_0}{k_0'^2}\,\mathrm{sn}(s,k_0)\,\mathrm{cn}(s,k_0) - \frac{\mathrm{dn}(s,k_0)}{k_0k_0'^2}E(s,k_0) \\ &+ \frac{2n}{k_0}\left(\frac{E(K_0,k_0)}{k_0'^2} - K_0\right)\mathrm{dn}(s,k_0),\end{aligned}$$

where we have used that $E(u - 2nK) = E(u) - 2nE(K)$, and since the last term is exactly $2n\,\mathrm{dn}(s,k_0)\frac{\mathrm{d}K}{\mathrm{d}k}\big|_{k=k_0}$, we conclude that

$$\frac{\partial\,\mathrm{am}}{\partial k}(s,k_0) = \frac{s}{k_0}\,\mathrm{dn}(s,k_0) + \frac{k_0}{k_0'^2}\,\mathrm{sn}(s,k_0)\,\mathrm{cn}(s,k_0) - \frac{1}{k_0k_0'^2}E(s,k_0)\,\mathrm{dn}(s,k_0).$$

□

Appendix B

Proof of the addition formulas

Here we prove the addition formulas for the elliptic functions as stated in Section 2.2. The modulus k is fixed. We write S_u, C_u, D_u for $\mathrm{sn}(u)$, $\mathrm{cn}(u)$, $\mathrm{dn}(u)$, respectively, and similarly for v, so the addition formulas are

$$\begin{aligned} \mathrm{sn}(u+v) &= \frac{S_uC_vD_v + S_vC_uD_u}{1-k^2S_u^2S_v^2} \\ \mathrm{cn}(u+v) &= \frac{C_uC_v - S_uS_vD_uD_v}{1-k^2S_u^2S_v^2} \\ \mathrm{dn}(u+v) &= \frac{D_uD_v - k^2S_uS_vC_uC_v}{1-k^2S_u^2S_v^2}. \end{aligned} \tag{B.1}$$

Denoting the right-hand sides of (B.1) by R_S, R_C, R_D, we have

$$\begin{aligned} \frac{\mathrm{d}R_S}{\mathrm{d}u} =& \frac{C_uD_uC_vD_v - S_vS_uD_u^2 - k^2S_vS_uC_u^2}{1-k^2S_u^2S_v^2} + \frac{2k^2S_uC_uD_uS_v^2(S_uC_vD_v + S_vC_uD_u)}{(1-k^2S_u^2S_v^2)^2} \\ =& \frac{k^2C_uC_vD_uD_vS_u^2S_v^2 + C_uC_vD_uD_v}{(1-k^2S_u^2S_v^2)^2} \\ &+ \frac{k^2D_u^2S_u^3S_v^3 - D_u^2S_uS_v + 2k^2C_u^2D_u^2S_uS_v^3 - k^2C_u^2S_uS_v + k^4C_u^2S_u^3S_v^3}{(1-k^2S_u^2S_v^2)^2} \\ =& \frac{k^2C_uC_vD_uD_vS_u^2S_v^2 + C_uC_vD_uD_v}{(1-k^2S_u^2S_v^2)^2} \\ &+ \frac{D_u^2S_uS_v\left(k^2S_u^2S_v^2 - 1 + k^2C_u^2S_v^2\right) + k^2C_u^2S_uS_v\left(D_u^2S_v^2 - 1 + k^2S_u^2S_v^2\right)}{(1-k^2S_u^2S_v^2)^2}. \end{aligned}$$

By use of (2.9) we find

$$\begin{aligned} k^2S_u^2S_v^2 - 1 + k^2C_u^2S_v^2 &= k^2S_v^2 - 1 = -D_v^2, \\ D_u^2S_v^2 - 1 + k^2S_u^2S_v^2 &= S_v^2 - 1 = -C_v^2, \end{aligned}$$

so we get

$$\frac{\mathrm{d}R_S}{\mathrm{d}u} = \frac{k^2C_uC_vD_uD_vS_u^2S_v^2 + C_uC_vD_uD_v - D_u^2D_v^2S_uS_v - k^2C_u^2C_v^2S_uS_v}{(1-k^2S_u^2S_v^2)^2}.$$

We now note that this is equal to R_CR_D. Similarly one can show that

$$\frac{\mathrm{d}R_C}{\mathrm{d}u} = -R_SR_D, \quad \frac{\mathrm{d}R_D}{\mathrm{d}u} = -k^2R_SR_C.$$

