Computation and Visualization of Ideal Knot Shapes

THÈSE N^{O} 4621 (2010)

PRÉSENTÉE LE 26 FÉVRIER 2010

À LA FACULTÉ INFORMATIQUE ET COMMUNICATIONS

CHAIRE D'ANALYSE APPLIQUÉE

PROGRAMME DOCTORAL EN INFORMATIQUE, COMMUNICATIONS ET INFORMATION

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Mathias CARLEN

acceptée sur proposition du jury:

Prof. M. A. Shokrollahi, président du jury
Prof. J. Maddocks, directeur de thèse
Prof. G. Abou Jaoudé, rapporteur
Prof. F. Cazals, rapporteur
Prof. E. Rawdon, rapporteur

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Suisse
2010

Abstract

We investigate numerical simulations and visualizations of the problem of tying a knot in a piece of rope. The goal is to use the least possible rope of a fixed, prescribed radius to tie a particular knot, e.g. a trefoil, a figure eight, and so on. The ropelength of the knot, the ratio to be minimized, is its length divided by its radius. An overview of existing algorithms to minimize the ropelength is given. They are based on different discretizations. Our work builds on the biarc discretization, for which we have developed an entire C++ library `libbiarc`. The library contains a variety of tools to manipulate curves, knots or links. The biarc discretization is particularly well suited to evaluation of thickness.

To compute ideal knot shapes we use simulated annealing software, which is also included in `libbiarc`, on a biarc discretization. Simulated annealing is a stochastic optimization algorithm that randomly changes the point or tangent data. In the quest to find appropriate moves for this process we arrived upon a Fourier representation for knots, which allows global changes to the curve in the annealing process. Moreover, with the Fourier representation we can enforce symmetries that a given knot might have. To identify these symmetries we use visualization of simulations where symmetry was not enforced.

Visualization of knot shapes and their properties is another important aspect in this work. It ranges from simple graphs of the curvature of a knot, through 2-dimensional plots of certain distance, circle or sphere functions, to 3-dimensional images of contact properties. Specially designed color gradients have been developed to emphasize crucial regions of the plots.

We show that the contact set of ideal torus knots is a curve that is ambient isotopic to the knot itself, which is a result first suggested by visualization. A combination of numerics and visualization made us aware of a closed trajectory within the trefoil knot, a 9-billiard. Consequently the symmetries and the billiard make it possible to represent the trefoil with only two curve sub segments.

We also anneal and visualize knot shapes in the unit 3-sphere or $\mathbb{S}^3$. In particular we present the contact set of a candidate for optimality, whose curved contact chords form Villarceau circles, which in turn span a Clifford torus embedded in the unit 3-sphere.

Finally some knots and contact surfaces are constructed as physical 3D models using 3D printers.

Keywords: Ideal knots, simulated annealing, Fourier knots, curve symmetries, sci-

entific visualization.

Résumé

Dans cette thèse nous étudions comment nouer une corde, au travers de calculs numériques et de la visualisation. L'objectif est d'utiliser le moins de corde possible d'un rayon fixé pour nouer un nœud de trèfle, un nœud en huit, etc. La "longueur de corde" d'un nœud, grandeur que nous cherchons à minimiser, est définie comme le rapport entre la longueur et le rayon de la corde. Nous présentons un vue d'ensemble des algorithmes existants pouvant servir à minimiser la longueur de corde. Ces algorithmes utilisent différentes discrétisations pour représenter un nœud. Notre travail est basé sur la discrétisation des biarcs, pour laquelle nous avons développé une bibliothèque C++ `libbiarc`. Cette bibliothèque contient une multitude d'utilitaires servant à manipuler et analyser des courbes ou des nœuds. La discrétisation des biarcs est particulièrement adaptée à l'évaluation de l'épaisseur.

Pour calculer des nœuds idéaux, qui minimisent la longueur de corde, nous utilisons un algorithme appelé "recuit simulé", qui est aussi contenu dans `libbiarc`. Le recuit simulé est une méthode stochastique pour résoudre des problèmes d'optimisation. Dans notre cas, il modifie aléatoirement les points ou les tangentes de la courbe. En cherchant une manière appropriée de modifier un nœud lors du recuit simulé, nous avons trouvé que la représentation d'un nœud en séries de Fourier se prête bien au processus, car elle permet d'appliquer des changements globaux à la courbe. En outre, cette représentation permet d'imposer à un nœud les symétries qu'il est susceptible de présenter. Afin de déterminer ces symétries, nous utilisons des techniques de visualisation des nœuds auxquels on n'a pas imposé de symétrie.

La visualisation de nœuds et de leurs propriétés est un aspect important de ce travail. Cela couvre de simple graphes de la courbure d'un nœud, en passant par des graphes en deux dimensions de fonctions de distance, cercles ou sphères, jusqu'à des images en trois dimensions des propriétés de contact. Nous avons développé des gradients de couleurs spécifiques afin d'accentuer la structure fine des graphes en deux dimensions.

Nous montrons que l'ensemble de contact des nœuds de tore idéaux est une courbe qui est une isotopie ambiante du nœud lui-même. Ce résultat a d'abord été suggéré par la visualisation. En effet, une combinaison de résultats numériques et de visualisation nous a fait découvrir une trajectoire fermée à l'intérieur du nœud de trèfle, le 9-billiard. Ceci implique qu'on peut décrire le trèfle avec seulement deux sous-segments du nœud intégral.

Des simulations et la visualisation ont aussi été appliquées à des nœuds dans la

sphère unité $\mathbb{S}^3$. En particulier, nous présentons l'ensemble de contact d'un candidat optimal, pour lequel cet ensemble forme des cercles de Villarceau qui génèrent le tore de Clifford plongé dans $\mathbb{S}^3$.

Finalement, nous avons fait imprimer en 3D quelque nœuds et des surfaces de contact.

Mots clés: Nœuds idéaux, recuit simulé, nœuds de Fourier, symétries de courbes, visualisation scientifique.

Acknowledgements

Credits and thanks

Supervisor	Prof. John H. Maddocks
Thesis committee member	Prof. Eric Rawdon
Thesis committee member	Prof. Frédéric Cazals
Thesis committee member	Prof. Georges Abou-Jaoudé
Thesis commitee president	Prof. Amin Shokrollahi
Coding mentor	Ben Laurie
IT & science	Philippe Caussignac
Secretary	Carine Tschanz
Team mate	Henryk Gerlach
Photographer	Bruno Favre
Roommate	Benoît Crouzy
Roommate	François Roy
IM and bike buddy	Gillian Coudray
Love and support	Parents & sister
My ♡	Maroussia Favre

Contents

Chapter 1

Introduction

In this thesis we elucidate how scientific visualization and computation can aid in the understanding of properties of thick curves and knots embedded in $\mathbb{R}^3$ and $\mathbb{S}^3$. In the absence of thickness these objects are well known and have been extensively studied in the rather abstract world of topology and differential geometry. Problems of thickened curves and tubes have also been studied. There are however still numerous points where clarification is needed. For example ideal knot shapes are a specific class of curves where one tries to tie a knot of given topological category with the least rope possible, subject to a width being prescribed. At first glance this seems like a very natural and easily solved problem. In practice, however, only a few simple cases are known exactly, e.g. the circle or unknot, and the Hopf link. The latter is formed by two circles running one through the other. Our concern in this text is dedicated to knots. The circle is not non-trivially knotted, and already for the first non trivial knot, namely the trefoil, no analytic expression for its ideal configuration is known. Visualization was key to understanding properties of numerically computed ideal configurations of knot shapes. In particular contact characteristics of ideal knot shapes have been investigated based on graphical representations of some form. Standard differential geometry describes quantities like curvature and torsion of a space curve that can easily be visualized on the curve in 3D to see where the curve is highly curved or wound. During this work graphical applications have been developed to do such tasks. A special type of 2-dimensional plots of curves related to line segments, circles and spheres running through, or tangent at, points on the curve are studied and visualized. The connection between a given knot in space and its 2-dimensional plots permitted conclusions to be drawn, eventually leading to hypotheses and formal proofs of conjectures motivated by visualization.

The structure of the thesis is as follows. A review of the mathematical concepts of space curves is given in Chapter 2. Further the notion of a thick curve and knot classes are discussed. The special class of ideal knot shapes is also introduced in that chapter. An ideal knot minimizes the ropelength, i.e. the length of the knot divided by the maximum tube radius, also called thickness of the curve. The second section of

this chapter describes the biarc discretization where a curve discretized by point-tangent data is interpolated by joining neighboring data points with two circular arcs in a $C^{1,1}$ fashion.

The third chapter, which is joint work with H. Gerlach, introduces another numerical approximation for curves based on a Fourier representation. A knot can be written as a Fourier series and the main reason for this parameterization is that symmetry can easily be enforced. The main result is the conjectured symmetry groups for the trefoil, the figure eight knot and the 5_1 knot, where the symmetries are reflected in the dependency of the Fourier coefficients of the knot.

Chapter 4 deals with the `C++` library `libbiarc` developed during this thesis which contains tools to manipulate, analyze and visualize open or closed curves and links based on the biarc discretization[1]. For our purposes the main feature of the biarc discretization is that thickness can be efficiently evaluated. In addition to the numerical tools included in the `libbiarc`, there are graphical applications to visualize different aspects of curves and knots. The viewer program `curview` has been used to inspect knot shapes and their properties and eventually grew to a visual scientific tool box. The investigation of an example curve in this chapter takes the reader through the different steps involved, when doing a "standard" analysis, in our sense, of a knot, or ideal knot. This includes the analysis of 2D plots and their visualization in 3D in the viewer. It is possible to generate most of the graphs and 2D plots presented throughout this thesis with `curview` or `libbiarc`. Further, the viewer interface allows the user to edit and create curves and export the final result as a Povray or renderman RIB file, so that they can be rendered in a photorealistic renderer. Data formats can cause severe headaches, which is why `libbiarc` includes conversion tools for a few of the standard 3D formats, such as STL, OBJ and PLY. To be able to use the curves we study in other software packages that are better suited for a specific task, we always had the data format problem in mind. Near the end of this work we used Blender to render most of the images and `libbiarc` contains a set of Python scripts that can import specific information related to ideal knots into Blender.

Chapter 5 gives an overview of existing or extended algorithms to minimize the ropelength of a knot. In particular we implemented the Fourier representation from Chapter 3 in a simulated annealing program where we enforce the symmetries suggested by other numerical computations and visualizations of the knots. Simulated annealing is a stochastic algorithm for optimization. This algorithm has been extended to knots in $\mathbb{S}^3$. Results are presented along with the algorithms used. The two last sections in this chapter detail the algorithm to compute the thickness of a biarc approximated curve and a parallelization thereof. Most of the simulations in this thesis use the biarc discretization to evaluate the thickness of a knot.

[1]The starting point was `knotlib` from B. Laurie who helped a lot during the different stages of the development of the code and the author would like to thank him for his interesting comments and helpful criticisms.

In Chapter 6 the notion of a contact set and contact surface of an ideal knot is introduced. The major part of the contact set discussion is dedicated to the trefoil in $\mathbb{R}^3$. A surprising discovery when analysing the trefoil was the existence of a closed trajectory in the trefoil's contact set. If we follow its contact chords from a specific location on the trefoil, then after 9 iterations we end up at the point we started. This and the knots' symmetries aid in understanding the contact connectivity of the trefoil, whose description can be reduced to only two sub segments. The contact sets of the figure-eight and the class of torus knots in $\mathbb{R}^3$ are treated after the trefoil. The torus knots contact properties lay the foundation for the homotopy argument in Section 6.6, which is a proof that, under some assumptions, the contact points of a torus knot are again a curve, which has the same knot type as the base curve. The homotopy proof is again joint work with H. Gerlach. The remaining part of the chapter is dedicated to an angle condition for ideal knot shapes where a compatibility condition on the angles between the normal of the knot and its contact chords is derived.

The final Chapter 7 is intended to be more artistic in nature and describes how to print a contact surface of a given knot shape on a 3D printer with a few images of the resulting models[2].

[2]We would like to thank G. Abou-Jaoudé for printing the first set of trefoil contact sets on his 3D printers. The photos of the models are by courtesy of B. Favre.

Chapter 2

Theoretical background

2.1 Curves, thick curves and ideal knots

In this section we introduce standard concepts related to space curves. Space curves are well studied for example in differential geometry [12]. We will also describe the less widely studied thick curves [34, 43], and ideal knot shapes.

A curve γ is the image of a continuous 3-dimensional vector function $\gamma(s) \in C^0(I, \mathbb{R}^3)$, where $s \in I$ is a parameterization variable. The interval I is a closed subset of the real line $\mathbb{R}$. A curve γ is a C^k-curve when $\gamma(s) \in C^k(I, \mathbb{R}^3)$. We call a curve γ smooth, if it is differentiable to any order and has a non-vanishing tangent field $\gamma'(s) \neq 0$ for all $s \in I$. A smooth curve can be reparametrized such that $|\gamma'(s)| = 1$. This is called a unit speed or arc-length parameterization. If a curve γ is parameterized by arc-length, then the interval I is $[0, L]$, where L is the length of the curve. A curve is closed if $\gamma(L) = \gamma(0)$, in which case we interpret the parameter s modulo L. The curve is smoothly closed if the derivatives agree up to any order at $s = 0$. And finally, a curve is simple if there is no self intersection, meaning that $\gamma(s)$ is an injective map where $\gamma(s) = \gamma(t)$ only if $s = t$.

For an arc-length parameterized curve $\gamma \in C^3(I, \mathbb{R}^3)$ the unit tangent field is $t(s) = \gamma'(s)$. We define the curvature κ as $t'(s) = \kappa(s)n(s)$, where n is called the principal normal with $|n(s)| = 1$ and $\kappa(s) \geq 0$. Therefore we have $\kappa(s) = |\gamma''(s)|$. The curvature κ is the scalar rate of change of the tangent field $t(s)$. The radius of curvature ρ at s is $\rho(s) = 1/\kappa(s)$. If we consider the binormal $b(s) = t(s) \times n(s)$, then the negative of the rate of change of the binormal is the standard torsion τ, that is $b'(s) = -\tau(s)n(s)$.

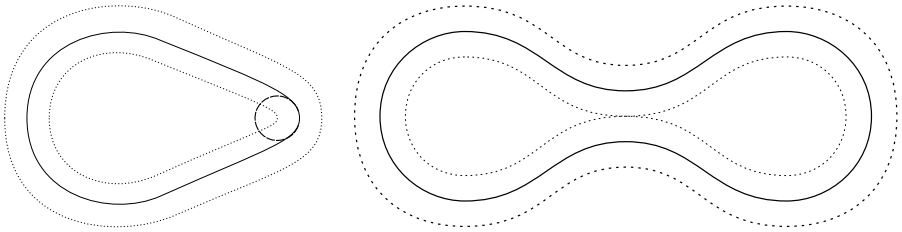
The definition of a curve as a one dimensional object embedded in space naturally generalizes to curves in $\mathbb{R}^N, N > 3$. In particular we will be interested in curves lying in $\mathbb{S}^3$, which is the unit sphere embedded in $\mathbb{R}^4$. Later in this text we will be interested in objects that physically have a certain volume, so we extend the definition of a curve

to a thick curve.

Definition 2.1. *Let $\gamma(s), s \in I$ be a closed, smooth curve. A thick curve with radius $r > 0$ is the tubular neighborhood*

$$T_r = \{p \in \mathbb{R}^3 \,|\, \exists s \in I \,|\, |p - \gamma(s)| < r\}.$$

It will be clear from the context whether we write γ for a one dimensional curve or a thick curve. If we consider the cross section of a thickened curve γ at s, where the cross section lies in the normal plane to $\gamma'(s)$ at s, then we can identify r as the radius of a tube around the curve γ. A thick curve can be seen as a physical piece of rope with a certain radius r. When we deform the rope its shape will always be constrained by the radius r. This is due to contact of the curve with itself. There are two different cases preventing the rope to intersect with itself, cf. Figure 2.1. The first case is a local effect. We can not bend the curve more than the radius of the rope itself. This means, if the local curvature κ gets larger than $1/r$, then we have self-intersection. The second case is a global contact and prevents the rope from passing through itself.



(a) (b)

Figure 2.1: (a) local, or (b) global self contact of a curve

Consider a thick curve γ and increase the tube radius r until we hit one of the two discussed barriers. Whether this is local curvature κ or a global contact depends on the shape of the curve. This maximal radius r is called the thickness $\Delta[\gamma]$ of the curve [24] and will be defined mathematically in what follows.

A chord $c(s, \sigma)$ on the curve γ is given by the oriented linear segment between two points on the curve

$$c(s, \sigma) = \gamma(s) - \gamma(\sigma), \qquad s, \sigma \in I. \tag{2.1}$$

We call a chord $c(s, \sigma)$, $s \neq \sigma$, on the curve γ double critical if

$$c(s, \sigma) \cdot t(s) = c(s, \sigma) \cdot t(\sigma) = 0. \tag{2.2}$$

The points $\gamma(s)$ and $\gamma(\sigma)$ where the chord is double critical are also called double critical points. Further we define the half-distance or *pp* function (point-point) [66] by

$$pp(s, \sigma) = \frac{1}{2}|c(s, \sigma)|, \tag{2.3}$$

which is a symmetric function whose smoothness depends on the curve γ.

The authors in [37] have shown that for a C^2 curve, the thickness of a curve is the smaller of the minimal radius of curvature and the minimal half-distance between double critical points. Therefore the thickness Δ of a curve gamma γ is

$$\Delta[\gamma] = \min\{\min_{s\in I}\rho(s), \min_{(s,\sigma)\in dc} pp(s,\sigma)\}, \tag{2.4}$$

where dc is the set of double critical points on γ with $s \neq \sigma$. The half-distance function is zero along the diagonal $s = \sigma$. A double critical chord c of norm 2Δ will also be called a contact or contact chord.

We now introduce two other functions pt (point-tangent) and tt (tangent-tangent) that can be used to define the thickness in another way. Three points $\gamma(s), \gamma(\sigma), \gamma(\tau) \in \mathbb{R}^3$ define a unique circle with radius r. If we take the limit $\tau \to \sigma$, then we obtain a circle tangent at $\gamma(\sigma)$ and passing through $\gamma(s)$. The radius of this circle is the pt function

$$pt(s,\sigma) = \frac{pp(s,\sigma)}{|\sin\theta|}, \tag{2.5}$$

where θ is the angle between the tangent at $\gamma(\sigma)$ and the chord $c(s,\sigma)$. The pt function is not symmetric. So there is also the function tp, where the tangency is in the first point. However in this thesis we will only consider pt explicitly, since the other function is equivalent in the sense that the needed results remain the same.

In [24] the thickness of a curve was defined for C^2 curves via the global radius of curvature function $\rho_G(s)$, where

$$\rho_G(s) = \rho_{pt}(s) = \inf_{\sigma\in I} pt(s,\sigma). \tag{2.6}$$

The thickness Δ given as follows is then equivalent [24] to (2.4).

$$\Delta_g[\gamma] = \inf_{s\in I}\rho_G(s) \tag{2.7}$$

The function $pt(s,\sigma)$ does not vanish along the diagonal. If the curve is C^2, then in the limit $\sigma \to s$ the circle converges to the osculating circle at s, whose radius is the radius of curvature $\rho(s)$. The curvature profile of a smooth enough curve γ is therefore given by the diagonal of the function pt. At a global contact the function $pt(s,\sigma)$ is exactly half the distance of the corresponding contact chord $c(s,\sigma)$. The infimum in Equation (2.7) picks out the smallest of these local and global contacts yielding the thickness of the curve.

For the third function tt we consider spheres on a curve γ [26, 11, 66]. This time we start with four points $\gamma(s), \gamma(t), \gamma(\sigma)$ and $\gamma(\tau)$ on a curve. Four non-coplanar points in space uniquely define a sphere with radius r. By taking the limits $t \to s$ and $\tau \to \sigma$ we obtain a sphere tangent to the curve at the points $\gamma(s)$ and $\gamma(\sigma)$. The radius of this sphere is the function tt given by

$$tt(s,\sigma) = pp(s,\sigma)\frac{|\sin\alpha|}{|t(s)\times t(\sigma)\cdot e(s,\sigma)|} \tag{2.8}$$

where e is the normalized chord $e(s,\sigma) = c(s,\sigma)/|c(s,\sigma)|$ and the angle α is that between the unit vectors $t(\sigma)$ and $R(e)t(s)$, where $R(e)$ is the reflection matrix

$$R(e) = 2e \otimes e - id_{\mathbb{R}^3}. \tag{2.9}$$

Note that $e \otimes e$ is the standard outer product and $id_{\mathbb{R}^3}$ the identity matrix. For a curve γ in $\mathbb{S}^3$ we have to adapt Equation (2.8). In particular the mixed product $t(s) \times t(\sigma) \cdot e(s,\sigma)$ is not defined. This product is the volume of a parallelepiped spanned by the vectors $t(s), t(\sigma)$ and $e(s,\sigma)$. The volume of this parallelepiped squared is also given by the Gramian [17, 4]

$$G(x_1, \ldots, x_n) = det(M^T M), \tag{2.10}$$

where M is the matrix whose columns are the vectors $x_1, \ldots, x_n \in \mathbb{R}^N$. In $\mathbb{S}^3$ we can write the norm of the mixed product in (2.8) as $\sqrt{G(t(s), t(\sigma), e(s,\sigma))}$. The tt function for a curve in $\gamma \in C^1(I, \mathbb{S}^3)$ is then

$$tt(s,\sigma) = pp(s,\sigma) \frac{|\sin\alpha|}{\sqrt{G(t(s), t(\sigma), e(s,\sigma))}}, \tag{2.11}$$

where the angle α is the same as in (2.8). Consider a critical chord $c(s,t)$ of a curve γ for some $s, t \in I$. If we draw a circle centered at $(\gamma(s) + \gamma(t))/2$ with radius $pp(s,t)$, the circle is tangent at $\gamma(s)$ and $\gamma(t)$. Changing the tangent on one side yields a larger circle since we lose orthogonality there. Thus $pp(s,t) \leq pt(s,t)$ as is already obvious from (2.5). A similar argument gives $pt(s,t) \leq tt(s,t)$. However, an interesting relation between the functions pp, pt and tt is at double critical points or contacts [26, 66]

$$pp(s,t) = pt(s,t) = tp(s,t) = tt(s,t). \tag{2.12}$$

In theory we are now able to compute the thickness Δ of a given closed curve. Knot theory deals with closed curves, namely knots and links. We will briefly explain what a knot or link is and how they are classified. More details can be found in standard knot theory references such as [1, 59].

Definition 2.2. *1. A knot or knot shape $K \subset \mathbb{R}^3$ is the image of a closed, injective (non-self-intersecting), continuous curve γ in $\mathbb{R}^3$.*

2. A link $L \subset \mathbb{R}^3$ is the union of a finite number of pairwise disjoint knots, $L = K_1 \cup \cdots \cup K_m$. A knot is a special case of a link.

3. An ambient isotopy is a continuous map $h : \mathbb{R}^3 \times [0,1] \to \mathbb{R}^3$ with $h(\cdot, 0) = id_{\mathbb{R}^3}$ and $h(\cdot, t)$ a homeomorphism for all $t \in [0,1]$.

4. Two knots K_1 and K_2 are ambient isotopic (notation: $K_1 \simeq K_2$) if an ambient isotopy $h : \mathbb{R}^3 \times [0,1] \to \mathbb{R}^3$ exists with $h(K_1, 1) = K_2$.

An ambient isotopy between two knot shapes exists if it is possible to transform a real knotted rope into the other without having to cut it. The definition of an ambient

isotopy also excludes the pull tight phenomenon, where one can remove a knot by pulling at both ends until the knot converges to a single point. The isotopy class of a knot K is written $[K]$. Previously we discussed that the radius of the tubular neighborhood of a curve K is determined either by local curvature or a global contact, this radius is the thickness $\Delta[K]$. If we now try to deform the knot K in isotopy class $[K]$ with the goal to make the thickness as large as possible by keeping its length $L[K]$ fixed, then we obtain a sub-category of knot shapes, namely ideal knots.

Definition 2.3. *An ideal shape K^* [7, 29, 67] of the isotopy class $[K]$ is a knot shape that minimizes the functional length/thickness within its isotopy class, i.e. an ideal knot shape is a solution of*

$$\frac{L[K]}{\Delta[K]} \to \min$$

subject to $K \simeq K^*$. The positive number $L[K]/\Delta[K]$ is called the ropelength of the knot K.

The notation $\mathbb{S}$ for the parameter interval I will often be used during this thesis and means that a closed curve can be parameterized with constant speed such that $I = \mathbb{S} = \mathbb{R}/\mathbb{Z}$. Another point is the function space in which ideal shapes live, which has been proved to be $C^{1,1}(\mathbb{S}, \cdot)$ for curves in $\mathbb{R}^3$ [8, 22, 25], and $\mathbb{S}^3$ [19].

We remark that ropelength is a special form of a knot energy [14, 47, 46, 57]. A knot energy is a functional $E[\cdot]$ on a knot K which can usually be minimized by strategically deforming K. Different knot energies will not necessarily lead to the same minimizing shapes. In this thesis we consider only the thickness and ropelength energy.

2.2 Biarcs

This thesis is heavily based on numerical computations of knot shapes. At this point we introduce several approximations for curves. This section gives a high-level overview of the concept of a biarc, following the definitions and conventions set in [6, 64, 11, 66, 71]. First we start off by introducing the data we want to interpolate with biarcs.

Definition 2.4. *Point-tangent data is of the form $(q, t) \in \mathcal{J} := \mathcal{P} \times \mathcal{T}$. A point-tangent data pair is of the form $((p_0, t_0), (p_1, t_1)) \in \mathcal{J} \times \mathcal{J}$ with $p_0 \neq p_1$.*

For curves in $\mathbb{R}^3$ the point space $\mathcal{P}$ is $\mathbb{R}^3$ and the normalized tangent space $\mathcal{T}$ is $\mathbb{S}^2$. For $\mathbb{S}^3$ we have $\mathcal{P} = \mathcal{T} = \mathbb{S}^3$. What follows applies as well to point-tangent data in $\mathbb{R}^N \times \mathbb{S}^{N-1}$ or data constrained to lie on a unit sphere $\mathbb{S}^N \times \mathbb{S}^N$ for $N > 2$ (cf. [71]).