From the definition of the elliptic functions we have

$$\begin{aligned}
\frac{\mathrm{d}\,\mathrm{sn}(u+v)}{\mathrm{d}u} &= \mathrm{cn}(u+v)\,\mathrm{dn}(u+v), \\
\frac{\mathrm{d}\,\mathrm{cn}(u+v)}{\mathrm{d}u} &= -\,\mathrm{sn}(u+v)\,\mathrm{dn}(u+v), \\
\frac{\mathrm{d}\,\mathrm{dn}(u+v)}{\mathrm{d}u} &= -k^2\,\mathrm{sn}(u+v)\,\mathrm{cn}(u+v),
\end{aligned}$$

so the left-hand sides of (B.1) satisfy the same system of differential equations (as functions of u) as the right-hand sides. If the two sets of functions agree at a point, they must be the same. We compute the right-hand sides for $u = 0$, recalling that $S_u(0) = 0$, $C_u(0) = D_u(0) = 1$, so we get

$$R_S|_{u=0} = S_v, \quad R_S|_{u=0} = C_v, \quad R_S|_{u=0} = D_v,$$

and observe that these are the same as the left-hand sides for $u = 0$, and we are done. $\square$

Appendix C

SciPy bug report

The discovery of errors in SciPy's elliptic functions led me to post the following bug report on GitHub[1] on 24 August, 2014. Within a few hours another user confirmed the bug, but there have been no further comments.

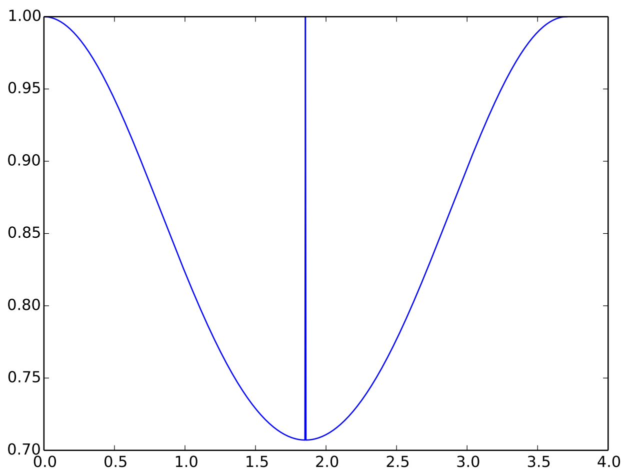
scipy.special.ellipj dn wrong values at quarter period

I'm using Numpy version 1.8.1, SciPy version 0.14.0. At the quarter period of the elliptic functions I get a wrong value in the third output of `ellipj` (third output is `dn`):

```
from scipy.special import ellipj
from scipy.special import ellipk
ellipj(ellipk(0.5),0.5)[2]
    1.0
```

This results is a discontinuity for `dn`.

```
import matplotlib.pyplot as plt
import numpy as np
A=np.linspace(0,2*ellipk(0.5),1001)
plt.plot(A,ellipj(A,0.5)[2])
```



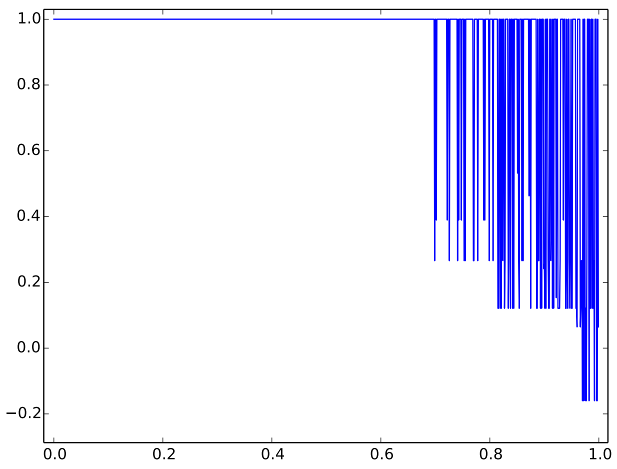
Using that `dn^2=1-m*sn^2`, I find that the correct value is

[1]`https://github.com/scipy/scipy/issues/3904`

```
np.sqrt(1-0.5*ellipj(ellipk(0.5),0.5)[0]**2)
    0.70710678118654757
```

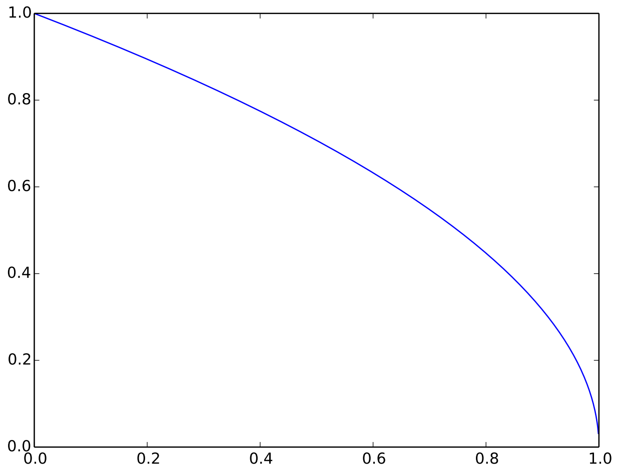
For m-values below 0.6 the output is apparently always 1.0, but for larger m-values it is not so (the values are still wrong though).

```
L=np.linspace(0,1,1000)
plt.plot(L,ellipj(ellipk(L),L)[2])
```



The correct graph would be

```
plt.plot(L,np.sqrt(1-L*ellipj(ellipk(L),L)[0]**2))
```



Bibliography

[AS72] Milton Abramowitz and Irene A. Stegun. *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical tables.* Dover Publications, Inc., New York, ninth Dover printing, tenth GPO printing edition, 1972.

[BBC$^+$16] D. Brander, J. A. Bærentzen, K. Clausen, A. Fisker, J. Gravesen, M. N. Lund, T. B. Nørbjerg, K. Steenstrup, and A. Søndergaard. Designing for robotic hot-blade cutting. In *Advances in Architectural Geometry 2016*, 2016. To appear.

[BBE$^+$15] D. Brander, A. Bærentzen, A. Evgrafov, J. Gravesen, S. Markvorsen, T. B. Nørbjerg, P. Nørtoft, and K. Steenstrup. Hot blade cuttings for the building industry. To appear. Preprint available on ResearchGate, 2015.

[BdB65] Garrett Birkhoff and Carl R. de Boor. Piecewise polynomial interpolation and approximation. In Henry L. Garabedian, editor, *Approximation of Functions (Proc. Sympos. General Motors Res. Lab., 1964)*, pages 164–190. Elsevier Publ. Co., Amsterdam, 1965.

[BGN16] D. Brander, J. Gravesen, and T. B. Nørbjerg. Approximation by planar elastic curves. *Advances in Computational Mathematics*, 2016. To appear. DOI: 10.1007/s10444-016-9474-z.

[BHN01] Alfred M. Bruckstein, Robert J. Holt, and Arun N. Netravali. Discrete elastica. *Appl. Anal.*, 78(3-4):453–485, 2001.

[Bis75] Richard L. Bishop. There is more than one way to frame a curve. *Amer. Math. Monthly*, 82:246–251, 1975.

[BNR01] A. M. Bruckstein, A. N. Netravali, and T. J. Richardson. Epi-convergence of discrete elastica. *Appl. Anal.*, 79(1-2):137–171, 2001.

[Bow61] F. Bowman. *Introduction to Elliptic Functions with Applications.* Dover Publications, Inc., New York, 1961.

[Cay62] A. Cayley. On the transcendent $\mathrm{gd}.u = \frac{1}{i}\log\tan\left(\frac{1}{4}\pi + \frac{1}{2}ui\right)$. *Philosophical Magazine*, 24(158):19–21, 1862. Fourth series.

[Die60] J. Dieudonné. *Foundations of Modern Analysis.* Academic Press Inc., New York and London, 1960.

[dSKBH02] Bram de Smit, Adrie Kooijman, Johan J. Broek, and Imre Horváth. Developing a tool for the direct cutting of freeform surfaces out of extruded polystyrene foam. In *Euro-u Rapid 2002 Conference*, 2002.

[dSMB$^+$00] Bram de Smit, Anthony J. Medland, Johan J. Broek, Imre Horváth, and Alex F. Lennings. Comparative analysis and experimental verification of the computed shape of a flexible blade tool for ff-tlom. In *Proceedings of DETC 2000*, 2000.

[Eul44] L. Euler. *Methodus inveniendi lineas curvas maximi minimive proprietate gaudentes*, chapter Additamentum I: de curvis elasticis, pages 245–310. Lausanne & Geneva, M.-M. Bousquet, 1744. Translation: [OEB33].

[FP10] Simon Flöry and Helmut Pottmann. Ruled surfaces for rationalization and design in architecture. In *LIFE in:formation. On Responsive Information and Variations in Architecture*, pages 103–109. Association for Computer Aided Design in Architecture (ACADIA), 2010. Proc. ACADIA 2010.