Definition 2.5. *A biarc $(a, \overline{a})$ is a pair of circular arcs, joined continuously and with continuous tangents, that interpolate a point-tangent data pair. The common end point m of the two arcs a and $\overline{a}$ is the matching point of the biarc.*

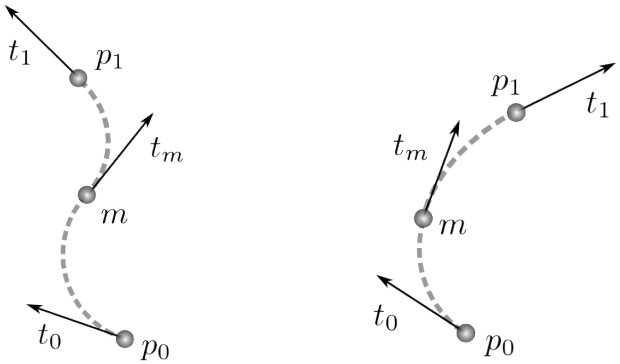


Figure 2.2: A biarc is a pair of circular arcs, assembled with a common tangent t_m at a matching point m, that interpolates a given pair of point-tangent data $((p_0, t_0), (p_1, t_1))$. The biarc lies entirely on a unique sphere defined by the two tangent pairs.

The construction of Definition 2.5 is illustrated in Figure 2.2. A matching point m is not unique for a given point-tangent data pair. In order to state where the matching points m can lie, it is necessary to define a few more objects. The data (p_0, t_0) and (p_1, t_1) define a unique sphere (neglecting pathological cases). This double tangent sphere has radius

$$r = \frac{|d|}{2} \frac{\sqrt{1 - (t_1^* \cdot t_0)^2}}{\sqrt{G(t_0, t_1, e)}}, \tag{2.13}$$

where $d = p_1 - p_0$ is the chord between the data points and e the normalized chord $e = d/|d|$. The vector t_1^* is a reflection of t_1 along e (see Figure 2.3). Using the definition of the reflection matrix (2.9) we can write $t_1^* = R(e)t_1$. In $\mathbb{R}^3$, the center of the sphere is given by

$$c = p_0 + \frac{|d|}{2} \frac{t_1^* \times t_0}{t_0 \times t_1 \cdot e}.$$

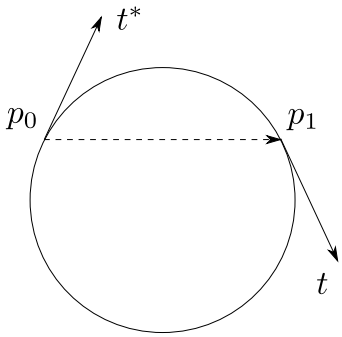


Figure 2.3: Action of the reflection matrix R on t along $p_1 - p_0$.

As illustrated in Figure 2.4, there is now a unique circle C_0, tangent to t_0 at p_0 and running through p_1. A similar circle C_1 exists with a tangency in t_1. The circle bisecting the angle between C_0 and C_1 on the tangent sphere is denoted C_+ (i.e. the circle with

tangent $t_0 + t_1^*$ at p_0 and passing through p_1). We are actually only interested in the matching points m lying on the smaller arc of C_+ between p_0 and p_1, denoted C_+^*. This description of the circles C_0, C_1 and C_+ is what the next proposition states.

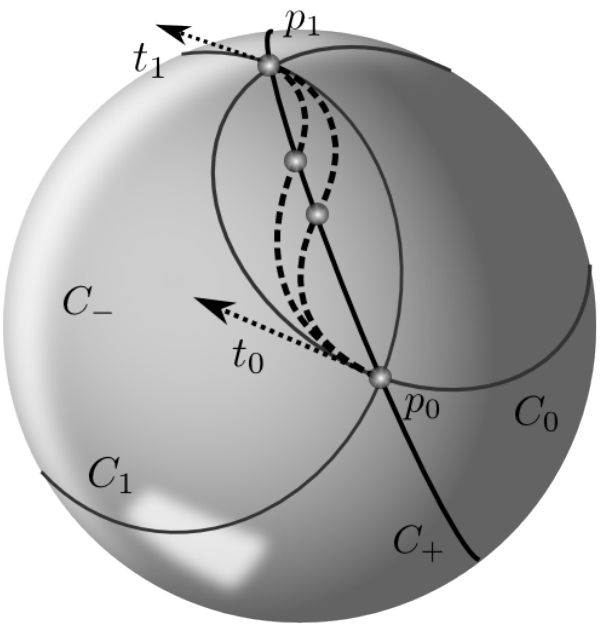


Figure 2.4: Biarc construction for point-tangent data pairs (p_0, t_0) and (p_1, t_1). The tangents define the circles C_0 and C_1. The bisector circle C_+ is at half the angle between the circles C_0 and C_1. The matching points m for the biarc lie on C_+, in particular we only consider m on the smaller arc of circle between p_0 and p_1 noted C_+^*.

Proposition 2.1 (Sharrock)**.** *For a point-tangent data pair $((p_0, t_0), (p_1, t_1)) \in \mathcal{J} \times \mathcal{J}$ that is non-cocircular ($C_0 \neq C_1$), there is a nonempty set of possible matching points m corresponding to compatibly oriented arcs. For each matching point m the biarc is unique. Moreover*

1. *The set C_+ of all possible matching points is a circle (punctured at p_0 and p_1).*
2. *All biarcs lie on the double tangent sphere generated by $((p_0, t_0), (p_1, t_1))$.*
3. *There is a unique biarc through every point on the double tangent sphere punctured at p_0 and p_1 (i.e. the set of all possible biarcs is a simple covering of the double tangent sphere).*

Since biarcs will be used to approximate a smooth curve, we have to impose conditions on the data itself.

Definition 2.6. *The point-tangent data pair $((p_0, t_0), (p_1, t_1)) \in \mathcal{J} \times \mathcal{J}$ will be called proper if*

$$(p_1 - p_0) \cdot t_0 > 0, \text{ and } (p_1 - p_0) \cdot t_1 > 0. \tag{2.14}$$

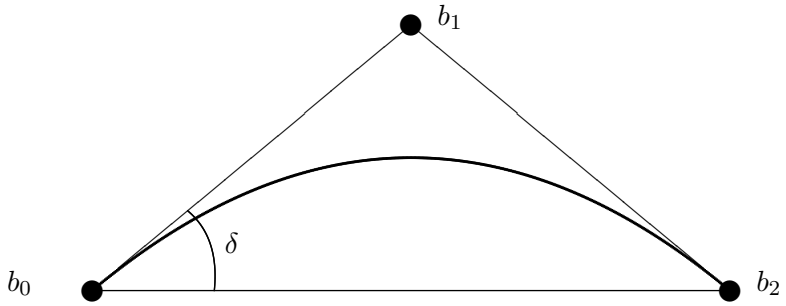


Figure 2.5: The isosceles Bézier control triangle of an arc $a(\tau)$, $\tau \in [0, 1]$.

Then the biarc interpolating this point-tangent data pair is also called proper, given that the matching point m lies on C_+^*.

Proper biarcs (given proper data pairs) interpolating the data imply a biarc curve with a continuous tangent field.

Proposition 2.2 (Smutny [66])**.** *For a proper point-tangent data pair $((p_0, t_0), (p_1, t_1)) \in \mathcal{J} \times \mathcal{J}$,*

- *The angle spanned by C_+^* on the tangent sphere is smaller than π.*
- *For any biarc with matching point $m \in C_+^*$, the angle spanned by both sub-arcs a and $\overline{a}$ is smaller than π.*

The biarc approximation of a sufficiently smooth curve is now given by a set of proper biarcs. The next step is the parameterization of the sub arcs of each biarc. There are many ways to parameterize an arc of a circle. The approach taken here is with an isosceles triangle of Bézier control points [52]. For the control points $b_0, b_1, b_2 \in \mathbb{R}^N$ as illustrated in 2.5 we have

$$a(\tau) = \frac{(1-\tau)^2 b_0 + 2\omega\tau(1-\tau)b_1 + \tau^2 b_2}{(1-\tau)^2 + 2\omega\tau(1-\tau) + \tau^2}, \qquad \tau \in [0, 1], \tag{2.15}$$

where ω is the cosine of the base angle δ of the isosceles control triangle, that is,

$$\omega = \frac{(b_1 - b_0) \cdot (b_2 - b_0)}{|b_1 - b_0|\,|b_2 - b_0|} > 0.$$

Note that the parameter τ is not an arc-length parameterization of the Bézier curve. Given a pair of point-tangent data, all the necessary parameters involved in constructing two Bézier control triangles are summarized in the following lemma.

Lemma 2.1 (Smutny)**.** *Given a proper point-tangent data pair $((p_0, t_0), (p_1, t_1)) \in \mathcal{J} \times \mathcal{J}$ and a proper biarc $(a, \overline{a})$, then:*

- *The Bézier control points of the arc C_+^* are given by*

$$b_0 = p_0, \quad b_1 = \frac{(t_0 \cdot d)p_0 + (t_1 \cdot d)p_1}{t_0 \cdot d + t_1 \cdot d} + \frac{|d|^2(t_0 - t_1)}{2(t_0 \cdot d + t_1 \cdot d)}, \quad b_2 = p_1. \tag{2.16}$$

- *The Bézier control points of the arc a are*

$$b_0 = p_0, \quad b_1 = p_0 + \Lambda \frac{|d|^2}{2t_0 \cdot d} t_0, \quad b_2 = m, \tag{2.17}$$

and the Bézier control points of the arc $\bar{a}$ are

$$b_0 = m, \quad b_1 = p_1 - \bar{\Lambda} \frac{|d|^2}{2t_1 \cdot d} t_1, \quad b_2 = p_1, \tag{2.18}$$

where $m \in C_+^*$ is the matching point given by the formula

$$m = \frac{\bar{\Lambda}(t_0 \cdot d)p_0 + \Lambda(t_1 \cdot d)p_1}{\Lambda t_1 \cdot d + \bar{\Lambda} t_0 \cdot d} + \frac{\Lambda\bar{\Lambda}|d|^2(t_0 - t_1)}{2(\Lambda t_1 \cdot d + \bar{\Lambda} t_0 \cdot d)}, \tag{2.19}$$

and Λ and $\bar{\Lambda} \in (0,1)$ are two parameters that are roots of the equation

$$0 = \Lambda\bar{\Lambda}(1 - t_0 \cdot t_1) + (\Lambda + \bar{\Lambda} - 1)\, 2t_0 \cdot e\, t_1 \cdot e. \tag{2.20}$$

If Λ is regarded as the independent variable, $\bar{\Lambda}$ is given by

$$\begin{aligned} \bar{\Lambda} &= \frac{2(1-\Lambda)\,(t_0 \cdot e)\,(t_1 \cdot e)}{\Lambda(1 - t_0 \cdot t_1) + 2(t_0 \cdot e)\,(t_1 \cdot e)} \\ &= \frac{2(1-\Lambda)\,(t_0 \cdot d)\,(t_1 \cdot d)}{\Lambda(1 - t_0 \cdot t_1)|d|^2 + 2(t_0 \cdot d)\,(t_1 \cdot d)}. \end{aligned} \tag{2.21}$$

Alternatively if a matching point m on C_+^* is given, the biarc parameters Λ and $\bar{\Lambda}$ are

$$\Lambda = \frac{t_0 \cdot d\,|m - p_0|^2}{t_0 \cdot (m - p_0)\,|d|^2} \qquad \textit{and} \qquad \bar{\Lambda} = \frac{t_1 \cdot d\,|p_1 - m|^2}{t_1 \cdot (p_1 - m)\,|d|^2}, \tag{2.22}$$

with d as defined in Equation 2.13.

The matching point m can be chosen freely on C_+^* for every biarc approximating a curve. One possibility is to choose m such that it has the same Euclidean distance from p_0 and p_1. The matching points m lie on C_+^* whose Bézier control points b_0, b_1 and b_2 are given by (2.16). The arc of circle is given by (2.15) and the midpoint m satisfying the equal distance condition is

$$m = a(1/2) = \frac{b_0 + 2\omega b_1 + b_2}{2(\omega + 1)}$$

for a pair of point-tangent data $(p_0, t_0), (p_1, t_1)$. Let $b, \bar{b}$ be the sub-arcs of circles of biarc $a(\tau), \tau \in [0,1]$ and $r(b), r(\bar{b})$ their radii. The symmetry of the triangle p_0, m, p_1 implies that $r(b) = r(\bar{b})$.

These are all the tools necessary to construct a biarc approximation of a smooth curve γ in $\mathbb{R}^N$ or $\mathbb{R}^N$ by interpolation of M point-tangent data pairs.

Chapter 3

Fourier representation of closed curves

3.1 Fourier knots

A periodic function $f(t)$ can be written as a Fourier series [68, 13] where the linearly independent base functions are $\sin(k\,t)$ and $\cos(k\,t)$ for $t \in [0, 2\pi)$. For a closed curve $\gamma(t), t \in \mathbb{S}$ embedded in $\mathbb{R}^3$ we can therefore define 3 Fourier series, one for each coordinate function of γ.

Definition 3.1. *Let C be a finite sequence of pairs of $\mathbb{R}^3$-vectors:*

$$C = \{(a_i, b_i)\}_{i=1,\cdots,k} \quad a_i, b_i \in \mathbb{R}^3.$$

We can use such a sequence as Fourier coefficients and define

$$\gamma(t) = \sum_{i=1}^{k} \left(a_i \cos(f_i t) + b_i \sin(f_i t)\right), \quad t \in [0, 1] \tag{3.1}$$

as a curve in $C^\infty(\mathbb{S}, \mathbb{R}^3)$ with frequencies $f_i = 2\pi i$. If the curve is injective, we call γ a Fourier knot.

Remark 3.1. *Note that the sum in the above definition starts at $i = 1$ and the constant a_0 is neglected. This term is independent of the parameter t and therefore just a translation of the Fourier knot.*

The Fourier representation of knots has been used in for example [30, 69], where the emphasis is on finding simple Fourier shapes of given knot types. Ideal knot shapes are known to live in $C^{1,1}(\mathbb{S}, \mathbb{R}^3)$ [8, 19, 22, 25] and Fourier series are functions in C^∞.

A Fourier representation has several advantages over the biarc representation introduced in Section 2.2.

1. They make it easy to inherently model symmetry (see Section 3.2) and by that improve computation speed.
2. Any sequence of coefficients yields a valid Fourier curve while point biarc data needs to be "proper" (Definition 2.6), which is sometimes difficult to ensure.
3. It is trivial to improve the approximation 3.1 by adding more coefficients.
4. If an ideal knot is given by its Fourier representation, which is a C^∞ approximation, then higher derivatives involved in curvature and torsion can be computed (even though the ideal shape might not itself be smooth enough to be differentiable).

At this point we have to ask the reader to trust that the thickness of a biarc curve can numerically be computed very precisely (the details will follow in Section 5.7). The main disadvantage for Fourier knots is, that no efficient computer-code exists to evaluate the said thickness. So we work around this problem by interpolating the Fourier knot with a sufficiently fine biarc curve and evaluate thickness on the latter. Further, unlike with biarcs, the Fourier representation is non-local.

3.2 Symmetry of curves

The definitions we give hereafter stress that a curve has a symmetry if and only if the resulting shape is identical to the original with respect to their parameterization. In other words, let γ be the original curve and γ_{sym} the shape after a symmetry has been applied. It is in fact only a symmetry if $\gamma(s) = \gamma_{sym}(s)$ for all s.

Definition 3.2 (G-Symmetry)**.** *Let $\gamma : \mathbb{S} \longrightarrow \mathbb{R}^N$ be some curve and $G \subset \mathbf{Aut}(C^0(\mathbb{S}, \mathbb{R}^N))$ a group acting on $C^0(\mathbb{S}, \mathbb{R}^N)$. We call γ G-symmetric iff*

$$g.\gamma = \gamma \quad \forall g \in G.$$

Here **Aut** is the automorphism group, which is the set of bijective mappings from a mathematical object to itself. Whenever $g.\gamma = \gamma, \forall g \in G$ (G maximal in some respect), we call G a symmetry-group of γ. Note that taking the maximal subgroup G of $\mathbf{Aut}(C^0(\mathbb{S}, \mathbb{R}^N))$ is not what we want since it is too large. Now we can state how to actually symmetrize a given closed curve.

Definition 3.3 (Symmetrized curve)**.** *Let $\gamma \in C^0(\mathbb{S}, \mathbb{R}^3)$ be some curve and $G \subset \mathbf{Aut}(C^0(\mathbb{S}, \mathbb{R}^3))$ a finite group. Then*

$$\gamma_G = \frac{\sum_{g \in G} g.\gamma}{|G|}$$

is the G-symmetrization of γ.

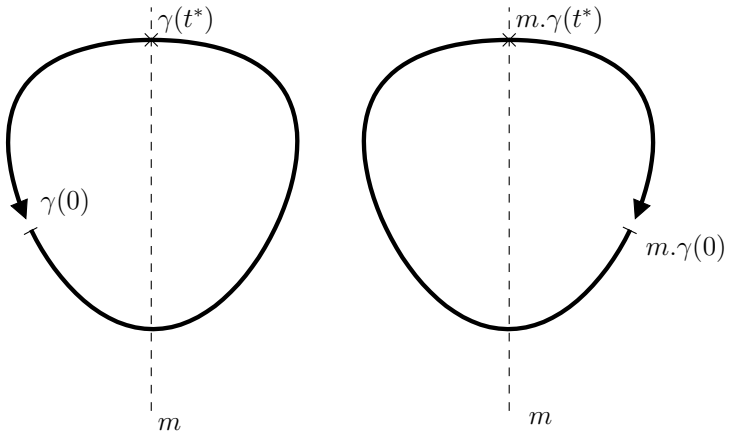


Figure 3.1: The symmetry-group of a curve must not only take the shape into account, but also the parameterization: The above egg-shaped curve is mirror symmetric to the dashed line as a point-set, but applying the reflection m does not yield the same parameterization.

As mentioned before, the parameterization of a knot plays an important role when dealing with symmetries. The tools needed to ensure that a curve and its symmetry image are identical are given now.

Definition 3.4 (Group of parameter shifts and reflections)**.** *For $x \in \mathbb{R}$ we define the parameter shift $S_x : C^0(\mathbb{S}, \mathbb{R}^3) \longrightarrow C^0(\mathbb{S}, \mathbb{R}^3)$ by x of a curve γ as*

$$S_x[\gamma](t) = \gamma(t + x)$$

and the parameter reflection $R_x : C^0(\mathbb{S}, \mathbb{R}^3) \longrightarrow C^0(\mathbb{S}, \mathbb{R}^3)$ around x as

$$R_x[\gamma](t) = \gamma(2x - t).$$

Shifts and reflections form a group.

Note that symmetry-groups of a curve are usually not just a subgroup of the orthogonal group $O(3)$ but a product of such a subgroup and a subgroup of parameter shifts and reflections. Neglecting the latter would only give a point-wise symmetrical shape not appropriate for our purposes. To make this difference clearer we give an example.

Example 3.1. *Consider an egg shaped curve γ as in Figure 3.1. The point-set $\gamma(\mathbb{S})$ is invariant under mirroring along the dashed line m. But the curve is not, because $\gamma(0) \neq m.\gamma(0)$. Choose a t^* such that $\gamma(t^*) = m.\gamma(t^*)$ and define $G := \{\mathrm{id}, m \circ R_{t^*}\}$ then γ will be G-symmetric.*

3.3 Symmetry of Fourier Knots

In the previous section we claim that symmetry is easy to enforce on Fourier knots, but before we can compute the symmetrization given in Definition 3.3 we need to explain how the Fourier coefficients transform with respect to a given symmetry operation. To rotate or reflect a point in $\mathbb{R}^3$ we have the following matrices.

Definition 3.5 (Rotation and Reflection Matrices)**.** *Let $v \in \mathbb{R}^3, ||v|| = 1$. We call $M_v(w) = w - 2\langle v, w\rangle v$ a reflection matrix. M_v mirrors on the hyperplane orthogonal to v. We call*

$$D_{v,\alpha} = \begin{pmatrix} \cos\alpha + v_1^2(1-\cos\alpha) & v_1v_2(1-\cos\alpha) - v_3\sin\alpha & v_1v_3(1-\cos\alpha) + v_2\sin\alpha \\ v_2v_1(1-\cos\alpha) + v_3\sin\alpha & \cos\alpha + v_2^2(1-\cos\alpha) & v_2v_3(1-\cos\alpha) - v_1\sin\alpha \\ v_3v_1(1-\cos\alpha) - v_2\sin\alpha & v_3v_2(1-\cos\alpha) + v_1\sin\alpha & \cos\alpha + v_3^2(1-\cos\alpha) \end{pmatrix}$$

a rotation around v by $\alpha \in \mathbb{R}$.

The next lemma describes how the coefficients in the Fourier representation transform when we reflect or rotate a Fourier knot or shift its parameterization.

Lemma 3.1 (Actions on Fourier knots)**.** *Let $\gamma(t) = \sum_i (a_i\cos(f_it) + b_i\sin(f_it))$ be a Fourier knot.*

- *Let $M \in O(3)$ be an orthogonal matrix. Then*

$$(M.\gamma)(t) = \sum_i \left(\bar{a}_i\cos(f_it) + \bar{b}_i\sin(f_it)\right)$$

with

$$\bar{a}_i = Ma_i, \qquad \bar{b}_i = Mb_i,$$

or

$$\begin{pmatrix}\bar{a}_i \\ \bar{b}_i\end{pmatrix} = \begin{pmatrix} M & 0 \\ 0 & M\end{pmatrix}\begin{pmatrix}a_i \\ b_i\end{pmatrix}$$

- *Let S_x be a parameter shift by some $x \in \mathbb{R}$. Then*

$$(S_x.\gamma)(t) = \sum_i \left(\bar{a}_i\cos(f_it) + \bar{b}_i\sin(f_it)\right)$$

with

$$\bar{a}_i = \cos(f_ix)a_i + \sin(f_ix)b_i, \qquad \bar{b}_i = -\sin(f_ix)a_i + \cos(f_ix)b_i,$$

or

$$\begin{pmatrix}\bar{a}_i \\ \bar{b}_i\end{pmatrix} = \begin{pmatrix} \cos(f_ix)I & \sin(f_ix)I \\ -\sin(f_ix)I & \cos(f_ix)I\end{pmatrix}\begin{pmatrix}a_i \\ b_i\end{pmatrix}$$

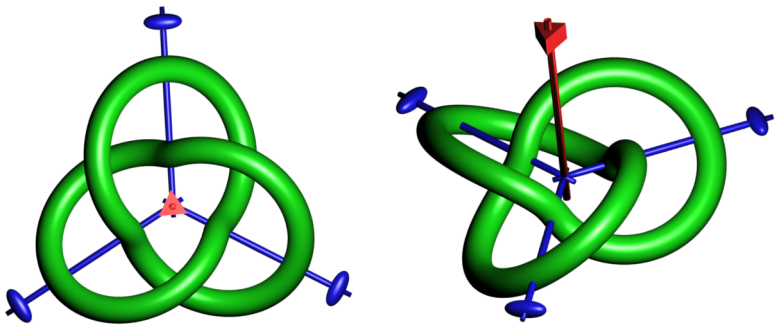


Figure 3.2: Trefoil symmetry axis for two different views. The red axis (prisma) is the 3-symmetry with rotation angles of $2\pi/3$. The three blue axis (ellipsoids) give the second symmetry, which is a rotation of angle π. The generators of the symmetry group are one element of both types.

- *Let R_x be a parameter reflection around $x \in \mathbb{R}$. Then*

$$(R_x.\gamma)(t) = \sum_i \Big(\bar{a}_i \cos(f_i t) + \bar{b}_i \sin(f_i t)\Big)$$

with

$$\bar{a}_i = \cos(2f_i x)a_i + \sin(2f_i x)b_i, \qquad \bar{b}_i = \sin(2f_i x)a_i - \cos(2f_i x)b_i,$$

or

$$\begin{pmatrix}\bar{a}_i \\ \bar{b}_i\end{pmatrix} = \begin{pmatrix}\cos(2f_i x)I & \sin(2f_i x)I \\ \sin(2f_i x)I & -\cos(2f_i x)I\end{pmatrix}\begin{pmatrix}a_i \\ b_i\end{pmatrix}$$

□

A priori we do not know a symmetry group for an ideal knot K within some isotopy class $[K]$ since there is no analytic expression for most of them. However studying approximately ideal shapes obtained by the numerical computations explained in later chapters suggests possible symmetries.

Conjecture 3.1 (Assumed symmetries of ideal 3_1, 4_1 and 5_1 knots.)**.** *After a reparametrization, a translation and a rotation the following symmetry-groups are suggested by the numerical data:*

- *Trefoil (3_1):*

$$G_{3_1} = \{(D_{(0\,0\,1)^t,2\pi/3} \circ S_{1/3})^i \circ (D_{(1\,0\,0)^t,\pi} \circ R_0)^j : i = 0,1,2, \quad j = 0,1\}.$$

The number of elements in this group is therefore $|G_{3_1}| = 6$.

- *Figure eight knot (4_1):*

$$G_{4_1} = \{(D_{(0\,1\,0)^t,\pi} \circ S_{1/2})^i \circ (M_{(0\,1\,0)^t} \circ D_{(0\,1\,0)^t,\pi/2} \circ S_{1/4})^j : i,j = 0,1\}.$$

This group has $|G_{4_1}| = 4$ elements.

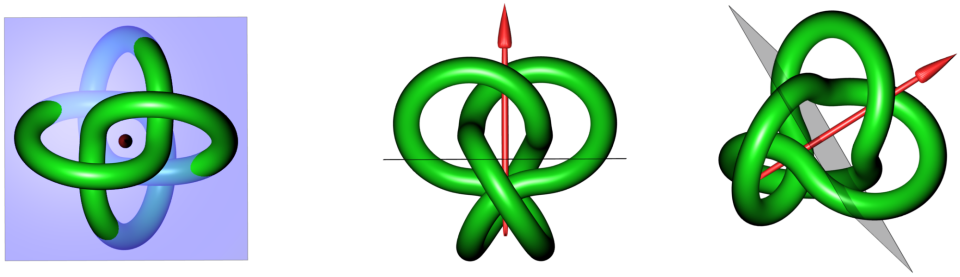


Figure 3.3: Three different views of the figure-eight knot symmetries. The first symmetry is a rotation of angle π around the red axis. The second generator of the symmetry group is a reflection through the light blue plane and then a rotation of angle $\pi/2$ around the red axis.

- *The 5_1-knot has only two elements in its symmetry group given by*

$$G_{5_1} \quad = \quad \{(D_{(0\,1\,0)^t,\pi} \circ R_0)^j : j = 0, 1\}.$$

These symmetries are visualized in Figures 3.2, 3.3 and 3.4. The trefoil has a 3-symmetry by rotating the knot by an angle of $2\pi/3$ around the red axis. The second set of symmetries is a rotation by π around the green axis from the center of the knot through the middle of an ear as seen in Figure 3.2. The generators of the symmetry group are one of each type stated before. With these two elements we can produce the whole group. The figure eight knot has a slightly less obvious symmetry group. A rotation by π about the red axis in Figure 3.3 is a first symmetry. Another symmetry is obtained by rotating the knot by $\pi/2$ around the same axis and reflect it through the transparent blue plane. These are the generators of a symmetry group with four elements. Finally the 5_1-knot can be mapped to itself by a rotation by π around the red axis as depicted in Figure 3.4. Note that we did not include the reparametrization in our explanation since it is intrinsic to the curve and not visualized in the figures.

Lemma 3.2 (Fourier coefficients of 3_1, 4_1 and 5_1)**.** *Assume that symmetry Conjecture 3.1 is true. Then the Fourier coefficients must fulfill the following equations:*

- *Trefoil: For $i \in \mathbb{N}$:*

$$\begin{aligned} a_{3i+1,1} = -b_{3i+1,2} \in \mathbb{R}, \quad & a_{3i+1,2} = a_{3i+1,3} = b_{3i+1,1} = b_{3i+1,3} = 0, \\ a_{3i+2,1} = b_{3i+2,2} \in \mathbb{R}, \quad & a_{3i+2,2} = a_{3i+2,3} = b_{3i+2,1} = b_{3i+2,3} = 0, \\ b_{3i+3,3} \in \mathbb{R}, \quad & a_{3i+3,1} = a_{3i+3,2} = a_{3i+3,3} = b_{3i+3,1} = b_{3i+3,2} = 0. \end{aligned}$$

- 4_1: *For* $i \in \mathbb{N}$:

$$\begin{aligned} a_{4i+1,1} = b_{4i+1,3}, a_{4i+1,3} = -b_{4i+1,1} \in \mathbb{R}, \quad & a_{4i+1,2} = b_{4i+1,2} = 0, \\ a_{4i+2,2}, b_{4i+2,2} \in \mathbb{R}, \quad & a_{4i+2,1} = a_{4i+2,3} = b_{4i+2,1} = b_{4i+2,3} = 0, \\ a_{4i+3,1} = -b_{4i+3,3}, a_{4i+3,3} = b_{4i+3,1} \in \mathbb{R}, \quad & a_{4i+3,2} = b_{4i+3,2} = 0 \\ & a_{4i+4} = b_{4i+4} = 0. \end{aligned}$$

- 5_1: *For* $i \in \mathbb{N}^*$:

$$\begin{aligned} a_{i,1} = a_{i,3} = b_{i,2} = 0, \\ a_{i,2}, b_{i,1}, b_{i,3} \in \mathbb{R}. \end{aligned}$$

Proof. Note that two Fourier knots describe the same curve iff all coefficients coincide, since $\sin(f_i t)$ and $\cos(f_i t)$ form an orthogonal basis. Furthermore if some curve γ is G-symmetric, then $\gamma = \gamma_G$.

- Trefoil: Let γ be a G_{3_1} symmetric curve. By Lemma 3.1 there exists for each $g \in G_{3_1}$ and each frequency f_i a 6-dimensional matrix M_g^i that acts on the coefficient vector such that $M_g^i \begin{pmatrix} a_i \\ b_i \end{pmatrix}$ is the i-th coefficient vector of $g.\gamma$.

  Computing

$$M_i = \frac{\sum_{g \in G_{3_1}} M_g^i}{|G_{3_1}|}$$

  yields:

$$M_{3i+1} = \begin{pmatrix} 1/2 & 0 & 0 & 0 & -1/2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ -1/2 & 0 & 0 & 0 & 1/2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}, \quad M_{3i+2} = \begin{pmatrix} 1/2 & 0 & 0 & 0 & 1/2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1/2 & 0 & 0 & 0 & 1/2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

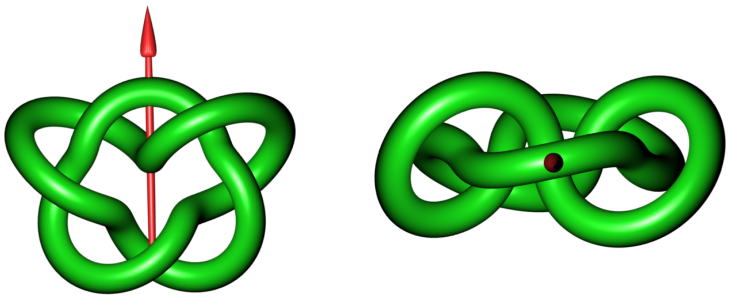


Figure 3.4: The symmetry group for the 5_1 knot has only a single generator, which is a rotation of angle π about the red axis.

and

$$M_{3i+3} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

Note that $M_i \begin{pmatrix} a_i \\ b_i \end{pmatrix}$ is the i-th coefficient vector of $\gamma_{G_{3_1}}$ and by the above reasoning we have

$$\begin{pmatrix} a_i \\ b_i \end{pmatrix} = M_i \begin{pmatrix} a_i \\ b_i \end{pmatrix}$$

which yields the wanted relations.

- 4_1 : For this knot the matrices are

$$M_{4i+1} = \begin{pmatrix} 1/2 & 0 & 0 & 0 & 0 & 1/2 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1/2 & -1/2 & 0 & 0 \\ 0 & 0 & -1/2 & 1/2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1/2 & 0 & 0 & 0 & 0 & 1/2 \end{pmatrix}, \quad M_{4i+2} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$M_{4i+3} = \begin{pmatrix} 1/2 & 0 & 0 & 0 & 0 & -1/2 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1/2 & 1/2 & 0 & 0 \\ 0 & 0 & 1/2 & 1/2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ -1/2 & 0 & 0 & 0 & 0 & 1/2 \end{pmatrix}, \quad M_{4i+4} = 0,$$

where M_{4i+4} is the zero matrix. These matrices imply the relations.

- 5_1 : And for the final knot we compute

$$M_i = \frac{\sum_{g \in G_{5_1}} M_g^i}{|G_{5_1}|}$$

to obtain

$$M_i = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

□

If one of the above knots is properly aligned and then symmetrized with respect to their assumed symmetry group by the equation in Definition 3.3, then the Fourier coefficients will be of the form claimed by Lemma 3.2. The original shape used in the symmetrization process has to be close to ideal and close to the enforced symmetries, otherwise the resulting knot might not even be in the same isotopy class. A circle for example is an ideal shape and has all the above symmetries and its Fourier coefficients do fulfill the relations in Lemma 3.2, however it is none of the knot types we discussed.

In order to produce a Fourier knot in the correct isotopy class, the number of coefficients approximating the knot has to be taken into consideration. More complex knot shapes need more coefficients, since higher frequency terms are necessary.

Chapter 4

Visualization

In this chapter we introduce the main tools used to visualize properties of knotted curves. The first part will be about color gradients for *pp*, *pt* and *tt* plots, which are designed to enhance the important features of these plots. Standard color gradients can not resolve the fine detail and are replaced or adapted. In the second part we describe a 3D viewer program used to investigate and analyze curves and in particular ideal knot shapes. The various features of the viewer are illustrated on an example curve.

4.1 Color gradients

The visible color spectrum $\mathcal{C}$ as perceived by a human eye can be parameterized in different ways. An often used representation of the color space is the RGB cube (where RGB stands for **R**ed, **G**reen and **B**lue). But there exist many other mappings like HSV (hue, saturation, value), CMYK or YCbCr which are used in different applications and are based on things like sensitivity to light of the human eye or the color spectrum of a computer or television screen [31, 16, 74, 35]. We will use the RGB space to illustrate gradients; it is analogous for other color spaces. We define RGB to be the unit cube $[0,1] \times [0,1] \times [0,1]$ and introduce the mapping

$$\text{RGB} \longrightarrow \mathcal{C}.$$

which symbolically means, that a color in $\mathcal{C}$ is described by a 3-tuple $(r,g,b) \in RGB$.

Definition 4.1. *A color gradient is a path $c(s), s \in [0,1]$ through the RGB cube from $c(0) = (r_0, g_0, b_0)$ to $c(1) = (r_1, g_1, b_1)$.*

Example 4.1. *A widely used path $c_1(s), s \in [0,1]$, is the rainbow color gradient or full*

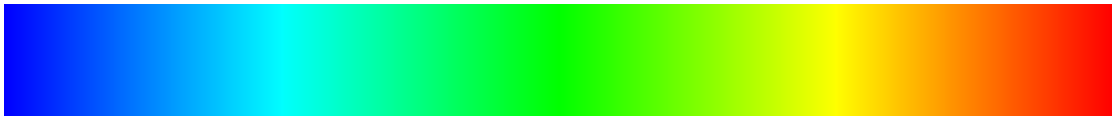


Figure 4.1: Rainbow or full saturation gradient running from blue to red over four edges of the RGB cube.

saturation spectrum parameterized by

$$c_1(s) = \begin{cases} (0, 4s, 1) & \textit{if} \quad s \in [0, \frac{1}{4}) \\ (0, 1, 2 - 4s) & \textit{if} \quad s \in [\frac{1}{4}, \frac{1}{2}) \\ (4s - 2, 1, 0) & \textit{if} \quad s \in [\frac{1}{2}, \frac{3}{4}) \\ (1, 4 - 4s, 0) & \textit{if} \quad s \in [\frac{3}{4}, 1] \end{cases}$$

With this gradient we travel from the color blue along four edges of the cube to the red corner (see Figure 4.1).

Color gradients are widely used in designer and drawing applications. Our interest in them is a little more scientific. When we are dealing with functions of one variable, then there is no special need for color, since all the information can be drawn in a $2D$ graph or plot. This is no longer true for functions of more than a single variable. Here it is necessary to mimic the supplementary dimension larger than two with a color. Consider a function $f(x, y)$, $x, y \in [-1, 1]$ whose values are between 0 and 1. Then we can map every pair (x, y) to a color $c(f(x, y))$, where $c(s)$ is a gradient. It is easy to read such a plot when using for example the rainbow gradient. High valued regions are in red and low valued in blue (hot and cold). This is sometimes sufficient to study and understand a function f. Suppose now that f has a lot of small, local variations. Using once again the rainbow gradient, we are not able to resolve the fine structure since the eye can not distinguish colors that are close in the RGB cube. The eye and brain are trained to spot contrasts and we need to introduce that in the way we construct appropriate gradients. The study of a given function is usually done in several steps and most of the time we would initially have some idea of how it looks. This information can then be used to construct an appropriate gradient to emphasize the small scale structure of the function. If we know or presume that f varies a lot in a region with small values, then we can tailor our gradient to change the color much faster in this regions to make the variations visible to the eye. This is not always trivial and might involve a thorough investigation of f beforehand. Fine tuning a gradient as we obtain more information is usually the way to proceed.

A gradient that well resolves some of the functions studied in this thesis is given by the RGB path $c_2(s), s \in [0, 1]$

$$c_2(s) = \begin{cases} c_*(s) & \text{if} \quad s \in [0, 1 - \varepsilon) \\ blend(h(s), c_*(s), c_1(h(s))) & \text{if} \quad s \in [1 - \varepsilon, 1] \end{cases}$$

Figure 4.2: Gradient $c_2(s)$ defined in the text, with $\varepsilon = 1/10$ and a *blend* function that linearly mixes in a rainbow color gradient in the interval $s = [9/10, 1]$.

where $blend(s, x, y)$ is a function that smoothly blends the colors $x, y \in RGB$ for $s \in [0, 1]$ with $blend(0, x, y) = x$ and $blend(1, x, y) = y$. The blending can be a linear interpolation between the two colors. In this thesis $blend(s, x, y)$, $s \in [0, 1]$ is used as follows

$$blend(s, x, y) = \begin{cases} 4s\,y + (1 - 4s)x, & s \in [0, 1/4), \\ y, & s \in [1/4, 1]. \end{cases}$$

The function $h(s)$ takes s from the interval $[1 - \varepsilon, 1]$ to $[0, 1]$. The path $c_*(s)$ is another gradient and given by

$$c_*(s) = (s, \frac{1}{2}(\sin(2\pi f s - \pi) + 1), \frac{1 - s}{2}(\sin(2\pi f s) + 1)), \qquad s \in [0, 1].$$

Here, f is the color oscillation frequency. The gradient $c_2(s)$ is depicted in Figure 4.2. This gradient will be very helpful to study *pp*, *pt* and *tt* plots of ideal knot shapes. All the action will be focused in a small interval close to 1. That is exactly where we mix in a rainbow color gradient in c_2.

Remark 4.1. *A gradient does not need to be a continuous function. We define the discontinuous pulse function $\Pi(x)$ by*

$$\Pi(x) = \begin{cases} 0 & \text{for } |x| > \frac{1}{2} \\ 1 & \text{for } |x| \leq \frac{1}{2} \end{cases}$$

A black and white contour gradient is the path

$$\Gamma_i(s) = \sum_{j=1}^{N} \Pi((s - x_j)/b_j), \quad i = 1, 2, 3,$$

where we put N pulses at x_j and each pulse has width b_j and the pulses should not overlap. The coordinates for the RGB color are either all zero (black) or all one (white). Applying this gradient to a function f is equivalent to using a standard contour plotter for the values x_j. We can control the width of the contour-lines with b_i. If a visually appealing image is required, then one might have to smooth the step function Π so that different levels of gray are also available and creates an anti-aliasing effect (color smoothing) at the borders of the contour plot. Anti-aliasing is the art of removing edgy artefacts due to pixelization in an image (see for example [70]).

Remark 4.2. *Sometimes it is useful to deal with a cyclic gradient. For a path $\Gamma(s), s \in [0, 1]$ we have the condition $\Gamma(0) = \Gamma(1)$. For the rainbow gradient $c_1(s)$ we simply*

connect the blue and red corner and we have a cyclic rainbow gradient. This is especially useful if the function has some periodicity. If we want to color the arc-length of a closed curve, it makes sense to use a cyclic gradient in order to get a smoothly varying color along the curve.

Remark 4.3. *The use of gradients like $c_2(s)$ can lead to formation of Moiré patterns. When the sampling of a function with a gradient is too low, then artificial shapes and patterns might become visible that do not reflect the real behavior of the function. A car wheel spinning backwards in movies is a trivial example of such a pattern, since the camera filming at a fixed frame rate can not resolve the correct motion of the wheel. Moiré patterns will arise later when the gradients introduced here are applied to ideal knot shapes.*

4.2 `curview`

Throughout the work for this thesis we investigated different properties of knotted curves. The different aspects of the analysis of curves led to a growing set of tools and utilities that have then been compiled into a C++ library called `libbiarc`, where the most frequently used functions are available as an API. For availability of the library see Appendix A. The core library provides functions to interpolate point-tangent data with biarcs, access information like curvature and torsion, compute the length and thickness, and much more. The native data format for curves used by `libbiarc` is PKF, initially designed by B. Laurie. The specification for the PKF file format is described in [10]. There is another widely used program to manipulate and visualize knots called `KnotPlot` [60] which has a different data format than PKF. The specific approximations used in this thesis forced us to build another software package.

Our software eventually grew to a collection of tools to manipulate and analyze open and closed curves and links. The library contains small programs to generate well known curves, such as circles, ellipses, helices, etc. Conversion tools help to prepare data that can be used in other software packages, for example 3D studios such as Blender [28], which has been used to render a lot of the images in this thesis. In an early stage of this work we mainly used scripts, and the rendering programs Pixie [3] and PovRay [51] to render images. The ease of use particular to Blender moved us away from them. The library can still be compiled with Pixie support, but this will not further be explained in this thesis.

The package also includes a viewer program called `curview` which uses Coin3D [33], a free implementation of the OpenInventor [48, 73] specification. OpenInventor is a high level graphics suite based on OpenGL [44, 65] to quickly develop graphics applications. The event management and user interface is done by SoQt [33], which is the binding element between the operating system, its window manager and the OpenInventor implementation used, in this case Coin3D. SoQt is based on Qt [45] a

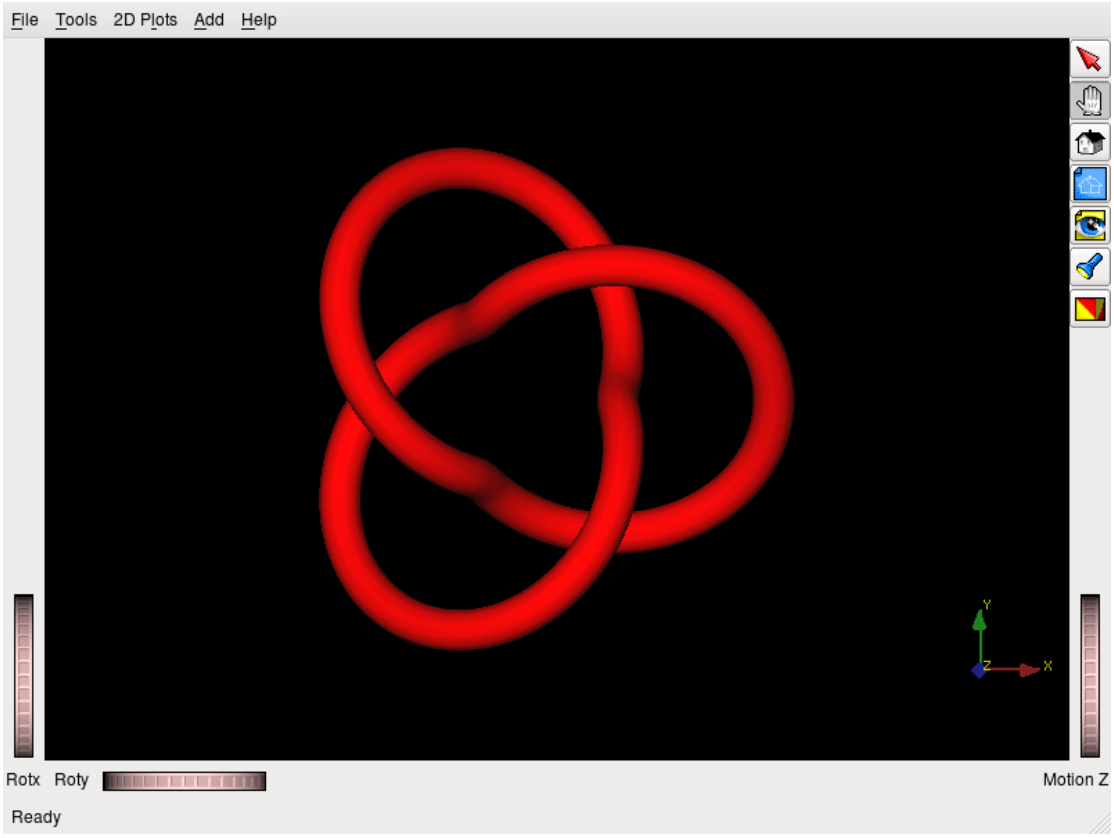


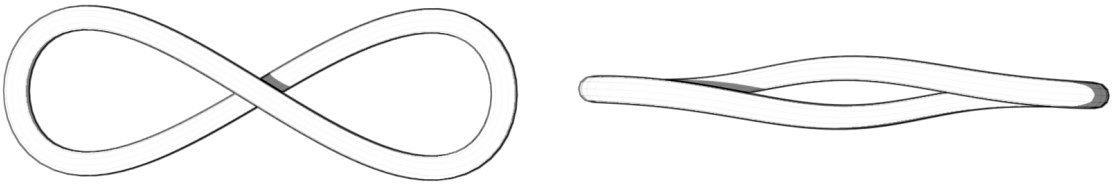
Figure 4.3: Screenshot of the `curview` interface with a trefoil.

very complete user interface framework.

We will focus on the viewer in greater detail since the other parts of the `libbiarc` are well documented and too voluminous to be listed here. A link to the online documentation or a PDF version (292 pages at the time of writing), the command line options and the key bindings for the viewer can be found in Appendix A.

The viewer interface is depicted in Figure 4.3. Standard actions as known from other 3D modelling programs are rotating and translating the curve object with left and middle mouse click plus dragging. A right click opens a pop-up with different viewer settings. Notice that a lot of the key bindings are not available in the viewer menu, and there are a few things in the menu that do not have key bindings.

The viewer has a minimalistic curve modeling feature when in `BIARC` view mode (refer to Appendix A.2 for an explanation of the different view modes). The button showing a red arrow in the right panel activates the picking mode. In this mode it is possible to select data points or tangents and edit them. The biarc representation is interactively updated. If the curve is not closed, then the `RETURN` key appends a new data point with tangent at the end of the current curve.



(a) (b)

Figure 4.4: The "infinity" example curve as defined in the text used to illustrate the functionality of `curview`.

The menu items in `File` load PKF curves, clear the scene, or save the current state of a curve. The second last menu `Add` is to load OpenInventor scene files (for the specification of `.iv` see [73]). Adding such scenes is useful to show supplementary information about a knot like externally computed normals or to simply add a sphere for curves lying on a sphere. In what follows we will explain the menus `Tools` and `2D Plots` in more detail.

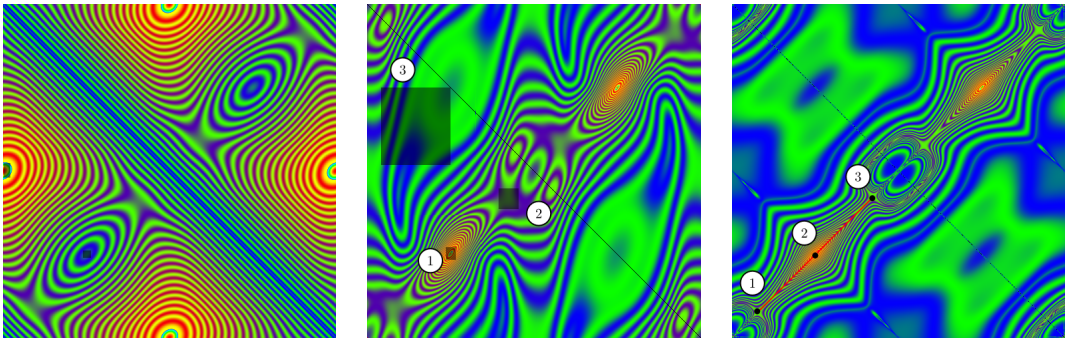
To that end we introduce the infinity or eight-shaped curve depicted in Figure 4.4. The program `inf` in the `objects` directory of `libbiarc` produces this shape. The parameterization is given by

$$\gamma(t) = \begin{pmatrix} a\,\cos(2\pi\,t) \\ b\,\sin(4\pi\,t) \\ c\,(e^{-(4\pi\,t-3\pi)^2} - e^{-(4\pi\,t-\pi)^2}) \end{pmatrix}, \qquad t \in [0,1], a, b, c \in \mathbb{R}. \tag{4.1}$$

The parameter values for the example curve are $a = 1, b = .3$ and $c = .1$. We will sometimes refer to the two outer loops as "ears" and the middle part as the "crossing", even though the curve has no self intersection as seen in Figure 4.4. This example curve is now used to describe the remaining features of the viewer.

4.3 pp, pt and tt plots

Before we deal with the second entry in the `Tools` menu, we move to `2D Plots`, where we can open a new window for each of the three plots, pp, pt or tt. The plots are generated for the current state of the curve. This means that if a user modifies the curve, then the plots update and reflect the changes in the active plots. The three plates in Figure 4.5 are the respective plots for our example curve.



(pp) (pt) (tt)

Figure 4.5: 2D plots of pp, Δ/pt and Δ/tt functions for the example curve. Darker colored areas, and the labels correspond to regions discussed in the text.

Here we need to say that the pt and tt plots are normalized such that the values are in $[0, 1]$. The graphs $pt(s, \sigma)$ and $tt(s\sigma)$ are therefore in a unit cube, since $s, \sigma \in [0, 1]$. The values used in the plots are Δ/pt and Δ/tt. This is a natural scaling, since the thickness Δ is a lower bound for pt and tt. And for a circle or a sphere radius that goes off to infinity, the value is zero. The pp plot does not have this scaling, we simply stretch the gradient between the smallest and the largest value for the pp plot[1]. These plots in 2D with the special color gradient are a start, but usually not enough to understand their relation to the actual curve in 3D. This is why the viewer lets the user click on the different plots and draws the appropriate object in the 3D scene displaying the curve. In other words, if the user clicks on a pixel (x, y) in the pp plot window, then the corresponding chord is drawn in the 3D scene. Our intuition for a distance plot is usually quite good, since we can tell by eye where the maximum and minimum values for pp will occur along the curve. To demonstrate this anyway we refer to Figure 4.6, where we selected two regions in the pp plot (small darker colored rectangles in Figure 4.5 (pp)) which can be done by dragging a square over the pp plot instead of picking a single pixel. The origin, i.e. $(0, 0)$, of the plots is in the upper left corner of the picture and the parameters s, t are arc-length. Several points or regions can be visualized in the scene by pressing shift and repeating the selection. For a highlighted area in the plot we do not draw the whole rectangle, but only a cross that runs vertically and horizontally in the middle of the rectangle. Drawing the entire rectangle would quickly fill the scene with line segments and then it is hard to see anything at all. The selected regions for the example curve correspond to areas close to the maximum and minimum distance values along the curve. Intuition already tells us that the maximum for pp must be between the two ears and the minimum in the crossing region. This is what we see in Figure 4.6.

Studying a pt plot is at the beginning not a trivial task. The same functionality as for pp is available for pt. As mentioned several times, pt is the radius of a circle tangent

[1] We use color gradient $c_2(s), s \in [0, 1]$ from Section 4.1, where for pp, we linearly map the interval $[0, pp_{max}]$ to $[0, 1]$. Since we plot Δ/pt and Δ/tt, these values can naturally be used with gradient $c_2(s), s \in [0, 1]$

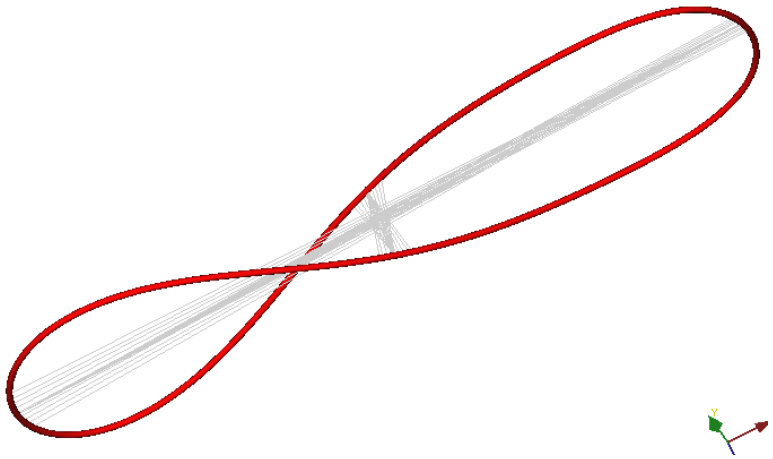
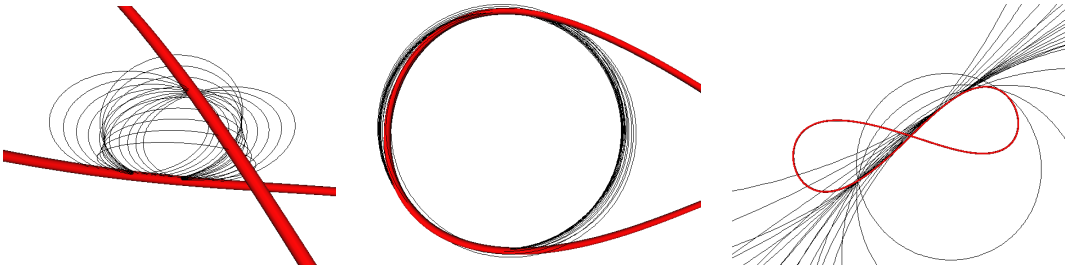


Figure 4.6: Visualization of the *pp* chords corresponding the the small darker areas in Figure 4.5 (pp).

at one point of the curve and running through another. So the corresponding object to visualize is a circle. Selecting a region in the *pt* plot draws circles the same way as it did the chords for *pp*. The visualization in Figure 4.7 matches the selected areas in Figure 4.5 (pt). Let us start with the region labeled with (1). The circles in 3D are drawn in Figure 4.7 (1). This is the neighborhood where the minimum of *pt* is achieved. Walking away from the crossing point to either side, the radii of the circles increase. Note that the tangency of the circles is at the curved bit running behind. If we move to one of the loops with both points, the tangency and the point through which the circle passes, then we reach the picture in Figure 4.7 (2), which corresponds to region (2) in Figure 4.5 (pt). Here the circles stay close to the shape of the ear, since it is close to circular, which is why this region of the *pt* plot is rather flat with a radius of the circles larger than at the minimum of *pt*. The circles get really large if the point and the tangency is on one of the straight segments between the loops, since they are almost co-linear. The radii decrease as soon as we either move the tangency or the point to a loop, since we move away from the co-linearity. With the *pt* plot window and the corresponding visualization it is straightforward to investigate the *pt* properties of the example curve.