[Gre59] Alfred George Greenhill. *The Applications og Elliptic Functions.* Dover Publications, inc., New York, 1959.

[Jac29] K. G. J. Jacobi. *Fundamenta nova theoriae functionum ellipticarum.* Sumptibus fratrum Borntræger, Regiomonti, 1829.

[Law89] Derek F. Lawden. *Elliptic functions and applications*, volume 80 of *Applied Mathematical Sciences.* Springer-Verlag, New York, 1989.

[Lev09] R. Levien. *From Spiral to Spline: Optimal Techniques for Interactive Curve Design.* PhD thesis, UC Berkeley, 2009.

[Lov20] A. E. H. Love. *A treatise on the Mathematical Theory of Elasticity.* Cambridge University Press, Cambridge, 1920. Third Ed.

[Mal77] Michael A. Malcolm. On the computation of nonlinear spline functions. *SIAM J. Numer. Anal.*, 14(2):254–282, 1977.

[Meh74] Even Mehlum. Nonlinear splines. In *Computer aided geometric design (Proc. Conf., Univ. Utah, Salt Lake City, Utah, 1974)*, pages 173–207. Academic Press, New York, 1974. With an appendix by W. W. Meyer.

[Mey01] Kenneth R. Meyer. Jacobi elliptic functions from a dynamical systems point of view. *Amer. Math. Monthly*, 108(8):729–737, 2001.

[MFS13] Wes McGee, Jelle Feringa, and Asbjørn Søndergaard. Processes for an architecture of volume. In Sigrid Brell-Çokcan and Johannes Braumann, editors, *Rob|Arch 2012: Robotic Fabrication in Architecture, Art, and Design*, pages 62–71, Vienna, 2013. Springer Vienna.

[OEB33] W. A. Oldfather, C. A. Ellis, and D. M. Brown. Leonhard Euler's elastic curves. *Isis*, 20(1):72–160, 1933.

[RJGK16] Romana Rust, David Jenny, Fabio Gramazio, and Matthias Kohler. Spatial wire cutting: Cooperative robotic cutting of non-ruled surface geometries for bespoke building components. In *Living Systems and Micro-Utopias: Towards Continuous Designing, Proceedings of the 21st International Conference of the Association for Computer-Aided Architectural Design Research in Asia CAADRIA 2016*, pages 529—-538, Hong Kong, 2016. The Association for Computer-Aided Architectural Design Research in Asia (CAADRIA).

[RKK12] Christian Raun, Mathias K. Kristensen, and Poul Henning Kirkegaard. Dynamic double curvature mould system. In Christoph Gengnagel, Axel Kilian, Norbert Palz, and Fabian Scheurer, editors, *Computational Design Modelling: Proceedings of the Design Modelling Symposium Berlin 2011*, pages 291–300, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[Saa13] Louis Saalschütz. *Belastete Stab unter Einwirkung einer Seitlichen Kraft.* Classic Reprint Series. Forgotten Books, 2013. Original published by B. G. Teubner, Leipzig in 1880.

[SFN$^+$16] A. Søndergaard, J. Feringa, T. Nørbjerg, K. Steenstrup, D. Brander, J. Gravesen, S. Markvorsen, A. Bærentzen, K. Petkov, J. Hattel, K. Clausen, K. Jensen, L. Knudsen, and J. Kortbek. Robotic hot-blade cutting: An industrial approach to cost-effective production of double curved concrete structures. In *Robotic Fabrication in Architecture, Art and Design 2016*, pages 150–164. Springer, 2016.

[SNS$^+$16] K. H. Steenstrup, T. B. Nørbjerg, A. Søndergaard, J. A. Bærentzen, and J. Gravesen. Cuttable ruled surface strips for milling. In *Advances in Architectural Geometry 2016*, 2016. To appear.

[Tru83] C. Truesdell. The influence of elasticity on analysis: The classic heritage. *Bull. Amer. Math. Soc. (N.S.)*, 9(3):293–310, 1983.

[WB05] Andreas Wächter and Lorenz T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Math. Program.*, 106(1, Ser. A), 2005. Published online: April 28, 2005.

[WW62] E. T. Whittaker and G. N. Watson. *A Course of Modern Analysis. An Introduction to the General Theory of Infinite Processes and of Analytic Functions: with an Account of the Principal Transcendental Functions.* Cambridge University Press, New York, 1962. Fourth edition. Reprinted.