The last candidate is the *tt* plot where spheres are drawn in the viewer window. By picking any point or region, the viewer will behave in the same way as with *pt*, only with spheres. The minimum of *tt* is the same as *pt* and Figure 4.8 (1) demonstrates the position of this sphere. In the *tt* plot this is the center of the rainbow colored region (see Figure 4.5 (tt)). If we pick two tangents in one of the loops, the center and radius of the sphere do not vary much. The two spheres corresponding to the points (1) and (3) in the *tt* plot are visualized in Figure 4.8 (2). If in this same plot we follow the red line, which is the anti-diagonal, then we observe what is drawn in Figure 4.8 (3). Note that with the chosen parameterization of the curve, $tt(0,1)$ is the outmost point in one of the loops and the anti-diagonal $tt(s,1-s), s \in [0,1]$ means that we symmetrically



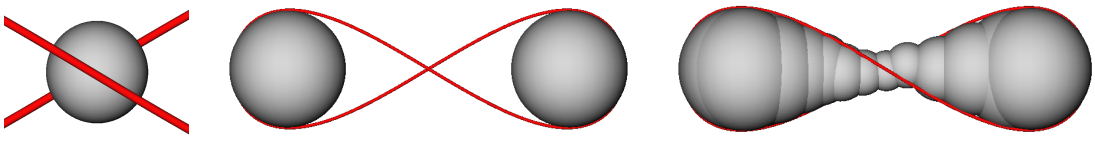
(1) (2) (3)

Figure 4.7: Visualization of the *pt* circles of the three regions in Figure 4.5. From left to right, close to the minimum of *pt* (a), large, flat plateau in the *pt* landscape (b) and *pt* values for almost co-linear point-tangents (c).

move away from this point in opposite directions. Another way of seeing this is that one tangent runs in the direction induced by the orientation of the curve and the other tangent runs backwards in the opposite direction. So the sphere starting in one loop shrinks to become the minimal sphere and grows again till it reaches the other loop of the curve. Since *tt* plots only involve tangents of the curve, this plot is the most aggressive one in the sense of small variations. If we wiggle at a tangent, the *tt* plot endures considerable changes. This is why numerical artefacts (noise) might appear close to the diagonal. This noise is also present in the *pt* plot, but, due to the resolution, not as visible.

We would like to point out that *pt* and *tt* plots make C^2 details visible. Consider a straight line with a small variation not visible to the eye. The *pt* and *tt* plots will make this feature visible. This makes the use of these plots on curves in $\mathbb{S}^3$ interesting.

This kind of analysis gives a lot of insight for a given curve and has been extensively used to study ideal knot shapes presented in sections to come. As explained in Section 2.1, the thickness of a curve is the infimum of the *pt* function. So aside the viewer plots in 2D and its ability to draw corresponding objects in the viewer window, we can also visualize the plots in 3D to see where the minimum of *pt* actually is, and what its neighborhood looks like. The plates in Figure 4.9 display the three plots stacked one on top of the other starting with *tt* at the bottom, *pt* in the middle and finally *pp* on



(1) (2) (3)

Figure 4.8: Visualization of the *tt* spheres, where (a) is the minimum of *tt* (labeled (1) in Figure 4.5 (tt)). In (b) the spheres for the labeled points (1) and (3) are shown and finally an envelope of spheres corresponding to the red path in Figure 4.5 (tt).

top. Earlier we explained that the minimum of pt or tt corresponds to the maximum of Δ/pt or Δ/tt, which is close to 1. In the plots in 3D we invert these values, namely $\tilde{pt} = (1 - \Delta/pt)$ and $\tilde{tt} = (1 - \Delta/tt)$, so that the pt minimum is actually a real minimum in the 3D terrain as well.

Using the plots in Figure 4.5 and the 3D height-maps in Figure 4.9 we can identify the regions we investigated before in the viewer. The pp function has a global minimum along the diagonal, where the distance is zero. Further we can spot a single hill and a dip in the landscape (pp is symmetric) and they correspond to the maximum between two loops and the local minimum between the crossing mentioned above.

This local minimum becomes global in the pt plot. Region (2) in Figure 4.5 (pt) is where the circles look similar to the loop and the 3D terrain shows a small flat plateau at this location. Walking uphills from region (2) to region (3) we reach another, larger plateau. The further we move up, the larger the radii of the circles. On the plateau the radii go to infinity. There are three other striking landmarks in the center of the terrain. The small hill exactly in the middle of the plot corresponds to the radius of curvature in the middle of the ear and the two dips close by (there are actually four along the diagonal) are the smallest local radius of curvatures of the curve. For an appropriate value of the parameter a, larger than the current one, in Equation (4.1), it is possible to bring the minimal local curvature at these four points down to the global minimum of pt. Then the thickness of the curve would at the same time be limited by local curvature and global contact. We then say that curvature is active.

A look at the tt landscape indicates that the interesting features are all close to the anti-diagonal. If one follows the red line in Figure 4.5 (c) on the tt terrain, then this path goes from a little plateau in the loop region to the global minimum and back to a little plateau. This is the smoothly changing sphere envelope visualized in 4.8 (c). Away from the anti-diagonal we quickly reach rather large flat planes where the radii of the spheres become very large. This is because the curve is almost in a plane.

4.4 Visualization of curves in $\mathbb{S}^3$

Investigating a curve in $\mathbb{S}^3$ is not as immediate as in our well known $\mathbb{R}^3$. In Section 5.6 we will introduce stereographic projections to "see" what the curve looks like. These projections are not required for the type of visualization shown here. Consider a curve γ on the 2-sphere $\mathbb{S}^2$. We can separate this sphere in a northern H_N and a southern hemisphere H_S [72]. Then we define the two projections

$$m_N : H_N \to D_N, \quad m_S : H_S \to D_S$$

where D_N, D_S are the two disks given by $\{p \in \mathbb{R}^2 | |p| \leq 1\}$, lying in the equatorial plane of the sphere. Figure 4.10 shows this procedure and the resulting two disks, where points

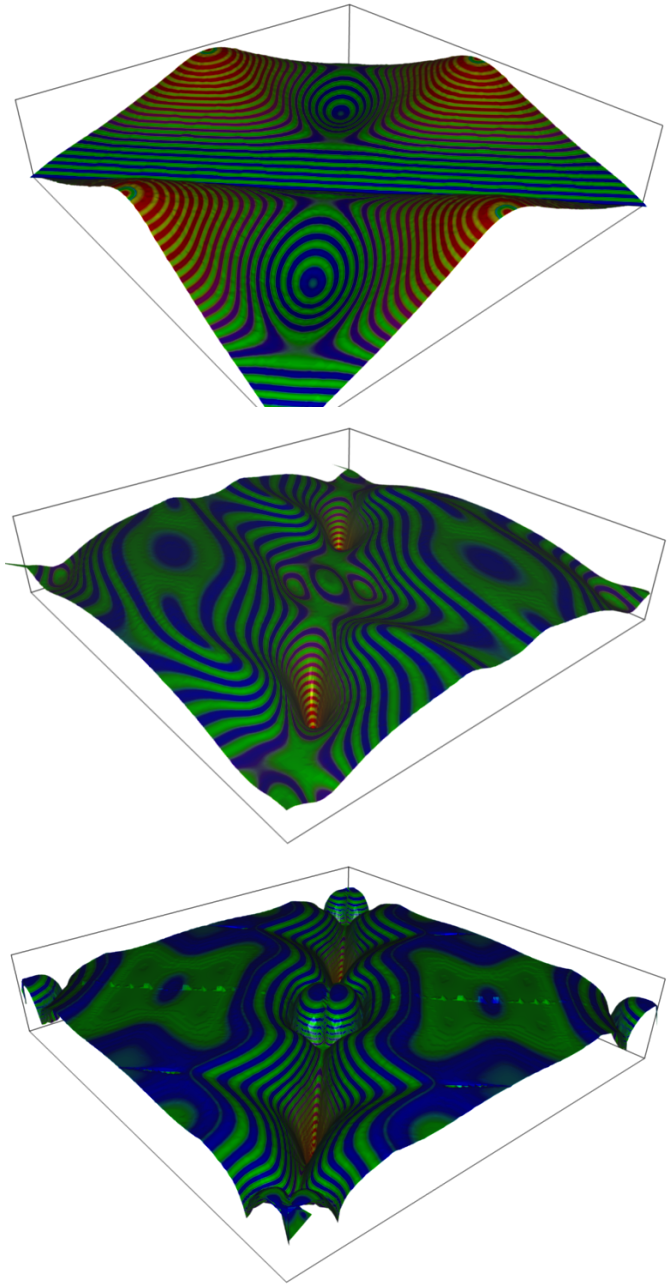


Figure 4.9: From top to bottom : pp, $\tilde{pt}$ and $\tilde{tt}$ landscape for the infinity curve. The pt and tt 3D visualization actually lies in a cube of edge size 1, but this would not be very useful. The height-fields are scaled down to about 1/4th of the other edge lengths.

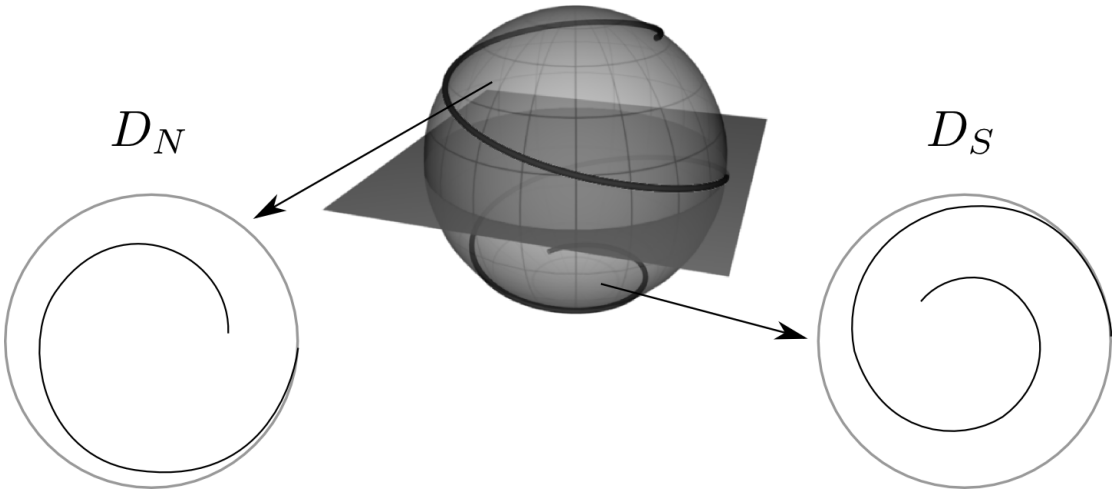


Figure 4.10: A curve on the 2-sphere decomposed into two hemispheres is projected onto two disks shown to the left and to the right of the sphere.

on the border ∂D_N and ∂D_S are identified. This means that a curve leaving disk D_N at p reenters disk D_S at the same point.

Inspired by the three dimensional case we can carry this construction over to curves on $\mathbb{S}^3$ [72]. Here the hemispheres are projected to two balls $\mathcal{B}_N$ and $\mathcal{B}_S$, where $\mathcal{B} = \{p \in \mathbb{R}^3 | |p| \leq 1\}$. In Figure 4.11 we show this on a closed curve in $\mathbb{S}^3$. The points on the borders of the balls are identified similar to the $\mathbb{R}^3$ case. The rainbow gradient coloring helps to follow the curve. The tool `s3viz` included in the `libbiarc` constructs two PKF files containing the components of the curve for each ball. They can then be viewed and analyzed in `curview`. We have not used this visualization. One of the reasons is that the brain needs practice to understand this way of seeing $\mathbb{S}^3$.

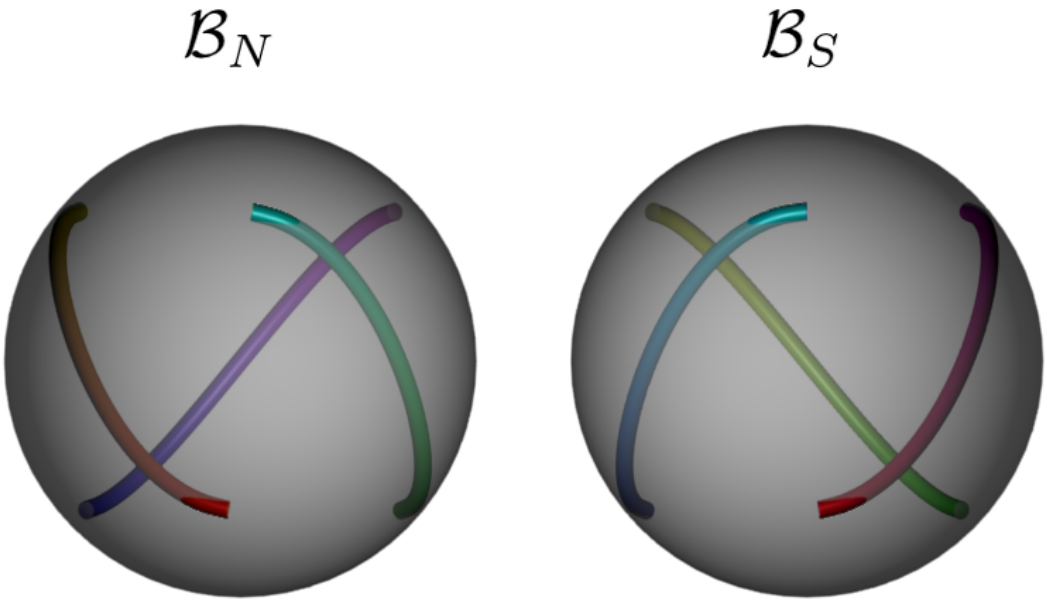


Figure 4.11: Separation of $\mathbb{S}^3$ into two hemispheres, where each hemisphere corresponds to a ball in $\mathbb{R}^3$ as shown. The curve visualized here is the trefoil on a Clifford torus discussed in Section 6.5 and arclength colored. Unlike in the 2D version, it is necessary to spin the two spheres in 3D to see how close the curves are from the center of the spheres.

Chapter 5

Computations of ideal knot shapes and the `libbiarc`

In Chapter 2 we have introduced the concept of an ideal knot K which is a minimizer with respect to its ropelength L/Δ in the isotopy class $[K]$. Now we will consider the computational side of this problem. Finding such a minimizer in a given isotopy class is also called tightening, shrinking, inflating or optimizing a knot. This nomenclature comes from the fact that we can see our problem of minimizing L/Δ from at least two different perspectives. Either we fix the radius of the knot and reduce the length of the curve (tightening, shrinking) or we fix the length of the curve and try to increase the radius of the tube (inflating). In what follows we will review known algorithms that perform these optimizations. The problem of ideal knots is in some sense a particular case of an optimal packing problem, where one tries to put N objects in the smallest area or volume possible. In the case of ideal knots it is packing one object, namely a tube, but the whole interest is determining the optimal shape of the deformable tube.

5.1 SONO

The first method is the so called Shrink-On-No-Overlap (SONO) algorithm as introduced in [53]. Inspired by the physical tightening of a piece of rope, the algorithm is based on rescaling a given knot and applying a repulsive hard-sphere potential on interacting bits of the curve. This algorithm is well suited for quickly finding a configuration of the knot in the neighborhood of the ground state, i.e. the state such that the knot minimizes the energy L/Δ. This algorithm is appealing for creating animations of the knot tightening process. During this thesis SONO has been reimplemented to quickly reach knot shapes that were then further processed with other algorithms. The original implementation of SONO was done in Fortran. Our version is in C++. We will briefly review the principle of the algorithm following the presentation in [53].

Input: $x_1, \ldots, x_N, D, Overlaps, ShrinkFactor, \varepsilon, Tol, Iter, M$
Output: $x_1, \ldots, x_N$

$L \leftarrow$ `Length`$(x_1, \ldots, x_N)$
$l \leftarrow L/N$
skip $\leftarrow (\pi D)/(2l)$
nn $\leftarrow$ `NearestNeighbours`$(x_1, \ldots, x_N$,skip,$\varepsilon)$
for $j \leftarrow 1$ **to** $Iter$ **do**
  every M times
    skip $\leftarrow (\pi D)/(2l)$
    `ShiftNodes`$(x_1, \ldots, x_N)$
    `ControlLeashes`$(x_1, \ldots, x_N)$
    nn $\leftarrow$ `NearestNeighbours`$(x_1, \ldots, x_N$,skip,$\varepsilon)$
  end
  overlap $\leftarrow$ `RemoveOverlaps`$(x_1, \ldots, x_N, nn, D(1+Tol))$
  if overlap $< Overlaps$ **then**
    `Shrink`$(x_i, l, ShrinkFactor)$
  end
end

Algorithm 1: Shrink on no overlap algorithm (SONO) used for knot tightening.

Let γ, embedded in $\mathbb{R}^3$, be the centerline of a thick knot in isotopy class $[K]$. A discrete piecewise linear version of this knot is given by N points x_i placed uniformly along γ. By uniformly we mean that the Euclidean distance of neighboring points is $l = L/N$. L is the length of the polygon formed by the points x_i. The pseudo-algorithm is given in Listing 1.

The algorithm takes as input a polygonal knot given by N vertices $x_i, i = 1, \ldots, N$ and a few more parameters. The main parameters are the imposed diameter of the thick knot D, the number of tolerated self-intersections $Overlaps$ and a scaling factor $ShrinkFactor$ (tightening of the curve). The secondary parameters are two tolerances ε and Tol, the number of iterations $Iter$, and an update parameter M. Their meaning will become clearer in the explanation of the subroutines. The first step is computing the piecewise linear approximation to length of the curve and the leash length l, which will determine the length of the segment between two vertices x_i and x_{i+1}. We only want global repulsion, which is why we use the local curvature as a bound to compute a *skip* value telling how many neighbor vertices can or have to be neglected. After that, the routine `NearestNeighbours` initializes an array of integers nn for the nearest neighbors. Every column in the table nn corresponds to a vertex x_i and contains the vertices $x_j, j \neq i$ that are at a distance closer than $D(1+\varepsilon)$. In the loop one recomputes every M times the *skip* value. The method `ShiftNodes` displaces by a small amount the points along the connecting edges; this is supposed to simulate thermal activity and is used to prevent getting stuck in a local minimum. The next routine `ControlLeashes`

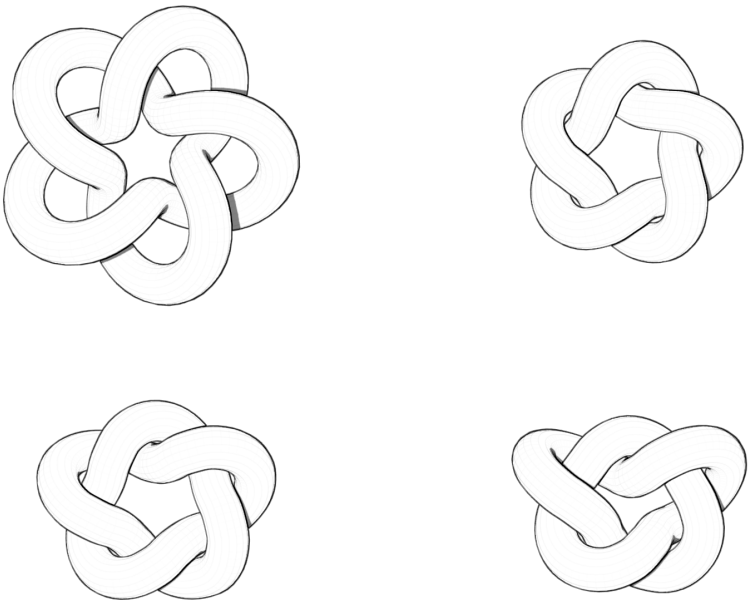


Figure 5.1: Four configurations of a 5.1-knot tightening run with SONO where the overlaps are pushed apart more than the prescribed diameter D.

readjusts the segment lengths between the vertices, since we would like to keep the polygonal knot nearly equilateral. Every M steps we also update the nearest neighbor table. The most substantial work is done in `RemoveOverlaps`, where we check for every vertex x_i if the distance constraint given by D is satisfied. This distance is only computed between x_i and the nearest neighbors as computed in `NearestNeighbours`. If a vertex x_j is closer than D to x_i, we symmetrically push them apart along the chord $c(x_i, x_j)$[1]. The resulting distance is then not D, but $D(1 + Tol)$. This helps to improve the convergence in an early state of the tightening process. The main loop is repeated $Iter$ times, but one could of course use a stopping criterion, which is, however, not easy to define. For a good convergence of the tightening process, the authors in [53] suggest for the number of vertices N a multiple of L/D.

SONO performs extremely well on shapes far away from ideal. Figure 5.1 illustrates a few snapshots of the tightening of the 5.1 knot. The algorithm quickly reaches a state close to ideal. The last frame in Figure 5.1 has been obtained after a computation time of about 30 seconds on a Thinkpad T60 with a 2GHz CPU. In the first stage of the shrinking process, once the tube starts touching itself it stays in that confirmation for quite a while. The knot has to break its symmetry and that can only be done due to the node shifting process, which emulates some entropy in the system to help the knot leave its symmetry constraint. As soon as the symmetry is broken, the knot quickly converges to a state, close to the ideal shape. There still seems to be a lot of space in

[1]For the chord we use the points as arguments and not arclength.

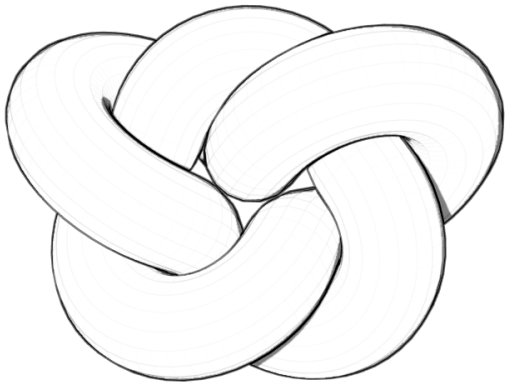


Figure 5.2: Approximately ideal 5.1 knot computed with SONO where the repulsion factor between interacting parts is reduced to zero. This removes the free space in the middle of the curve (see Figure 5.1).

the center of the knot, which is due to the fact that this quick run has been done with a tolerance $Tol = 0.1$ in the `RemoveOverlaps` step. Every time a contact occurs, the points are pushed 10 percent more apart than the prescribed diameter D, hence the space in the center. Gradually decreasing the tolerance Tol will then further improve the knot shape. After about 10 more seconds using the last shape as shown in Figure 5.1 and with a tolerance $Tol = 0$, we obtain what is shown in Figure 5.2. Note that there is no formal proof that this configuration is indeed close to ideal, but the image in Figure 5.2 looks convincing.

5.2 Gradient flows and `RidgeRunner`

The SONO algorithm is just like carrying out a physical experiment. It is however difficult to mathematically track what actually happens during this process. The process is actually a particular case of a constrained gradient flow, where one numerically computes the gradient of the energy and uses it to perform the optimization. In SONO, the energy is the length of the polygonal curve and its gradient would show where the curve can be shortened. This gradient usually points in the normal direction and depends on the curvature along the curve, since we prohibit self penetration and the problem of a pure curvature flow (see [49]) becomes a constrained gradient flow. These constraints are enforced in the overlaps removal stage in SONO. In the rest of this section we present a different approach of a constrained gradient flow as explained in [56, 9]. The implementation of this method is called `RidgeRunner`.

The thickness introduced in Chapter 2 requires a curve to have a certain regularity.

In particular, the definitions do not apply to curves approximated by linear segments. A polygonal knot V is a set of vertices $v_i, i = 1, \ldots, N$ joined by linear segments. The authors in [9] tackle the regularity problem by inscribing arcs (not biarcs) of circles in a polygonal curve when they compute the thickness. However, the optimization algorithm is done on the polygonal representation only. They show that their definition of polygonal thickness tends to the thickness of a curve. The polygonal thickness is the minimum of the curvature computed using angles between linear segments and the minimum of the length of double critical chords. The definition of curvature gives a bound on the double critical chords that have to be considered. This means that chords between points close together along the polygonal curve can be discarded. The curvature and chord distances are used as constraints $g_i = 0$ and the length L of the polygonal curve is the function to be minimized. The gradient ∇L is uniquely decomposed into an infinitesimal motion i, where the directional derivative of the constraint satisfies $g_i \geq 0$. The second set of motions, orthogonal to the infinitesimal motions, are called resolvable motions and are given by

$$r = -\sum \lambda_i \nabla g_i, \text{ with } \lambda_i \geq 0.$$

The optimization problem is now a constrained steepest descent method. In steepest descent one follows the negative direction of the gradient, in this case

$$-\nabla L = i + r.$$

Directly following $-\nabla L$ is not possible, since the constraints g_i have to be taken into account. The constrained steepest descent is given by the infinitesimal motions i. To compute i at each step of the minimization, it is necessary to solve the non-negative least squares problem

$$\min |Ax - \nabla L|, x \geq 0, \tag{5.1}$$

where A is a matrix of dimension $3N \times E$ with E the number of active constraints $g_i = 0$. Far from optimal, at every step only a few constraints are active and the matrix A is extremely sparse. The resolution of systems like Equation 5.1 has extensively been studied in numerical linear algebra [55]. Once x is computed, the constrained step to the next configuration of the polygonal curve is given by

$$i = Ax - \nabla L.$$

Due to numerics, after a step is taken in direction i, some of the constraints might be violated and [9] explains an error recovery for these cases. A motion in the opposite direction of the violated constraints g_i can be generated, since

$$A^T u = [\langle \nabla g_i, u \rangle] = [\nabla_u \, g_i],$$

the recovery motion u can be computed by solving $A^T u = -R$, where R is a column vector with the corrections for the constraints g_i. This global move u, in contrast to SONO, helps the knot to recover from inter-penetration even when the knot is already in a very tight configuration. `RidgeRunner` produces the best knot shapes we know of.

5.3 Metropolis Monte-Carlo

The two previous algorithms are deterministic in nature and the chance of getting trapped in a local minimum can not be excluded since the ropelength energy landscape seems very bumpy. Now we present another type of minimization algorithms based on a thermodynamic system and therefore stochastic processes. The first goal is to compute the equilibrium value of a thermodynamic quantity such as the density, entropy or free energy of a system of particles. Consider the state or phase space Ω and $P(x), x \in \Omega$ a normalized probability function $P(x)$ on Ω. The normalization condition is $\int_{x\in\Omega} P(x)d^n x = 1$. The mean value wrt $P(x)$ of a quantity A is defined as

$$\langle A\rangle = \int_{x\in\Omega} A(x)P(x)d^n x. \tag{5.2}$$

The standard Monte-Carlo method randomly samples Ω to compute $\langle A\rangle$. An often cited example is the estimation of π by Monte-Carlo integration. Uniformly sample the unit square $[0,1]\times[0,1]$ with N points x_i. The ratio of the points $|x_i| \leq 1$ and N yields an estimate of $\pi/4$. Depending on the problem, the phase space Ω might be huge. If on top of that, the probability distribution is extremely localized in one or more regions, uniform sampling wastes a lot of resources. The trick to reduce this problem is called importance sampling. The probability function $P(x)$ can be used to guide the integration but most of the time $P(x)$ is not explicitly known for a given system.

A modified version of the above has first been introduced by [41]. They replace the random sampling of Ω by a Markov chain, which is a path through the phase space where the current state only depends on the previous state. Another hypotheses motivated by statistical physics is to use the canonical ensemble for $P(x)$ and Equation 5.2 becomes

$$\langle A\rangle = \frac{\int_{x\in\Omega} A(x)e^{\frac{-E(x)}{kT}}d^n x}{\int_{x\in\Omega} e^{\frac{-E(x)}{kT}}d^n x} \tag{5.3}$$

where T is the temperature, k the Boltzmann constant and E the energy of a given state $x \in \Omega$. If a statistical system is ergodic, then the time average is equal to the space average wrt the equilibrium measure (or canonical ensemble) $P(x)dx$

$$\lim_{T\to\infty}\frac{1}{T}\int_0^T A(X(t))dt = \int_\Omega A(x)P(x)dx$$

for a Markov process $\{X(t)\}_{t>0}$. The importance sampling in the Metropolis algorithm is now achieved as shown in Algorithm 2.

The output of Algorithm 2 is $A_i, i = 1,\ldots,N$, where $A_i = A(X_i)$ for a time-discrete Markov chain $\{X_i\}_{i=1,\ldots,N}$. For the discrete value of the quantity A at step i and we have

$$\langle A\rangle = \frac{1}{N}\sum_{i=1}^{N} A_i.$$

Input: inital state $s \in \Omega$, iterations $Iter$, T
Output: $A_i, i = 1, \ldots, N$

for $i \leftarrow 1$ **to** $Iter$ **do**
  $A_i \leftarrow$ `EvaluateA(`s`)`
  $s^* \leftarrow$ `Move(`s`)`
  $\Delta E \leftarrow$ `Energy(`s^*`)` $-$ `Energy(`s`)`
  if $\Delta E < 0$ **then** `/* accept move */`
    $s \leftarrow s^*$
  else `/* accept with prob` p `*/`
    $p \leftarrow \exp(-\Delta E/(k\,T))$
    $r \leftarrow$`Random([0, 1])`
    if $r < p$ **then**
      $s \leftarrow s^*$
    end
  end
end

Algorithm 2: Metropolis Monte-Carlo algorithm is used to compute the value of a physical quantity A in a thermodynamic system at temperature T considered to be in an equilibrium state.

The importance sampling is due to the fact that we avoid entering high energy regions, since they do not contribute to the integral in Equation 5.2. We now observe that with this algorithm a system evolves towards a minimal energy state. It can therefore be used to find a global minimizer of some energy functional E. In the optimization process of ideal knots this energy functional is often the ropelength, other knot energies are discussed in [47, 57, 14].

5.4 Simulated annealing in $\mathbb{R}^3$

The original Metropolis Monte-Carlo runs for a fixed temperature T where the aim is to compute the equilibrium value of some quantity A in that thermodynamic system. By reducing the temperature T as a function of time, we can simulate the chrystallization of a physical object, for example water becoming ice. If this cooling process is done extremely fast, the crystal structure does not have time to properly arrange itself. Slowly cooling the material permits to build a perfect internal crystal structure. This is what simulated annealing does [32]. Simulated annealing is a Metropolis Monte-Carlo method, where the system is cooled down as we give it time to converge to an energetically optimal state. Initially a high temperature lets the system get out of large local valleys. But by slowly reducing the temperature the chances to escape from such valleys shrinks and the "chrystallization" process has time to converge to the global minimum of the phase space. Note that in the tightening process we do not have a phase change as is the case

for many physical systems although the symmetry breaking of the 5_1 knot might be said to do so. In simulated annealing we do not need an explicit gradient, as is usually the case for optimiztion problems. The pseudocode of simulated annealing is listed in Algorithm 3.

Input: inital state $s \in \Omega$, iterations $Iter$, T, cooling C
Output: final state $s \in \Omega$

for $i \leftarrow 1$ **to** $Iter$ **do**
  $s^* \leftarrow$ `Move(`s`)`
  $\Delta E \leftarrow$ `Energy(`s^*`)` $-$ `Energy(`s`)`
  if $\Delta E < 0$ **then** `/* accept move */`
    $s \leftarrow s^*$
  else `/* accept with prob` p `*/`
    $p \leftarrow \exp(-\Delta E/(k\,T))$
    $r \leftarrow$`Random([0, 1])`
    if $r < p$ **then**
      $s \leftarrow s^*$
    end
  end
  $T \leftarrow T(1 - C)$
end

Algorithm 3: Simulated annealing algorithm with decreasing temperature T.

Polygonal curves

A polygonal curve is a set of vertices $x_i \in \mathbb{R}^3, i = 1, \ldots, N$ connected by linear segments. This is similar to Section 5.2 with the difference that no arcs of circles are inscribed. Simulated annealing has been discussed for polygonal knots for example in [36, 56]. In the deterministic algorithms, shortening the curve is the preferred way of optimizing the ropelength. The simulated annealing approach seems to favor minimizing the ropelength L/Δ of a curve by keeping the length L fixed and moving the vertices of a curve to maximize the thickness Δ.

The phase space of this problem is $3 \times N$ dimensional, since every vertex has 3 degrees of freedom and we can displace any of the N vertices. The interval in which each coordinate can lie is the real line $\mathbb{R}$. In practice this interval is much smaller since we have to stay in the same knot class and moving a vertex too far away from the rest of the curve will introduce a sharp turn, the thickness goes down and the ropelength up. Hence it is not necessary to restrict the interval for the coordinates, the energy will take care of that.

If the knot is still far away from ideal, then the vertex moves should be rather large.

Large moves might change the knot type (isotopy class). So the usual way is to start with a knot already close to ideal - obtained for example with SONO - and annealing that shape with small moves.

The simulated annealing algorithm can be seen as Algorithm 2 where we follow a trajectory through the phase space - to reach a global minimum - as we decrease the temperature T by some cooling factor. Further we associate a step size with each vertex x_i which helps to adjust the move size in different regions along the curve. If a move involving a vertex x_i gets accepted, then we increase the step size for x_i and decrease it otherwise. So regions with a high acceptance rate are supposed to move quicker than regions where we encounter more resistance and are supposed to choose smaller moves. The stopping criterium is usually related to the temperature T, if T is too small, then we stop the algorithm.

Biarc curves

The main reasons why the authors in [11] moved away from a polygonal representation of knots are that ideal shapes are known to live in the space $C^{1,1}(\mathbb{S}, \mathbb{R}^3)$ and that curvature and arc-length of the curve are explicitly known. Biarcs do have the same $C^{1,1}$ regularity as ideal knots and are therefore good candidates to compute with. The configuration or phase space is now larger, since we have N point-tangent data pairs (p_i, t_i). So for a curve in $\mathbb{R}^3$ we have $6N$ different degrees of freedom and N constraints (unit tangents). The simulated annealing per se stays the same as in the polygonal case, the energy is the ropelength of the biarc curve and a move is either changing a point p_i or a tangent t_i. Here the step size is even more important since the points p_i are in $\mathbb{R}^3$ and the tangents t_i in $\mathbb{S}^2$. Points and tangents have completely different scales. Simulated annealing will usually first need some time to find appropriate scalings of point and tangent data.

Extending the simulated annealing program from the polygonal to the biarc case is straightforward. The shapes computed in [66, 11] were done with the annealing code from Ben Laurie. During the work of this thesis we developed a software library that includes simulated annealing code for biarc curves. In our implementation we provide base classes for the moves and the annealing process implementing functionality that remains the same for any simulated annealing problem. Developing new annealing problems by deriving from the base classes then required little time. The different approximations for the knot shapes, and the fact that we wanted annealing to run in $\mathbb{R}^3$ and $\mathbb{S}^3$ justified the class design. Appendix A.3 explains the base classes and presents a few example implementations.

Using this new code, several knot shapes up to nine crossings have been annealed. Part of the shapes used as starting condition came from Eric Rawdon et al.[2] as computed with `RidgeRunner` outlined in Section 5.2.

Fourier Knots

The difficulty of improving the knot shapes approximated either by linear segments or biarcs brought our attention to another representation, namely Fourier knots as introduced in Chapter 3. The main reason for switching to Fourier coefficients was actually the striking symmetry observed in the trefoil. Enforcing symmetries on knots based on linear segments or biarcs is rather difficult. As described in Chapter 3, a Fourier coefficient representation of a knot shape makes symmetrization very natural. We will now outline the various steps involved in converting a point-tangent curve to a Fourier knot and then annealing that shape.

Recall that a Fourier knot γ is given by

$$\gamma(t) = \sum_i \left(a_i \cos(f_i t) + b_i \sin(f_i t)\right), \quad t \in [0,1], \quad i = 1, \ldots, \tag{5.4}$$

where $f = 2\pi i$ is the frequency and the parameter t is chosen to be in the interval $[0,1]$. The coefficients $a_i, b_i \in \mathbb{R}^3, i > 0$ are

$$\begin{aligned} a_i &= 2\int_0^1 \gamma(t)\cos(2\pi i t)dt, \\ b_i &= 2\int_0^1 \gamma(t)\sin(2\pi i t)dt. \end{aligned} \tag{5.5}$$

We can write the curve $\gamma(t)$ as a Fourier series for each coordinate due to the periodicity in the parameter t. We ignore the coefficient a_0, since it is a constant vector corresponding to a translation of γ. In the case of links this term would play its role, but not for knots. Notice that the parameter t is not arc-length.

The first goal is to construct a member of some isotopy class $[K]$. The coefficients can be approximated by numerically computing the integrals in Equation 5.5. In practice we have a knot $\gamma(t)$ as a polygonal or a biarc curve which can be reparametrized such that $t \in [0,1]$. Then we compute the integrals on the right hand side of Equation 5.5 with some numerical integration scheme. This scheme does not have to be very sophisticated, a simple rectangle summation scheme is enough to get a coarse approximation for a_i and b_i. The number of coefficients needed depends on the complexity of the shape. From the coefficients $a_i, b_i, 0 < i < N$ we can synthesize the knot γ using Equation (5.4), and verify visually that the knot is still in the correct isotopy class.

Annealing Fourier knots is now done in the same fashion as point or point-tangent sampled curves. In each iteration step of the simulation, we pick a coefficient vector a_i or b_i, modify it and recompute the ropelength of the knot. Then the current state gets accepted or rejected according to the rules of simulated annealing.

An important detail that we omitted in the previous paragraph is how to compute the energy or the ropelength of a Fourier knot. It seems difficult to compute L/Δ based

directly from the Fourier coefficients. The length L is

$$L = \int_0^1 |\gamma'(t)|dt$$

but there is no expression for the thickness Δ. So we have to reconstruct a biarc representation using Equation 5.4. With the biarc curve we are able to efficiently compute Δ and therefore an energy for the annealing simulation. The simulation parameters are the usual ones like temperature and cooling rate. An additional parameter is the number of biarcs used to interpolate the curve. Notice that the Fourier annealing can actually be seen as a way to generate global moves for biarc annealing, since in the end the energy L/D is always evaluated on a biarc curve.

5.5 Ideal Fourier knot results

We will now present the Fourier trefoil, the figure eight and the 5_1-knot resulting from Fourier annealing as explained in the previous section. Initial runs have been executed with a low number of non related coefficients, since at that point we did not yet know how the symmetries suggested in Conjecture 3.1 would affect the coefficient structure as proved in Lemma 3.2. We discovered the pattern by first aligning the knot's symmetry axes with respect to the $x-$, $y-$ and $z-$axis. Then, during the annealing process, we regularly symmetrized the knot. Observing the coefficients during this process suggested, that the coefficients are not independent and a few coeffiecients even vanish. We recall that the coefficient structure for the trefoil is

cos			sin		
-A	0	0	0	A	0
B	0	0	0	B	0
0	0	0	0	0	C
		...			

where every row contains first the coefficient vector for the cos terms and then for the sin terms. Restricting annealing to only the independent and non zero coefficients of the trefoil gives a tremendous speed up compared to traditional annealing on all coefficients. The size of the configuration space is actually six times smaller. At the beginning we annealed a knot with only a few coefficient rows and added a coefficient as soon as we felt necessary. Notice that we proportionally increased the number of nodes used for the biarc representation compared to the number of Fourier coefficients. The necessity of constantly changing simulation parameters required a lot of "baby-sitting". In about a week of simulation time we could reach, and even beat the ropelength of the biarc trefoil (which we name γ_B) computed by J. Smutny in [66] where the time to obtain the shape took more than 6 months.

Interestingly, our Fourier trefoil (call it γ_F) given by only 6 independent double values had a ropelength of about twice the one for γ_B. If we build γ_F with 18 independent

values, then we are only 1‰ away from the ropelength of γ_B. The trefoil γ_B is made of 512 biarcs, which corresponds to $512 * 6 = 3072$ independent floating point double precision values. The Fourier trefoil with a ropelength lower than γ_B is given by only 165 doubles. The radius of curvature, pp, pt and tt plots are depicted in Figure 5.3. The symmetric pp and tt plots are in the same image. Then we show the 2D and 3D version of the pt plot. An important feature in the 3D plot is the two deep valleys, where the minimum of pt is achieved. The floor of a valley is rather flat, but actually has two minima for each value of s. In other words, if we pick an s in the 2D plot and follow the parameter t of $pt(s,t)$, then we encounter two minima, and this for every $s \in \mathbb{S}$. The flat plateaus close to the diagonal correspond to the three "ears" of the trefoil which are close to circular. As mentioned earlier, the diagonal of pt is the curvature of the curve and on the plateaus looks almost constant. Compare this to the radius of curvature plot in Figure 5.3 (a). The two peaks in the curvature profile are points where local curvature might be active. An interesting point is that for a low coefficient Fourier trefoil these peaks are very low, only when we added more coefficients, then the peaks would rise towards one. Gerlach observes in his thesis [19], that a knot's curvature profile can be manipulated locally without significantly reducing the thickness. Therefore, the curvature profile for two knots ϵ away from ideal can look very different. With a low number of Fourier coefficients we did observe Gibbs phenomena close to the peaks, which is an indication for discontinuity in curvature. Increasing the number of coefficients gradually removed these oscillations.

The results for the figure eight knot presented in Figure 5.4 clearly show the Gibbs oscillations close to the four peaks in the curvature plot (a). This knot has 96 independent Fourier coefficients. We did not run the 4_1 as carefully as the trefoil, but we could still go below the ropelength of the 4_1 knot presented in [66]. The oscillations in curvature for a lower ropelength is exactly what we explained before with Gerlach's argument. The 2D plots and the pt 3D version do not show valleys as for the trefoil but several disconnected "holes" that we will call space invaders due to their shape (see 6.3).

Figure 5.5 shows the results for the 5_1 knot which is not very converged. There are two reasons for this. We have spent most of the time to anneal and analyze the Fourier trefoil and we only have a single symmetry for the 5_1 knot, which is not an extreme speed up in annealing. But we can use its curvature plot in Figure 5.5 (a) to illustrate that the peaks can only form for a high number of Fourier coefficients. The little bumps in the curvature profile would correspond to the early stage of the three trefoil peaks. The 5_1 knot Fourier representation has 250 coefficients which seems large compared to the two previous knots. However, the effective number of coefficients for a symmetrized knot depends on its symmetry group. For the 5_1 we probably increased the number of coefficients too fast.

The symmetries can also be identify in the pp, pt and tt plots. We will discuss the pt plot but the same carries over to the two other functions. Consider for the trefoil

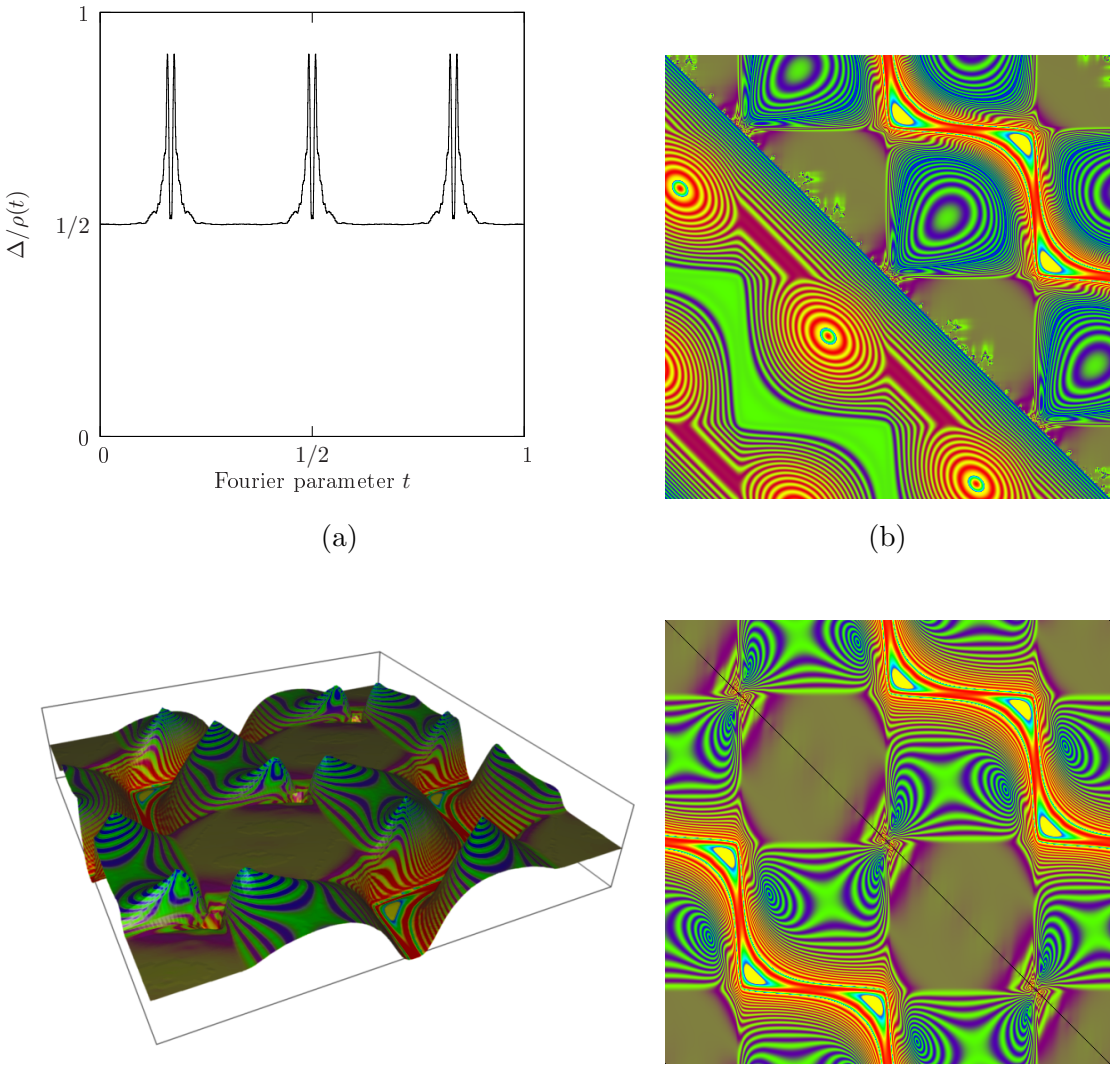
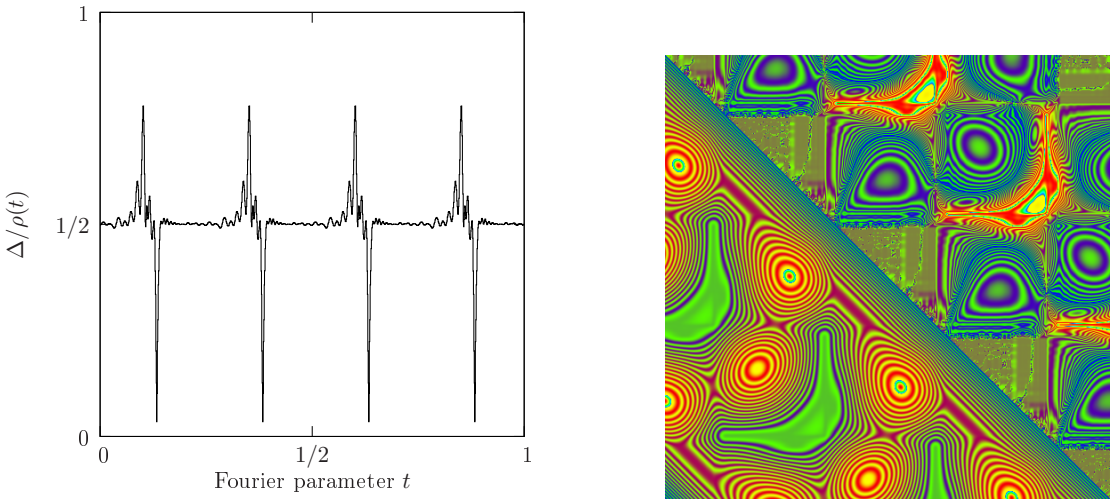


Figure 5.3: Fourier trefoil results. From top left to bottom right there is (a) a curvature plot, (b) the symmetric *pp* and *tt* plots merged at the diagonal into one image, (c) 3D and (d) 2D versions of the *pt* function.



(a) (b)

(c) (d)

Figure 5.4: Fourier figure-eight (4_1) results. From top left to bottom right there is (a) a curvature plot, (b) the symmetric *pp* and *tt* plots merged at the diagonal into one image, (c) 3D and (d) 2D visualizations of the *pt* function.

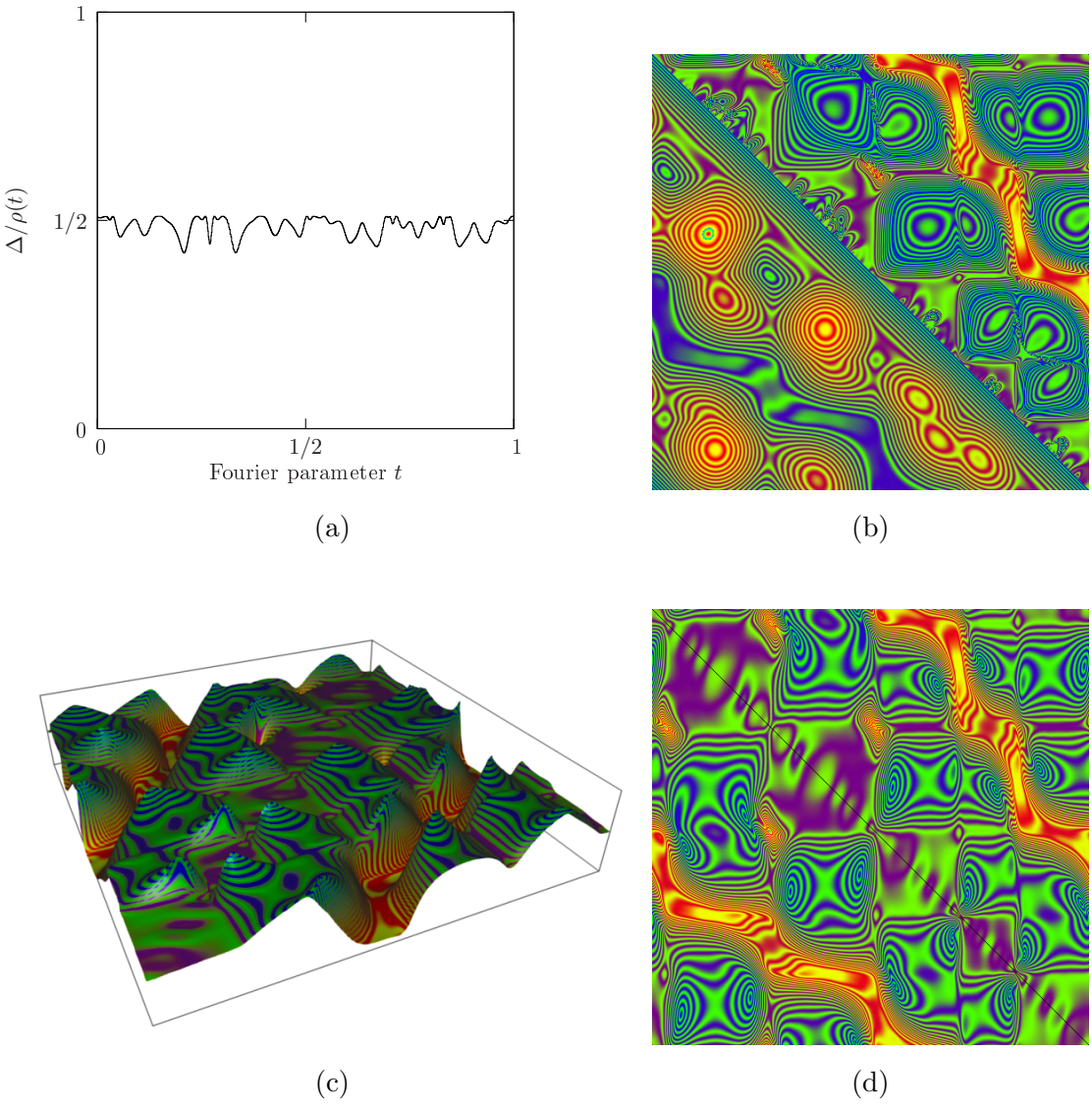
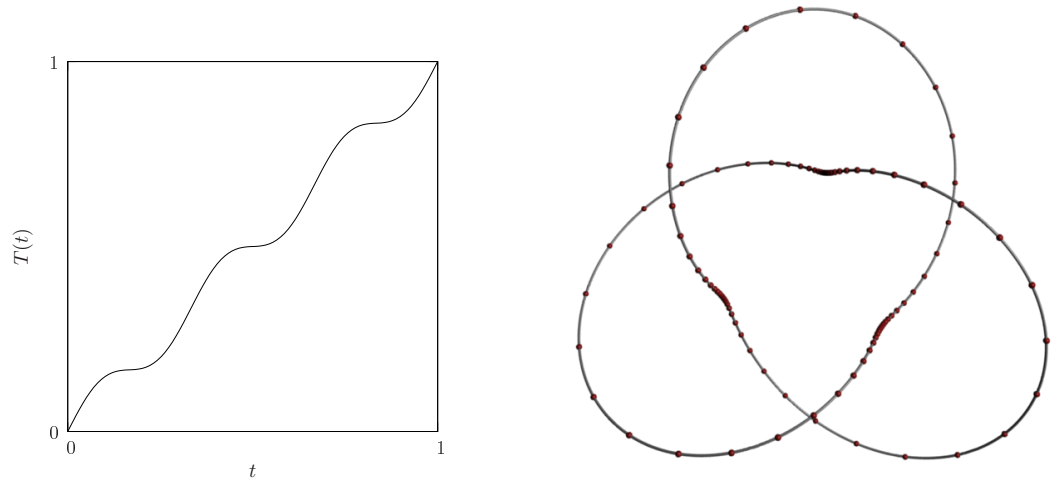


Figure 5.5: Fourier 5_1 knot results. From top left to bottom right there is (a) a curvature plot, (b) the symmetric *pp* and *tt* plots merged at the diagonal into one image, (c) 3D and (d) 2D versions of the *pt* function.



(a) (b)

Figure 5.6: Reparametrization function $T(t)$ (a) for a non-uniform sampling of the Fourier trefoil (b).

$\gamma_F(s), s \in [0,1]$ only $pt(s,t), s \in [0,1/6], t \in [0,1]$, then we can construct the rest of the function. Chapter 3 states that the parameterization of the curve has to be taken into consideration. For the symmetry about the axis from the center through an ear the reparametrization function is $R_0[\gamma_F](s) = \gamma(-s)$. This means that $pt(-s,-t) = pt(s,t)$ which gives us the interval $s = [-1/6, 0]$. Finally the remaining $2/3rd$ is given by the rotation about the central axis, where the reparametrization is $R_{1/3}[\gamma_F](s)$. So shifting the pt patch we currently have $1/3$ in the s and t direction we get the complete function pt. The figure eight pt function is constructed by taking $s \in [0, 1/4]$ and shifting the plot four times by $1/4$. Taking $s \in [0, 1/2]$ for the 5_1 knot and the parameter reflection yields the second part.

A remaining open question concerning pt plots is whether we can reconstruct a curve given its pt plot. To construct a curve curvature and torsion is necessary, curvature is available along the diagonal of pt, but the torsion must be extracted by some other means. For a C^3-curve, the torsion τ is available on the diagonal of tt.

Remark 5.1. *The biarc sampling for the trefoil γ_B is non uniform and the refinement has been done in the curvature peak regions. We observed that this helped as well to improve the Fourier trefoil. During the synthesization of $\gamma_F(t)$ with Equation 5.4, a reparametrization function $T(t)$ given by*

$$T(t) = x + \frac{19\sin(6\pi t)}{20 \cdot 6\pi}$$

is used to slow down in high curvature regions as illustrated in Figure 5.6 (a). Therefore $\gamma_F(T(t)), t \in [0,1]$ yields a non uniform sampling as shown in Figure 5.6 (b) with more points in the three regions of interest.

5.6 Simulated annealing in $\mathbb{S}^3$

The existence of ideal knots has not only been shown for curves in Euclidean three-space [8, 22, 25], but also for $\mathbb{S}^3$ [19]. There are no knots in $\mathbb{R}^4$, since we have one more dimension in which we can "open" the knot. To illustrate this claim we consider a knot in $\mathbb{R}^3$ and add temperature as a fourth dimension. Then we can always open a knot by heating the part of the rope we need to cross and cooling down the other part. This process will always lead to an unknot. Restricting the knot from $\mathbb{R}^4$ to $\mathbb{S}^3$ removes this degree of freedom and, knots do exist in $\mathbb{S}^3$ (even ideal knots [19]).

It is easier to construct a knot γ in $\mathbb{R}^3$, since we can visually check whether or not it has the knot type (is in the isotopy class) we expect. The same procedure is less obvious in $\mathbb{S}^3$ since we can not simply look at the shape. The stereographic projection P is given by

$$P : \mathbb{S}^3 \to \mathbb{R}^3, \qquad x \mapsto \frac{2}{2 - \langle x, (0,0,0,1)\rangle} x,$$

which maps a point x on $\mathbb{S}^3$ to $\mathbb{R}^3$ with respect to the northpole at $(0,0,0,1)$ and the projection plane through $(0,0,0,-1)$ as shown in Figure 5.7. The meaning of the larger sphere S in this illustration will become clear after the next definition ([57]).

Definition 5.1. *An inversion in a sphere S with radius r and center c in $\mathbb{R}^N$ is given by*

$$T : \mathbb{R}^N \to \mathbb{R}^N, \qquad x \mapsto c + \left(\frac{r}{|x-c|}\right)^2 (x - c)$$

where $c \mapsto \infty$ and $\infty \mapsto c$.

We can write a stereographic projection from $\mathbb{S}^3$ to $\mathbb{R}^3$ with northpole c as an inversion in a sphere S with center at c and radius $r = 2$ (see again Figure 5.7). The same mapping T is used to project $\mathbb{R}^3$ to $\mathbb{S}^3$. For polygonal curves this would suffice to map a knot γ - embedded in $\mathbb{R}^3$ - to $\mathbb{S}^3$.

In the case of biarc curves we are also interested in properly mapping the tangents onto the unit 3-sphere. To this end we compute the differential of T at $x \neq c$ to be

$$T'(x) : \mathbb{R}^N \to \mathbb{R}^N, \qquad t \mapsto \frac{r^2}{|x-c|^2} t - 2\frac{r^2}{|x-c|^4}(x-c)\langle x - c, t\rangle.$$

With T and T' it is now straightforward to embed a biarc curve into $\mathbb{S}^3$. Note that the tangents are not of unit norm after the mapping. Unit speed parameterization of the biarc curve is enforced by normalizing all the tangents.

In practice, a curve $\gamma(s) \in \mathbb{R}^3$ is projected to $\mathbb{S}^3$ by embedding it first in the plane $(x, y, z, -1)^T$. Then applying T and T' yields the result. The converse is done by first projecting and then discarding the fourth coordinate of the curve which is now again embedded in the plane $(x, y, z, -1)^T$. Note that $T^2 = id$. A knot's type does not change under inversion.

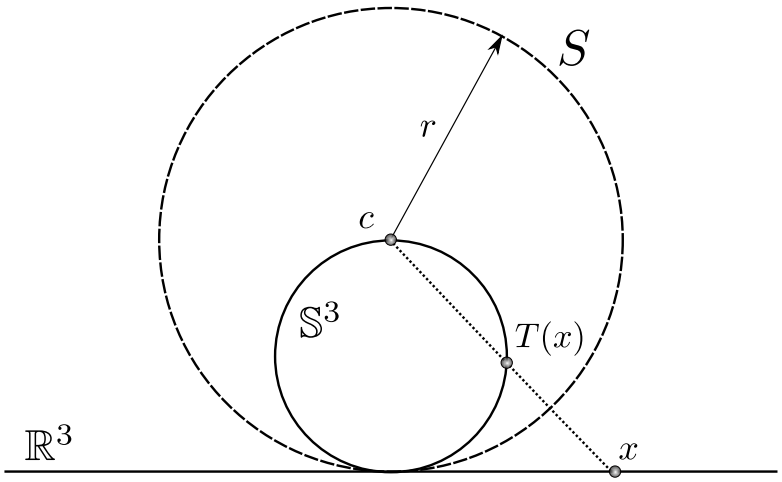


Figure 5.7: Illustration of an inversion in a sphere S with center c and radius r, where a point p in $\mathbb{R}^3$ is mapped to $T(x)$ on $\mathbb{S}^3$.

Especially after manipulating a knot in $\mathbb{S}^3$ it is often desirable to project it back to $\mathbb{R}^3$ for inspection. Depending on how the knot lies in $\mathbb{S}^3$ it might be necessary to change the northpole of the projection (in particular the center c of the inversion). The resulting knot will no longer lie in the plane $(x, y, z, -1)^T$ as explained before. So suppose an inversion center $c^* \neq (0, 0, 0, 1)^T$. We obtain $c^* = (0, 0, 0, 1)^T$ by a change of basis in $\mathbb{S}^3$. This change of basis is done with Givens rotations [21].

Definition 5.2. *A Givens rotation is represented by a $N \times N$ dimensional matrix*

$$G(i,k,\theta) = \begin{pmatrix} 1 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & & \vdots & & \vdots \\ 0 & \cdots & \cos\theta & \cdots & -\sin\theta & \cdots & 0 \\ \vdots & & \vdots & \ddots & \vdots & & \vdots \\ 0 & \cdots & \sin\theta & \cdots & \cos\theta & \cdots & 0 \\ \vdots & & \vdots & & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 & \cdots & 1 \end{pmatrix}$$

where the sin *and* cos *terms appear in the i^{th} and k^{th} row and column.*

The product $G(i, k, \theta)x$ rotates a vector $x \in \mathbb{R}^N$ by an angle θ in the (i, k)-plane. Repositioning the inversion center $c^* = (x_1, x_2, x_3, x_4)^T$ to $(0, 0, 0, 1)^T$ is then obtained by three consecutive Givens rotations in the $(x_i, x_4), i = 1, 2, 3$ planes with angles

$$\theta_i = \frac{\arccos x_4}{\operatorname{sign}(x_i)\sqrt{x_i^2 + x_4^2}}.$$

The inversion in S embeds the knot in the plane $(0, 0, 0, -1)$, where we simply discard the fourth coordinate to obtain a curve in $\mathbb{R}^3$.

Up to now we have dealt with how to construct knot shapes in $\mathbb{S}^3$. We are interested in computing ideal shapes analogous to knots in $\mathbb{R}^3$, where we minimize the ropelength L/Δ. The space $\mathbb{S}^3$ is compact and we no longer have scale invariance for knots, which means that the ropelength is no longer an appropriate energy functional. There are actually four natural and different problems one can consider in $\mathbb{S}^3$ [8, 19] :

1. maximize the thickness Δ,
2. maximize thickness Δ for a fixed length L,
3. minimize length L for fixed thickness Δ,
4. maximize the length L for a fixed thickness Δ.

The last problem in this list is not generic in the sense of ideal knot shapes, although it belongs to the field of optimal packing problems as discussed in [20] for curves in $\mathbb{S}^2$.

In a first step we interpolate the projected knot with biarcs. Two point-tangent data pairs span a sphere as explained in Section 2.2. For point-tangent data in $\mathbb{S}^3 \times \mathbb{S}^3$, where the tangents lie in the $\mathbb{S}^3$ tangent space, this sphere is exactly $\mathbb{S}^3$. The biarc machinery and the thickness computation is still valid for curves in $\mathbb{S}^3$. With this, all the necessary tools are available for simulated annealing on biarc knots in $\mathbb{S}^3$. The first simulations carried out were maximizing the thickness of the set of knots up to seven crossings. For this we used both a biarc and a Fourier representation, where for the Fourier case we had to use a knot in $\mathbb{R}^3$, change its coefficients, project it to $\mathbb{S}^3$ and only then compute the energy $1/\Delta$.

There is the claim [19] that an ideal shape in $\mathbb{S}^3$ where we maximize thickness Δ has to be in both hemispheres. The projection of a small knot in $\mathbb{R}^3$ will result in a curve close to a pole. To stretch the knot to both hemispheres during simulated annealing we introduced a scale move, which is actually more a conformal transformation, where one picks a centre, a direction and the magnitude and applies a scaling motion to the knot in order to push part of the knot to the second hemisphere. On Fourier knots, simply rescaling all the coefficients with the same factor will change the scale of the knot on $\mathbb{S}^3$ and can be seen as a conformal move of the curve on $\mathbb{S}^3$. This transformation helped especially in an early stage of the annealing process.

As already mentioned, annealing knots regularly demands user interaction, which made us focus on only a few, namely the trefoil, the figure eight and the 5_1 knot. There is no point for the moment in presenting the different plots and graphs for all these knots. We will revisit these shapes in the chapter about knot self contact and close for now the discussion about knots in $\mathbb{S}^3$.

5.7 Thickness computation

Until now we implicitly assumed that the thickness of a biarc curve can be computed. On the next few pages we explain the algorithm to do so. The bottleneck of simulated annealing is the compuation of the thickness and is therefore worth optimizing. The thickness algorithm was then parallelized to speed up simulated annealing runs.

Given an arc-length parameterized curve or knot $\gamma(s), s \in [0, L]$, we want to compute $\Delta[\gamma]$. Unfortunately there is no explicit expression for Δ and it has to be approached numerically. The curve γ is sampled at the points $\gamma(s_i), i = 1 \dots N$ with point-tangent data. A first possibility to compute Δ is

$$\Delta[\gamma] \leq \min_{i,j} pt(s_i, s_j),$$

where we approximate the thickness by the smallest radius of the discretized *pt* function. This value is an upper bound, since the thickness might be achieved between data points along the curve. This way of computing the thickness has been used in early stages of an annealing runs, where approximate values for Δ were not an issue. For a sufficiently fine sampling of the curve this approximation converges to the thickness of γ [66]. The author in [66] developed an algorithm based on biarcs. Instead of using a large number of data points for the computation, the arcs of the biarc sampled curve are considered. The shortest distance between the base segments of all pairs of arcs is computed, then every arc pair split into 4 new arcs and mutual distances computed again. Based on two different tests, the double criticality test and the distance test, candidate arc pairs that can not achieve thickness are discarded. This procedure computes the thickness Δ of a biarc sampled curve γ to a relative tolerance ε. Note that this algorithm was developed in $\mathbb{R}^3$ but remains valid in $\mathbb{S}^3$.

Input: n arcs a_i, ε
Output: thickness Δ, length L

$\mathsf{L} \leftarrow$ `Length`(a_i)
$r_{min} \leftarrow$ `MinRad`(a_i)
ArcPairs $\leftarrow$ `DblCritTest`(a_i)
ArcPairs $\leftarrow$ `DistTest`(ArcPairs)
$\varepsilon_{rel}, \Delta_{ub}, \Delta_{lb} \leftarrow$ `ThickBounds`(ArcPairs)
while $\varepsilon < \varepsilon_{rel}$ *and* $\Delta_{lb} < r_{min}$ **do**
  ArcPairs $\leftarrow$ `Bisect`(ArcPairs)
  ArcPairs $\leftarrow$ `DblCritTest`(ArcPairs)
  ArcPairs $\leftarrow$ `DistTest`(ArcPairs)
  $\varepsilon_{rel}, \Delta_{ub}, \Delta_{lb} \leftarrow$ `ThickBounds`(ArcPairs)
end

Algorithm 4: Given a curve γ made of n arcs of circles a_i and a tolerance ε, compute the thickness $\Delta[\gamma]$.

The algorithm as presented in Listing 4 takes as input n arcs a_i given as a Bézier triangle with control points $(a_0, a_1, a_2)_i$, two for each biarc of the curve. The length of the curve is the sum of the arc lengths. In the next step `MinRad` computes the smallest local radius of curvature of the curve r_{min}, the local constraint to the thickness. After that, the tests `DblCritTest` and `DistTest` discard candidate pairs that can not achieve thickness. The double critical checks whether a double critical chord is possible or not within a given pair of arcs. The condition fails for an arc pair $(a_0, a_1, a_2), (b_0, b_1, b_2)$ if any one of the four statements fails

$$\begin{array}{lll} \langle w, t_0\rangle < -\sin\gamma & \text{and} & \langle w, t_1\rangle < -\sin\gamma \\ \langle w, t_0\rangle > \sin\gamma & \text{and} & \langle w, t_1\rangle > \sin\gamma \\ \langle w, \hat{t}_0\rangle < -\sin\gamma & \text{and} & \langle w, \hat{t}_1\rangle < -\sin\gamma \\ \langle w, \hat{t}_0\rangle > \sin\gamma & \text{and} & \langle w, \hat{t}_1\rangle > \sin\gamma \end{array} \tag{5.6}$$

where

$$\begin{aligned} t_0 &= \frac{a_1 - a_0}{|a_1 - a_0|} & t_1 &= \tfrac{a_2 - a_1}{|a_2 - a_1|} \\ \hat{t}_0 &= \frac{b_1 - b_0}{|b_1 - b_0|} & \hat{t}_1 &= \tfrac{b_2 - b_1}{|b_2 - b_1|} \\ \sin\gamma &= \frac{|a_0 - a_2| + |b_0 - b_2|}{|a_0 + a_2 - (b_0 + b_2)|} & w &= \tfrac{a_0 + a_2 - (b_0 + b_2)}{|a_0 + a_2 - (b_0 + b_2)|}. \end{aligned} \tag{5.7}$$

This condition is necessary, but not sufficient. However as proved in [66], it becomes sufficient when the arc length goes to zero, which is the case after enough bisections (splits). Note that in the first double critical test, candidate pairs that are next neighbors are ignored.

The distance test discards pairs if the smallest possible distance between the arcs is still larger than the largest possible distance of the currently best candidate, i.e.

$$mindist(a, b) - \varepsilon_a - \varepsilon_b \geq mindist(a^*, b^*) + \varepsilon_{a^*} + \varepsilon_{b^*}$$

where $mindist(a, b)$ is the smallest Euclidean distance between the segments $a_2 - a_0$ and $b_2 - b_0$. The arc pair (a^*, b^*) is the currently best candidate. The error terms ε_a are given by

$$\varepsilon_a = w_a \sqrt{\frac{1 - w_a}{1 + w_a}} |a_0 - a_1|$$

with $w_a = \cos\delta_a$, where δ_a is the base angle of the Bézier triangle. The error terms for b, a^* and b^* are analogous.

The function `Bisect` splits an arc a into two sub-arcs b and c as shown in Figure 5.8. The error terms for the sub arcs are given by

$$\varepsilon_b = \varepsilon_c = \frac{\varepsilon_a}{2 + \sqrt{2(1 + w_a)}}.$$

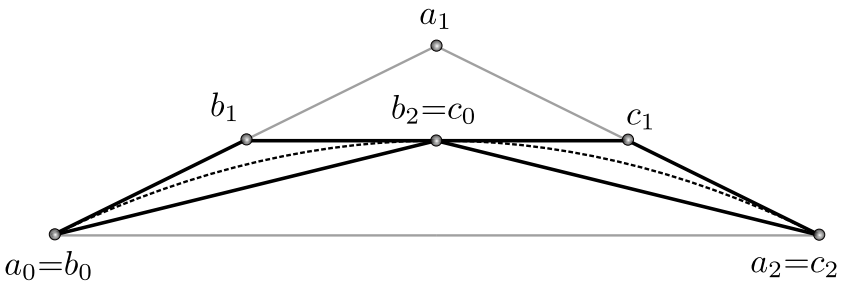


Figure 5.8: Bisection of an arc of circle given by the Bézier control triangle (a_0, a_1, a_2) in two sub-arcs (b_0, b_1, b_2) and (c_0, c_1, c_2).

Finally the thickness bounds function `ThickBounds` computes the following quantities

$$\begin{aligned}
\Delta_{lb} &= \min(r_{min}, \min_i \frac{1}{2}(mindist((a,b)_i - \varepsilon_a - \varepsilon_b)) \\
\Delta_{ub} &= \min(r_{min}, \min_i \frac{1}{2}(mindist((a,b)_i + \varepsilon_a + \varepsilon_b)) \\
\varepsilon_{max} &= \max_{(a,b)_i}(\varepsilon_a + \varepsilon_b) \\
\varepsilon_{rel} &= \frac{\varepsilon_{max}}{\Delta_{lb}}
\end{aligned}$$

where Δ_{lb} and Δ_{ub} are the lower and upper bounds of the current thickness and $(a,b)_i$ is the i^{th} candidate pair.

The algorithm ends when the relative tolerance ε_{rel} is below $\overline{\varepsilon}_{rel}$. Whenever we will state or use a thickness, we mean

$$\Delta = \Delta_{lb},$$

which is the smallest possible radius of the curve with certainty of no self intersection of its tubular neighborhood.

5.8 Parallelizing the thickness algorithm

Now we would like to compute the thickness in parallel using more than a single CPU. This is natural since the bisections are done on a per arc pair basis. Once we have a set of candidate arc pairs the only remaining thing to do is compute the minimal distance on arcs containing a double critical point. The smallest of these values is the thickness of the knot.

There is a master process and several slave processes depending on the available number of CPUs, where every slave process gets a few candidate arc pairs and computes the minimal distance on double critical arcs, if it exists. The master handles in a client-server way the communication with the slaves by sending new arc pairs and the currently best thickness value. If a slave reports to be done with its work it sends back

the distance found. If the computed distance is either larger than the one received from the master or there are no double critical points found, then the distance value sent back to the master is simply a large number. In what follows we will discuss in more detail the work of each process, present the details of the complete algorithm, go into implementation issues and present some benchmark results. The last part points out possible improvements.

The master or root process initializes the knot and computes a first set of candidate arc pairs. Now it has to send one or more pairs to every available slave, which will compute the shortest distance on the pairs. Once every slave has gotten its first workload, the master cycles through them and waits for the worker processes to be done with their task. They report back the computed distance and the master immediately sends the next arc pair (or more than one) to constantly keep the work processes busy. Every time the master receives a distance value, it updates the thickness and sends this value in future jobs. As soon as all the arc pairs are processed, the master notifies the processes that their work is done and ends.

MPI: A Message-Passing Interface Standard

The MPI API specification is a standard for MIMD parallel computing that defines how different processes with separate memory spaces communicate to fulfill a computation or program execution. The communication is done with a message passing protocol between the various processes running in parallel. The communication between processes can be point-to-point (blocking) or asynchronous (non-blocking). A point-to-point communication is considered to be finished when the sender and receiver exchanged the data. For the non-blocking case the sender does not wait until the data has been received by its communication partner.

Different computer and cluster manufacturers have different implementations of the MPI interface tailored to their hardware. A relatively widespread implementation is MPICH, which has also been used as a basis for other MPI implementations. LAM or openMPI are other implementations of this interface standard.

We will not present the entire list of MPI functions. A minimal program is the following code.

```
#include <stdio.h>
#include <mpi.h>

int main (int argc, char *argv[]) {
  int rank, size,

  MPI_Init(&argc, &argv);
```

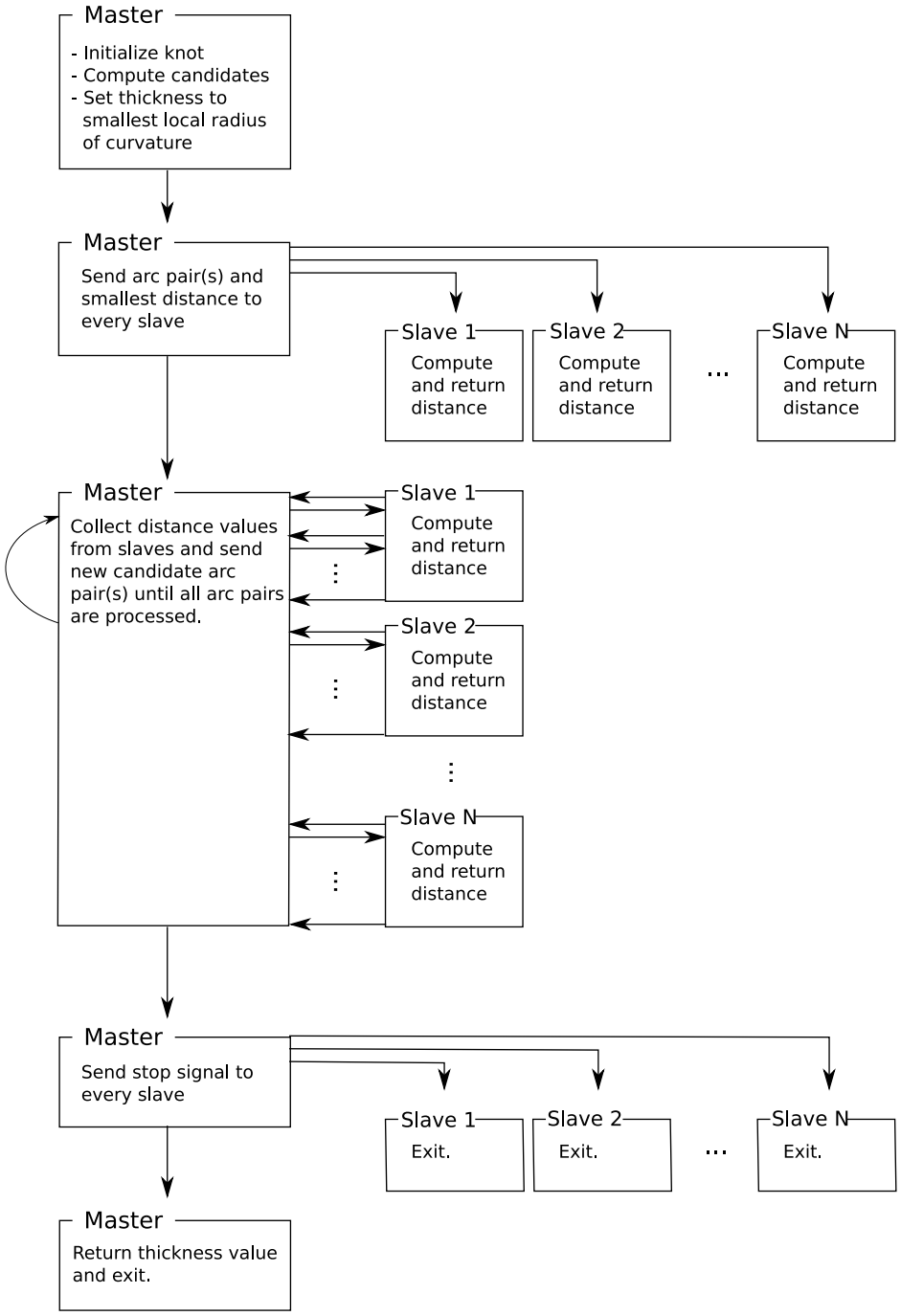


Figure 5.9: Flowchart of the thickness parallelization. The master initializes the candidate arc pairs and distributes them to the slave processes. They compute the shortest distance and report it back to the master. The algorithm stops when all candidates are processed and the master returns the thickness Δ of the curve.

```
  MPI_Comm_rank(MPI_COMM_WORLD, &rank);
  MPI_Comm_size(MPI_COMM_WORLD, &size);

  printf("Process %d of %d\n", rank, size );

  MPI_Finalize();

  return 0;
}
```

In the program listing we use `MPI_Init` to initialize the MPI environment and `MPI_Finalize` to end the execution. The function `MPI_Comm_rank` asks what processor ID this process has and `MPI_Comm_size` requests information about how many CPUs are available in total.

Running MPI programs in a parallel environment such as a computing cluster depends on the different implementations. A sample invocation with MPICH using 4 CPUs would be the following

```
mpirun -machinefile machines -np 4 program
```

where the file `machines` contains a list of host machines and their number of CPUs. Two other routines are worth mentioning since they are the basis of every parallel program for sending and receiving messages.

```
int MPI_Send(void *buf, int count, MPI_Datatype datatype, int dest,
             int tag, MPI_Comm comm)
int MPI_Recv(void *buf, int count, MPI_Datatype datatype, int source,
             int tag, MPI_Comm comm, MPI_Status *status)
```

The various arguments are `buf`, a pointer to the data to be sent or received - `count`, the amount of data of type `datatype`. The integers `dest` and `source` define the rank of the partner process, `tag` is a message identification and `MPI_Comm` is a communication handle. A common handle is `MPI_COMM_WORLD` which includes all the available processes. It is possible to define other communicators depending on the network and computation topology implemented in the program.

Collective communication is possible as well. A broadcast sends a message to all the processes, scatter and reduce sends one task to all the processes in a group or reduces values on all processes to a single value on one process.

Custom datatypes

The MPI standard defines a series of datatypes like MPI_INT or MPI_DOUBLE. In this algorithm we have to send information concerning the candidate arc pairs to the computing children. Every message passed from one process to another must have a specified datatype. We want to send a message containing enough information for the child processes to be able to compute the minimal distance. That is why a new MPI datatype was necessary. It is possible to define an entire structure as a new MPI datatype, which in our case would be the Candi<Vector> class, which describes a candidate arc pair. The class Vector is either a 3 or a 4 vector. However, the size of the candidate structure is 248 bytes, which would cause considerable communication overhead. In order to compute the minimal distance we only need the Bézier points ($9+9$ double) for the arc pair, error information ($2+2$ double) associated with each arc and the currently best minimal distance (1 double) for the subdivision process. This is a total of 23 doubles adding up to 184 bytes, which is slightly better than sending the whole structure. We add one more double used as a signal. Implementing signals with floating point values is usually a bad idea due to rounding errors, but here we set this value larger than 0 if we want the receiving process to compute the distance, and smaller than 0 if we want to inform the process that it can exit. An alternative could be a MPI mixed data type, where we would use a single bit as a signal, and the 23 doubles. The implementation is easier with only doubles and with a single bit alignment problems could occur due to fixed buffer lengths in the sending and receiving process.

The new datatype is therefore an array of 24 doubles, initialized at the beginning of the MPI program with

```
MPI_Datatype MPI_Candi;
MPI_Type_contiguous(24, MPI_DOUBLE, &MPI_Candi);
MPI_Type_commit(&MPI_Candi);
```

This datatype can then be used just like MPI_INT or MPI_DOUBLE. The next thing to do is prepare the data to be sent to the worker processes. For that purpose we have the following two routines

```
void pack_candidate(const Candi<Vector> &c,
                    const double curr_min,
                    double res[24]);
void unpack_candidate(const double res[24],
                      Candi<Vector>* c,
                      double *curr_min);
```

These functions are not part of MPI, which does have functions to pack and unpack data, but we use our own implementation to deal with the specialized candidate structure.

The function `pack_candidate` fills an array of doubles with the information needed from `Candi<Vector>` and `unpack_candidate` initializes the `Candi<Vector>` from `res`. Since the master process has a list of candidates to process, it packs the data in an array of doubles and sends it to the slave process, which then unpacks it into a `Candi<Vector>` object. The minimal distance gets computed, and only this double value is sent back to the master.

Batching candidates

Subdivision on one arc pair can finish very quickly depending on how close we already are to the final thickness value. The consequence is that the child processes will ask for new candidate pairs and the master might become a bottleneck if it is already busy sending out new candidates to the children. To reduce this communication overhead we can play with the number of arc pairs sent at once. This is called batching and needs more memory to store all the arc pairs, but then the workers simply process the whole list and only when finished, they ask for new pairs. The sending and receiving MPI routines are

```
MPI_Send(serial, BATCH, MPI_Candi, i, TAG, MPI_COMM_WORLD);
MPI_Recv(serial, BATCH, MPI_Candi, 0, TAG, MPI_COMM_WORLD, &stat);
```

where `serial` is the send/receive buffer, `BATCH` the number of arc pairs, $i = 1, \ldots, \#CPUs-1$ is the process rank (0 for the master). Whether batching speeds up the computation depends on various parameters like the number of arc pairs, the knot shape, active local curvature, etc.

Discussion

In what follows we use an approximately ideal trefoil to test the program. Everything has been computed on an Intel "Woodcrest" cluster with one master and 14 bi-dual nodes. Figure 5.10 (a) shows the time needed to compute the thickness as a function of the number of CPUs. Up to 10 CPUs, the speed improvement seems reasonable. For more than 10 CPUs the gain flattens out. The master can not serve the workers at full speed, since there are too many of them, which slows down the whole computation.

The next experiment is shown in Figure 5.10 (b), where we vary the number of knot data points, keeping the number of CPUs at 8 for all the runs. This is compared to the standard thickness computation on a single CPU. We can see a more or less linear behavior with a slope visibly smaller for the multi CPU case. Parallelization becomes interesting if we have to compute with extremely refined knot shapes as is the case at the so called end-game, which is when a knot shape is believed to be close to ideal and, as we will see in the next chapter, there are many contact chords.

Earlier we discussed that the master could send more than a single candidate pair to the child processes. Figure 5.10 (c) plots the time versus the number of arc pairs we send in a row. The knot shape has 512 data points and we vary the batch size between 1 and 100. The computation gets faster the more candidates we send at once. A batch size larger than 100 slows the computation down. The master's minimal distance value allows us to discard a lot of candidates early on, but this is no longer the case if we send batches to each child process.

Another delicate point is the actual shape. Different shapes might lead to completely different timings depending on how close various parts of the curve are to the final thickness. If a knot has several regions where points are at distances close to Δ, then it will take much longer to converge. All the subdivisions have to be carried out until the relative tolerance is reached and only then can the master compare the various values and compute the smallest one. The computations in Figure 5.10 (d) show that the speed-up can be significantly different from one knot shape to another. For the knots where the time of a single CPU comes very close to that of the 8 processor run could be explained by this worst case scenario: Suppose that we compute 7 arc pairs in parallel and the master receives 7 distance answers, where the smallest is used for the next 7 arc pairs. If we would follow these 7 candidates in the single CPU case it might happen that the first pair is already the smallest out of the 7 and the next 6 are discarded since they can not yield a lower value. If this is true for almost every block of 7 pairs, then the times for the single and multi processor runs have to be close.

A different analysis, compared to the various parameters discussed above, is the time the master needs to do the initial setup and preliminary computations. Table 5.11 summarizes this for a series of knots with a number of crossings up to 9. The knot j_{3_1} has been computed by [66] and is a very good approximation for an ideal trefoil knot. These times indicate, that the master spends a lot of time in the initialization step. For the k_{3_1} the master needs the same time as the slaves, for the k_{5_1} the worker processes do basically nothing compared to the master. In contrast, for the j_{3_1} knot, the initialization step takes only a fraction of the computation by the processes. The reason is that the j_{3_1} knot is very close to ideal and a lot of candidate pairs have to be considered in the thickness computation. The other knots have fewer nodes (around 200) and are further away from ideal. Note that the batch size has been set to one for these computations.

Possible improvements

The above analysis of the various parts and parameters of the algorithm points out a few weak points and ideas for optimizations. The master might have a lot to do with interprocess communication if the children rapidly finish their computations, however if this is not the case, then the master is idle, waiting for child processes to send back their values, when it could also compute some candidate pairs. If we make the master compute as well, then some of the children might have to wait until the master again answers

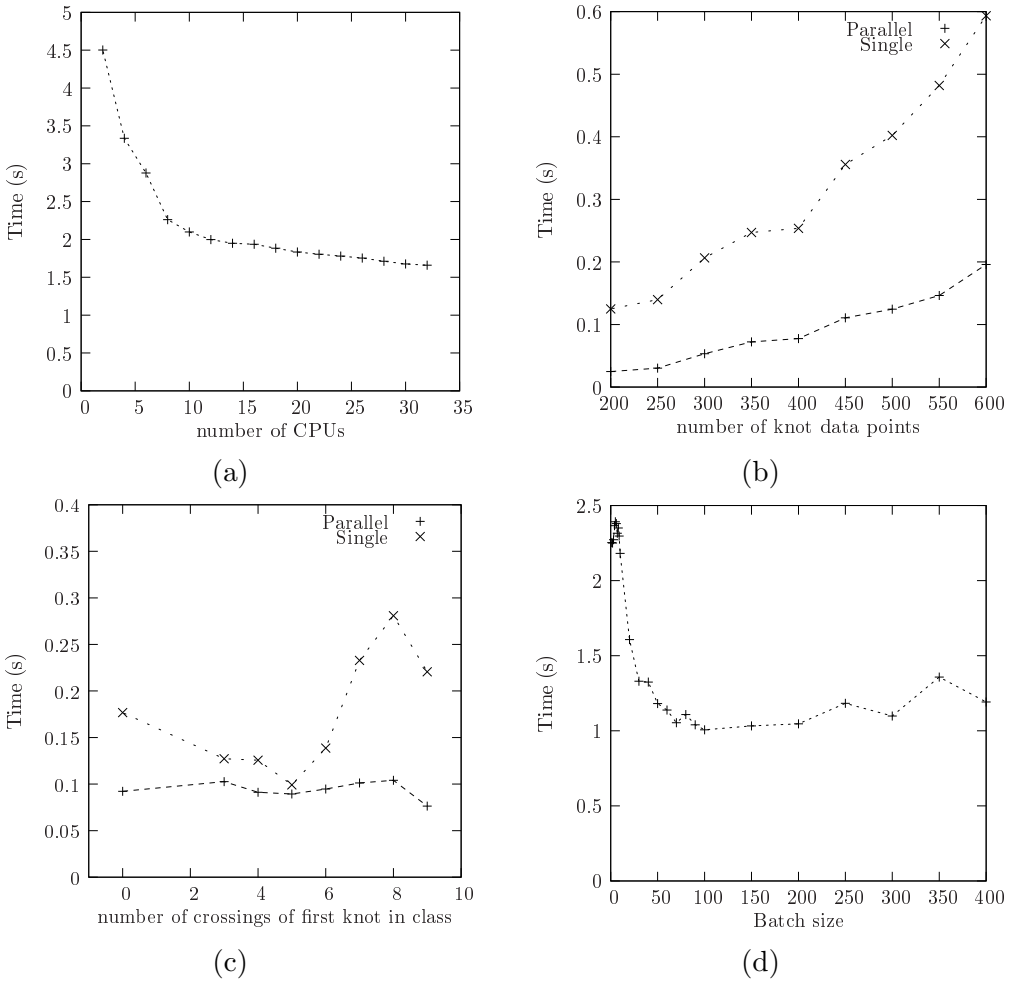


Figure 5.10: Four different benchmarks for the thickness parallelization. The plots are (a) the number of CPUs, (b) the discretization of the curve, (c) different knots and (d) the batching parameter, all versus time in seconds.

knot	data points	master (sec)	workers (sec)	total (sec)
j_{3_1}	512	0.165336	2.092650	2.258000
k_{3_1}	160	0.016368	0.016206	0.032578
k_{4_1}	208	0.024819	0.018046	0.042868
k_{5_1}	232	0.031302	0.000165	0.031471
k_{6_1}	280	0.042829	0.000162	0.042995
k_{7_1}	304	0.052276	0.013455	0.065735
k_{8_1}	352	0.067262	0.012944	0.080209
k_{9_1}	376	0.067325	0.000118	0.067447

Figure 5.11: Timings for the master, slaves and total time spent to compute the thickness of the given knot.

the phone, wasting precious time. This issue could be surmounted by implementing an interrupt system, where the children stop the master in its computation, send the value and receive new pairs.

The master's minimal distance value is important to quickly discard pairs if they can not achieve better. Keeping the children up to date by sending them a new distance value whenever the master receives a better thickness (for example by broadcasting) could take care of that problem. Table 5.11 pointed out that the master spends a lot of time in the initialization step, mainly by setting up the arc pairs and the first double critical filter. Delegating part of this first filtering to the children would reduce the master's work.

Chapter 6

Contact sets and contact surfaces

In this chapter we will discuss self contact of ideal knot shapes. A self contact is where the tubular neighborhood of the knot touches itself. Therefore we are interested in computing double critical chords that are of length twice the thickness, or local contact where curvature is active (see also Chapter 2). Physically this corresponds to increasing the radius of the thick curve up to the thickness, and identifying points between which we have self contact.

Definition 6.1. *Given a closed, non-intersecting curve* $\gamma(s) \in C^1(\mathbb{S}, \mathbb{R}^3 or \mathbb{S}^3)$ *we define the contact set* CS *to be*

$$CS = \{(s,\sigma) \in I \times I | pt(s,\sigma) = \Delta[\gamma]\}.$$

The set of contact points CP *located on the tubular neighborhood of a thick curve with radius equal to its thickness is*

$$CP = \{p \in \mathbb{R}^3 | \text{center of the } pt(s,\sigma) \text{ circle}, (s,\sigma) \in CS\}.$$

For a double critical contact chord $c(s,\sigma)$, this corresponds to the center of the chord $p = (\gamma(s) + \gamma(\sigma))/2$.

Since we only have numerical approximations of ideal knot shapes, it is necessary to introduce a tolerance to specify what we consider to be a contact. We introduce a parameter ε, such that the contact chords satisfy $pt(s,\sigma) \leq \Delta[\gamma](1+\varepsilon)$. We will also call the set of chords $c(s,\sigma)$ for all $(s,\sigma) \in CS$ the contact set. The meaning will be clear from the context.

Remark 6.1. *Consider a circle with radius* 1 *and thus* $\Delta = 1$*. For this case every pair* (s,σ) *on the circle is an element of* CS*, since* $pt(s,\sigma) = \Delta$*,* $s, \sigma \in [0, 2\pi]$*. Nevertheless,* CP *is the simple point in the center of the circle.*

6.1 Contacts and the *pt* function

Evaluating *pt* on the nodes of a point-tangent discretized curve is straightforward. But from the definition of a contact set we see that this point-wise evaluation is not sufficient. Consider a biarc discretized curve γ where we would like to compute the smallest *pt* value between a point p and some point p^* on an arc a. In order to do that we first need to know how to compute the shortest distance between a point and a circle.

Definition 6.2. *Suppose we are in* $\mathbb{R}^3$. *The polar axis* A *of a circle* $C(s), s \in [0, 2\pi]$ *with radius* r *and center* c *is*

$$A = \{c + \lambda b | \lambda \in \mathbb{R}\}$$

where b *is the binormal of the circle.*

Every point of a circle is equidistant to points lying on the polar axis, since it is the straight line orthogonal to the plane in which the circle lies and running through its center. The shortest and longest distance between a point not on the polar axis and the circle are stated in the following lemma.

Lemma 6.1. *Let* p *be a point in* $\mathbb{R}^3$ *and* $C(s), s \in [0, 2\pi]$ *a circle with radius* r *and center* c, *with* $p \neq c$. *Then the distance function* $d(s) = |p - C(s)|$ *is constant if* p *lies on the polar axis of* C *or has a minimum and a maximum for two points* p_- *and* p_+ *on the circle* C. *These points are given by*

$$p_\pm = c \mp r \frac{\overline{p} - c}{|\overline{p} - c|} \tag{6.1}$$

where $\overline{p}$ *is the point* p *projected on the plane in which* C *lies, i.e.* $\overline{p} = p - b \cdot (p - c)b$ *with* b *the binormal of the circle.*

Proof. First we rewrite the parameterization of the circle C in the base (e_1, e_2, e_3), where

$$e_1 = \frac{\overline{p} - c}{|\overline{p} - c|}, \quad e_2 = b, \quad e_3 = e_1 \times e_2.$$

The circle is then

$$C(s) = c + r\cos(s)e_1 + r\sin(s)e_3, \qquad s \in [0, 2\pi].$$

Next, considering that e_i, $i = 1, 2, 3$ is an orthonormal frame and $(c - p) \cdot e_3 = 0$ by construction, we differentiate $|p - C(s)|^2$, which yields

$$\frac{\partial}{\partial s}\langle p - C(s), p - C(s)\rangle = 2r\sin(s)(p - c) \cdot e_1.$$

Since $(p - c) \cdot e_1 \neq 0$, the minimum of the distance function is at $s = 0$ and $s = \pi$. □

The line from p through the center of the circle generated by the bezier arc (a_0, a_1, a_2) intersects the latter in p_-.

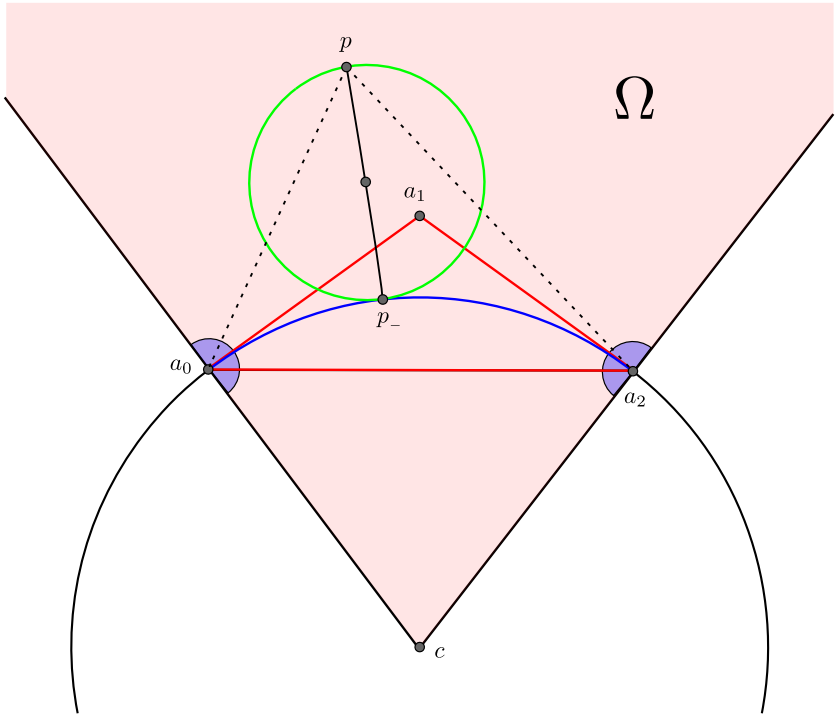


Figure 6.1: Smallest *pt* circle between a point $p \in \Omega$ and an arc of a circle with Bézier control points (a_0, a_1, a_2). In order for *pt* to be have minimum on this arc, the point p must lie in Ω. The point p_- is given by the shortest distance between p and the circle on which the arc lies.

Remark 6.2. *In $\mathbb{S}^3$ the binormal of the circle used in the previous lemma is not simply available by a cross product. Given $p \in \mathbb{S}^3$ and a circle $C(s) \in \mathbb{S}^3$ with center $c \in \mathbb{R}^4$ and radius r, we first construct an arbitrary orthonormal basis $\{e_1, e_2\}$ in $\mathbb{R}^4$ for the plane of $C(s)$. Then the point p is projected onto the plane spanned by $\{e_1, e_2\}$, i.e.*

$$\overline{p} = c + e_1 \cdot (p - c)e_1 + e_2 \cdot (p - c)e_2.$$

The point closest to p on $C(s)$ is now

$$p_- = c + r\frac{\overline{p} - c}{|\overline{p} - c|}.$$

The same construction is of course valid in $\mathbb{R}^3$.

As outlined earlier we are interested in computing the minimum of *pt* between a point p and an arc a. An in depth discussion and a proof for all the cases that can occur when computing *pt* between a point and a circle and in particular an arc of a circle has been carried out in [66]. However, to compute the contact set of a biarc approximated curve γ we do not need to consider all the cases. Figure 6.1 illustrates the only case used to compute the contact set. It shows an arc of a circle a with its Bézier control triangle (a_0, a_1, a_2) and a point $p \notin a$. The only relevant case is when p lies in Ω. Note

that the region Ω is a cake piece which extends to $\pm\infty$ in the direction perpendicular to the plane in which the arc of the circle lies. A point p is in volume Ω when

$$(a_0 - a_1) \cdot (a_0 - p) \geq 0 \quad \text{and} \quad (a_2 - a_1) \cdot (a_2 - p) \geq 0. \tag{6.2}$$

For a point $p \in \Omega$, the minimum of pt is achieved inside the arc a at p^* given by the minimum of the half distance (pp), i.e. $p^* = p_-$. In other words, the radius of the circle going through p and tangent at p^* on the arc a is half the distance between p and p^*. The other cases can be neglected. Here is why :

- If pt is achieved as a maximum of pp then a point $p^\dagger$ close to p^* on the arc a must be closer to p than p^*. Suppose $\frac{1}{2}|p - p^*| = \Delta$ this would mean $\frac{1}{2}|p - p^\dagger| < \Delta$, which is impossible.
- If p is in the region to the left or to the right of Ω (see again Figure 6.1), then pt has its minimum on one of the endpoints a, call it p^* and the corresponding tangent t^*. Since only double critical chords can be part of the contact set we must have $(p - p^*) \cdot t^* = 0$. According to 6.2, for an endpoint of a this is only possible if p lies on the border of Ω.

Input: $2N$ arcs $a_i = (a_0, a_1, a_2)_i$, thickness Δ, tolerance ε
Output: list of contacts CS

$CS \leftarrow \emptyset$
for $b \in a_i$ **do**
  for $c \in a_i$ **do**
    $p \leftarrow b_0$
    if `ptCondition`(p, c) **then**
      $p_- \leftarrow$ `ClosestPoint`(p, c)
      if $|p_- - p|/2 \leq \Delta(1 + \varepsilon)$ **then**
        add chord (p_-, p) as a new contact to CS
      end
    end
  end
end

Algorithm 5: Given a curve γ by $2N$ arcs of circles, the thickness Δ and tolerance ε, extract the contact chords.

Listing 5 explains how to extract the contact set for a curve γ. We approximate γ with N biarcs. The $2N$ sub arcs a_i of the biarcs or actually the Bézier control points thereof are given as input to the algorithm as well as the thickness $\Delta[\gamma]$. For every possible pair of arcs (b, c), $b \neq c \in \{a_i\}$ we check if Condition 6.2 is satisfied with $p = b_0$ and c the arc of circle on which we want to compute the minimum of pt (function `ptCondition`).

If the condition is satisfied, then we compute the minimum of pt as the minimum of $pp = \frac{1}{2}|p - p_-|$, where p_- is given by 6.1 (`ClosestPoint`). Finally the chord $p - p_-$ is added to the contact set if $|p - p_-|/2 \leq \Delta(1 + \varepsilon)$.

6.2 A classic : the $\mathbb{R}^3$ trefoil

We will now discuss the contact set of a biarc interpolation of the Fourier trefoil γ presented in Section 3.3 and introduce at the same time the notion of contact surface. We first give the characteristics of the trefoil, namely the thickness Δ, computed with Algorithm 4, the length L and the ropelength L/Δ

$$\begin{aligned} L &= 1 \\ \Delta &= 0.0305397 \\ L/\Delta &= 32.744237. \end{aligned}$$

The thickness Δ and Algorithm 5 lead to a set of contact chords $c_i(s,t)$ where we recover the arc-length parameters s and t by finding the biarc on which the point lies and bisecting it until we find s^* and t^* such that $\gamma(s^*)$ and $\gamma(t^*)$ are the endpoints of $c_i(s,t)$ for all the contact chords. This yields the contact set shown in Figure 6.2. Note that the contacts correspond to the minimum of pt along the knot where the radius of the circle is Δ. As mentioned in an earlier chapter the pt landscape has a valley with two minima at the bottom. A necessary condition for an ideal shape is that every point is either in contact unless it is a straight line [24]. The plot in Figure 6.2 shows that every point $\gamma(s)$ is indeed in contact with two other points noted $\gamma(\sigma(s))$ and $\gamma(\tau(s))$. We will call $\sigma(s)$ and $\tau(s)$ the contact functions since the contact set of the trefoil is $CS = \{(s, \sigma(s))|s \in I \cup (s, \tau(s))|s \in I\}$. Note that the function τ is the inverse of σ, and it is therefore sufficient to consider only one of the functions for the trefoil's contact set. Further the contact functions are periodic, smooth and monotone. These conclusions are for the moment motivated by numerics, but will be proven in a later section. Contact chords $c(s, \sigma(s))$ will be called outgoing contacts and $c(s, \tau(s))$ incoming contacts. We now introduce the concept of a contact surface.

Definition 6.3. *A ruled surface Σ is a surface generated by translating and rotating a straight, possibly variable length, line segment through space.*

A parameterization for Σ is for example given by

$$\Sigma(s,h) = p(s) + h\mu(s)v(s), \quad s,h \in [0,1]$$

where $p(s)$ is a curve in space, $v(s)$ a vector on the unit sphere, and $\mu(s) \in \mathbb{R}$ a scaling factor for v. We can now define the two border curves $p(s) = \Sigma(s,0)$ and $q(s) = \Sigma(s,1)$, and give another parameterization for Σ :

$$\Sigma(s,h) = (1-h)p(s) + hq(s).$$

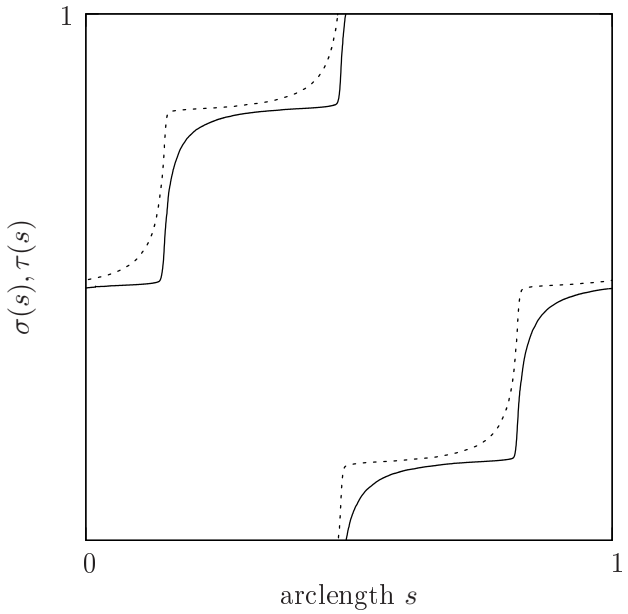


Figure 6.2: Fourier trefoil contact set separated in the two contact functions $\sigma(s)$ (solid line) and $\tau(s)$ (dashed line).

Consider the outgoing contact chords of the trefoil $\gamma(s), s \in I$ parameterized by $\sigma(s)$. The two points $\gamma(s)$ and $\gamma(\sigma(s))$ are the endpoints of the outgoing contact chord at s. Running along s smoothly varies the endpoints of the corresponding contacts.

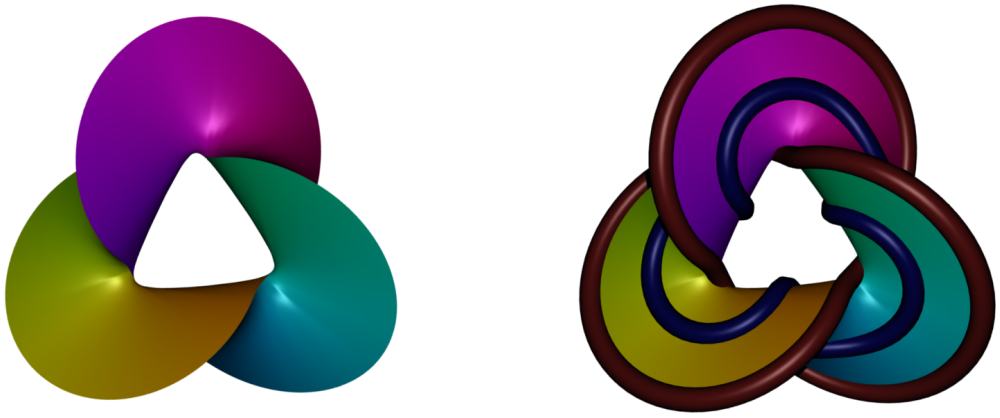
Definition 6.4. *The contact surface of an ideal knot $\gamma(s), s \in \mathbb{S}$ in space and contact set $\sigma(s), s \in \mathbb{S}$ is the ruled surface Σ defined as*

$$\Sigma(s,h) = (1-h)\gamma(s) + h\gamma(\sigma(s)), s \in \mathbb{S}, h \in [0,1].$$

The contact surface Σ of an ideal knot γ is a constant width band, where the length of the linear segments used to construct Σ is twice the thickness $2\Delta[\gamma]$. The contact curve of the knot is then given by

$$\gamma_{1/2}(s) = \Sigma(s, 1/2).$$

Note that $CP = \gamma_{1/2}(I)$. This is where the tubular neighborhood of the knot touches itself. Note that we constructed the contact functions $\sigma(s)$ and $\tau(s)$ by linear interpolation since the initial discrete contact set lacks resolution in some regions due to the parameter used to extract it. Figure 6.3 (a) shows the contact surface constructed by triangulating sufficiently many chords, parameterized by $c(s, \sigma(s))$ walking along the knot by varying s between 0 and 1. In Figure 6.3 (b) we also add the trefoil knot and the contact curve. Studying the contact surface of the trefoil brought our attention to billiards on knots.



(a) (b)

Figure 6.3: Contact surface (a) for the Fourier trefoil and (b) visualization of the same surface including the original trefoil curve with a radius of the tube much smaller than Δ and the thickened contact curve.

Definition 6.5. *Let γ be a curve in $C^{1,1}(\mathbb{S}, \mathbb{R}^n)$ and $\sigma(s), s \in \mathbb{S}$ its contact set. A k-billiard on γ at $s^* \in \mathbb{S}$ arises when*

$$\underbrace{\sigma \circ \ldots \circ \sigma}_{k \text{ times}}(s^*) = s^*.$$

This definition of a billiard on a curve means that if we start at a point $\gamma(s)$ on the curve and follow the contact chords according to σ, then after k steps we reach $\gamma(s)$ again.

Example 6.1. *The only known analytic shape of an ideal knot is the unknot. Consider a circle of radius* 1*. Every point s on the circle is in contact with every other point. The double critical contacts are given by*

$$\sigma(s) = s + \pi, \quad s \in [0, 2\pi].$$

The double critical contact chords give a 2-billiard at every point s, since $\sigma^2(s) = s$ and the billiards starting at s and $s + \pi$ are identical.

Figure 6.4 is a plot of $\sigma(s), \sigma^4(s)$ and $\sigma^9(s)$ for the Fourier trefoil. Powers of σ have been computed by linear interpolation. This gives a hint as to where we might find a billiard. A candidate for a 9-billiard is around $s = 0$. So we recursively compute the contact chords along the knot starting at $\gamma(0)$. The adopted parameterization of the trefoil identifies $\gamma(0)$ to be the point furthest away from the center of the knot. So it is on the π rotation symmetry axis in the middle of one of the "ears". Figure 6.5 shows how the 9-billiard is located in the trefoil. Notice in Figure 6.5 (b), that the closed trajectory is not planar. The star shaped path has the correct symmetries with respect to the trefoil and seems to be the only billiard existing. But this is again only suggested by numerics and visualization.

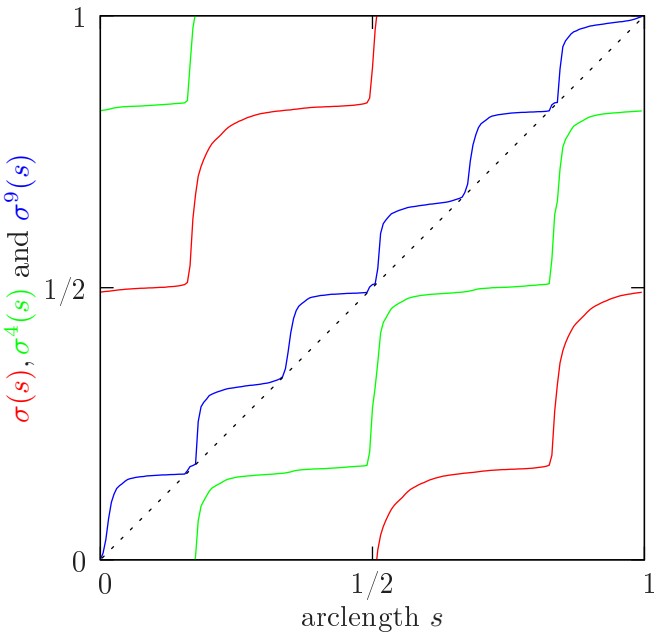
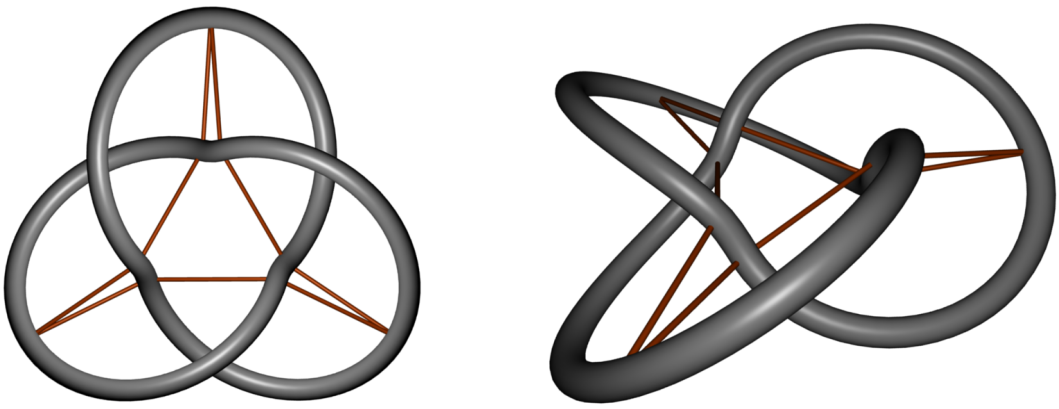


Figure 6.4: Plot of different powers of the contact function $\sigma(s)$ for the Fourier trefoil, namely σ, σ^4 and σ^9.



(a) (b)

Figure 6.5: Visualizations of the 9-billiard contact chords on the trefoil.

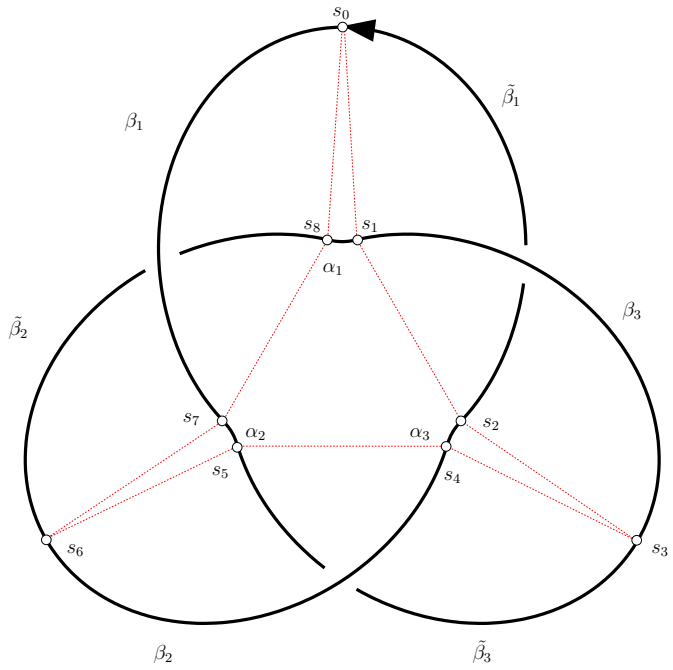


Figure 6.6: The parameters $s_i = \sigma^i(0)$ of the 9-billiard partition the trefoil in 9 curves: α_i, β_i and $\tilde{\beta}_i$, $i = 1, 2, 3$. These curves are related with respect to the trefoil's symmetries. The contact function σ maps the parameter interval of each curve bijectively to the parameter interval of another of the curves.

Remark 6.3. *The powers σ^{11} and σ^{20} show points that come very close to the diagonal, but in order to be an 11 or 20-billiard we would need 11 or 20 such points along the diagonal. The number of these points is however much lower, and these powers can therefore not yield a billiard.*

The contact star leads to another observation : it is possible to construct the whole trefoil with only two curve parts. Recall that due to symmetries only $1/6^{\text{th}}$ of the trefoil is needed. We define the fundamental domain that generates the symmetric trefoil by

$$\begin{aligned} \alpha(s) &= \gamma(s), \quad s \in \left[\sigma(0), \sigma^{-1}(0)\right], \\ \beta(t) &= \gamma(t), \quad t \in \left[0, \sigma^{-2}(0)\right]. \end{aligned}$$

The complete trefoil is then obtained by symmetry operations on the segments α and β. There is a total of 9 curve segments $\alpha_i, \beta_i, \tilde{\beta}_i$, $i = 1, 2, 3$ and this partition comes from the 9-billiard. We label the nine arclength parameters of the billiard by s_i, $i = 0, \ldots, 8$, and the nine components of the trefoil are illustrated in Figure 6.6.

For the following discussion we drop the index i and consider the fundamental domain α and β. The π rotation symmetry axis through the middle of α means that α is self

symmetric with respect to that axis. This same symmetry maps β to $\tilde{\beta}$. Rotating α, β and $\tilde{\beta}$ by 120 degrees about the center axis yields the finished trefoil.

We now establish the contact connectivity between the different curve segments. The 3D visualization in Figure 6.7 reveals how the congruent curve segments β (blue) and $\tilde{\beta}$ (red) are in contact with α (green). This holds for all three "ears", but only one case is shown. The slightly transparent blue surface simply emphasizes that the red surface is "behind" the blue. The contact surface patches are delimited by the contact billiard. The supplementary two contact chords visualized in the same picture seem to form an angle extremely close to π, are orthogonal to the π rotation symmetry axis and are located in the middle of an α segment. This however is not yet the whole contact surface. There is also contact between β and $\tilde{\beta}$, which is easy to identify using the contact surface in Figure 6.3.

Using the notation $\cdot \rightsquigarrow \cdot$ (in contact with) and the curve segment labels given in Figure 6.6 we now have the connectivity

$$\begin{aligned} \beta_i &\rightsquigarrow \alpha_i, \\ \tilde{\beta}_i &\rightsquigarrow \alpha_i, \qquad i = 1, 2, 3 \\ \beta_i &\rightsquigarrow \tilde{\beta}_{i+1}, \end{aligned}$$

where we wrap i from 3 back to 1.

A last remark about the symmetrized trefoil is that it lies in a circumsphere with radius $r_{max} = |\gamma(0)|$ which touches the curve $\gamma(s)$ at the points $\gamma(k\pi/3), k = 0, 1, 2$. There also exists an insphere with radius $r_{min} = |\gamma(1/2)|$, touching the curve at the points $\gamma(t_i)$, $t_i = (2i-1)/6$, $i = 1, 2, 3$, which is the point in the middle of the segments α_i. All these points either touching the circumsphere or the insphere lie on a symmetry axis of the trefoil. For the discussed Fourier trefoil of arclength 1, the radii are

$$r_{min} = 0.0362947, \quad r_{max} = 0.0973923.$$

Note that the thickness for this knot is $\Delta = 0.0305397$ and there is therefore unoccupied space in the center of the knot. This must be the case. For suppose $\Delta = r_{min}$, there would exist a ball of radius Δ centered at the origin touching the curve in three points. A theorem of Gerlach [19] quoted in Theorem C.1 implies that the curve would have to be a circle, a contradiction.

6.3 Surgery on a space invader

The next knot we inspect is the 4_1 or figure-eight knot. Figure 6.8 shows the contact set for a slightly more annealed version of the 4_1 knot presented in [11] with thickness $\Delta = 0.02374408$ and $L = 1$. As mentioned earlier, even though the thickness of the Fourier 4_1 in Section 5.5 is slightly better, the curvature profile is less converged. These

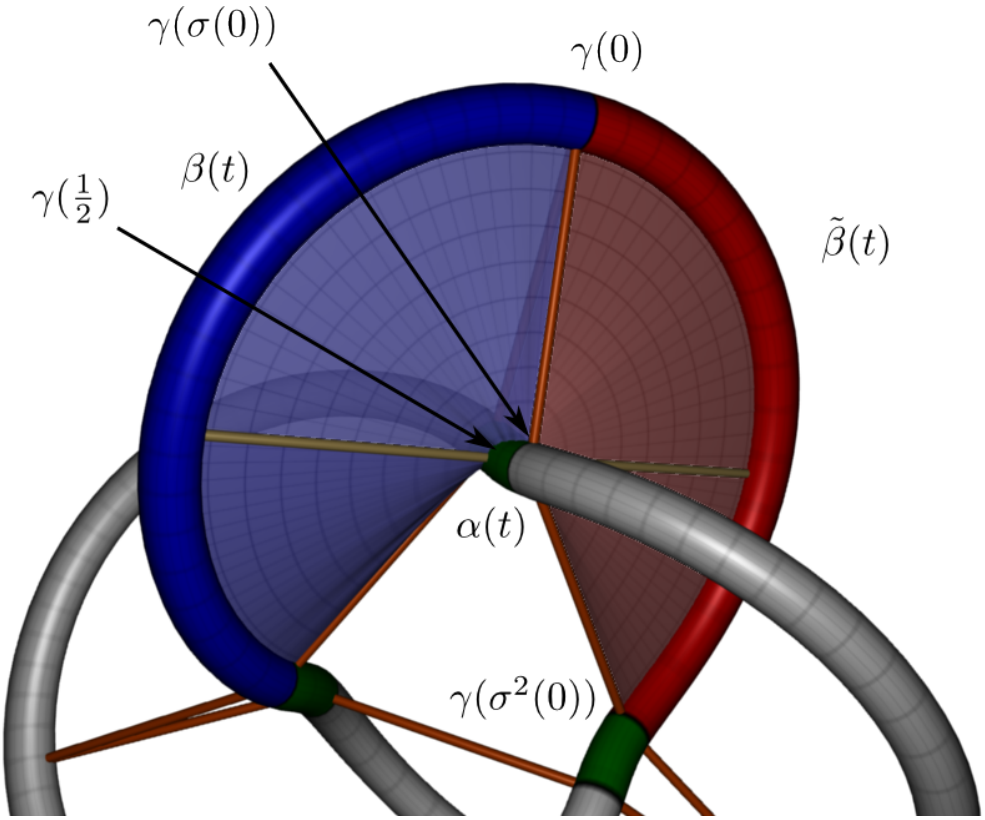


Figure 6.7: Symmetric trefoil contact properties between a segment α and the two congruent components β and $\tilde{\beta}$ (blue and red). The straight successive pair of contact chords, and the 9-billiard are also shown.

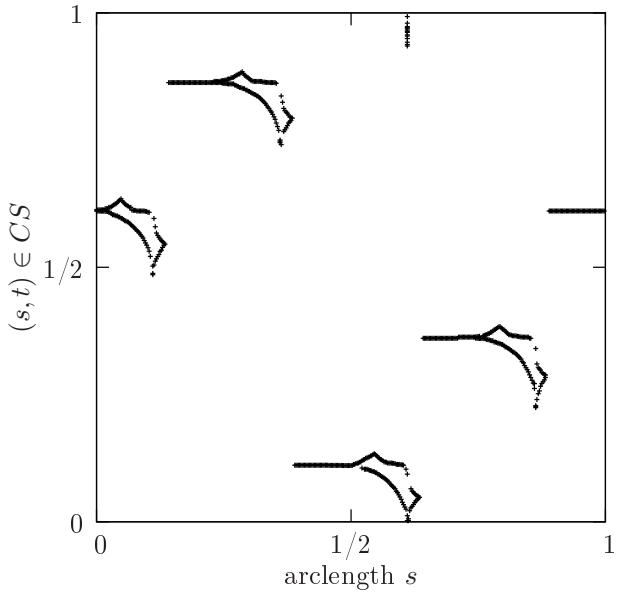
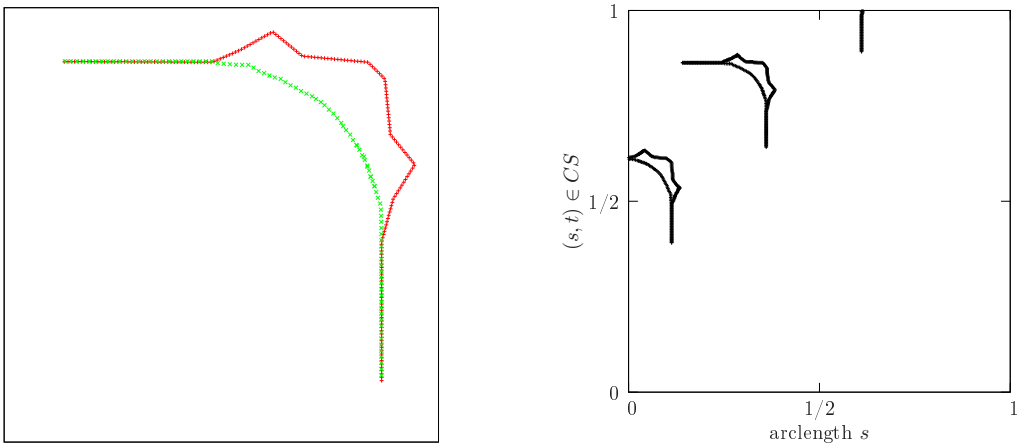


Figure 6.8: Contact set for the figure eight knot.



(a) (b)

Figure 6.9: (a) decomposed space invader of the figure eight contact set and (b) the total resampled contact set for the 4_1 knot. Symmetry of the minima of the *pt* function implies that only two copies of a space invader need to be considered.

small variations along the curve lead to a less complete contact set. This is the reason why we chose the biarc annealed version of the 4_1 to extract the contact set.

The contact set immediately reveals that it can not be expressed as a function $\sigma(s)$ which is smooth, connected and monotonic as was the case for the trefoil. Rather there are two disconnected components where the shape of each component looks like an enemy in the classic arcade game "Space Invaders". A single space invader can further be decomposed into a body and legs. The body is a ruled surface similar to the contact surface of the trefoil, and each leg is a double sided sheet. The way the contact set is numerically sampled right now can not lead to smooth surface. We need again a uniform sampling as for the trefoil. The discrete contact set is not monotonic, so we rotate the plane in which the space invader lies by $\pi/4$ and split it into an upper and a lower curve, where the legs are included in both curves. Uniform sampling of the two curves and rotating back by $-\pi/4$ leads to the picture in Figure 6.9 (a). Note that we thicken the double sided sheets (corresponding to the legs) slightly so that the result is a single, closed, non-smooth surface since there are two extreme 180 degree turns at the tips of the legs. We are interested in a visual impression of the contact surface and this surgery will eventually lead to that.

Due to the 2-symmetry of the knot we can construct the second component from the first by a shift of $(+1/4, +1/4)$ in the (s,t)-plane as remarked in Section 3.3. The contact set shows 4 space invaders, but only two are relevant since the minimum of a *pt* function for an ideal knot from which we compute the contacts is symmetric. This, and the fact that one of the figure-eight symmetries involves a parameter shift $S_{1/2}$, is actually why a single space invader is symmetric with respect to a diagonal axis cutting the head through the middle. The final result of the resampled contact set is shown in

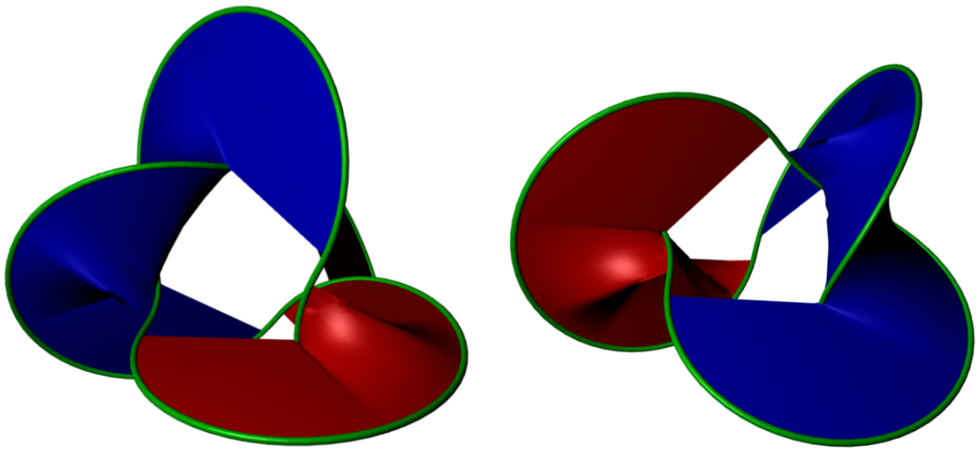


Figure 6.10: The surfaces in red and blue are the two components of the figure eight contact set.

Figure 6.9 (b). It is now straightforward to construct a triangulated contact surface for each component. Figure 6.10 shows the two components in different colors as well as the figure-eight shape with a tube radius much smaller than its thickness.

The details of the 3D shape are difficult to see on a static image. It is still possible to follow the red or blue surface from one double sided sheet to the other (the legs of the space invader). The two components of a single space invader colored in Figure 6.9 can be identified in the 3D version. The green component is the smoothly rotating side of the surface and the red component has a tent-shaped feature just before the double sided sheet at both side. If we consider both components, then we see that the end of a double sided sheet comes close just opposite to a tent of the other component, but without touching it, which could indicate that the connecting segment along the curve is a straight line. Straight bits on curves do not have to have contact with the rest of the curved knot. This can also be observed in the contact set (Figure 6.9 (b)). The difference in arc-length s between the lower end of a leg of one space invader and the upper tip of a small triangle in the body of the other space invader is $ds = 0.0077$ (about 8‰ of the length of the knot).

6.4 Torus knot contact surfaces

In the last two sections we have discussed the contact properties of the trefoil and the figure-eight knot. The trefoil knot belongs to a particular class of knots called torus knots.

Definition 6.6. *Let $T : [0,1] \times [0,1] \to \mathbb{R}^3$ be a torus with radii $a, b \in \mathbb{R}$. A (p,q)-torus knot also noted $T_{p,q}$ is a knot that is ambient isotopic to a knot that wraps p times around T in one direction and q times in the other.*

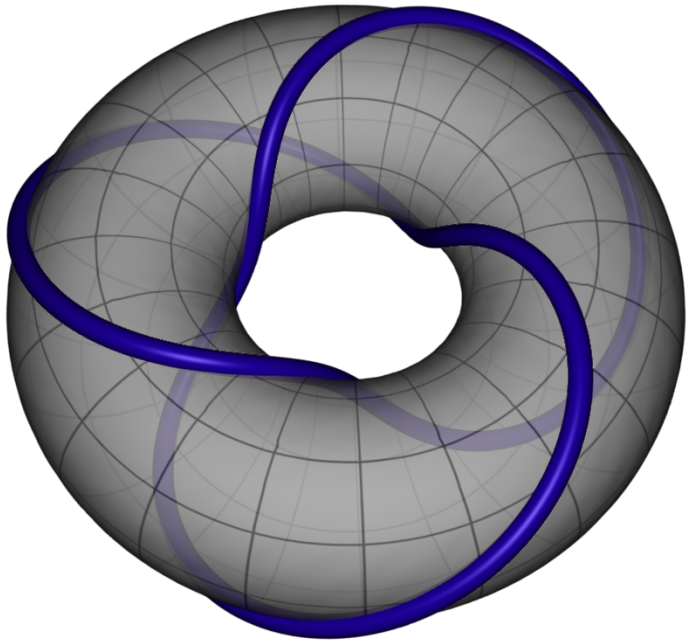


Figure 6.11: The blue curve is a trefoil or $T_{2,3}$ torus knot lying on a gray shaded torus in $\mathbb{R}^3$.

The trefoil is a $(2,3)$-torus knot as illustrated in Figure 6.11. A parameterization for a knot γ wrapping p and q times around a torus $T \subset \mathbb{R}^3$ of major radius R and minor radius r is

$$\gamma(s) = \begin{pmatrix} \cos(p\,s)(R + r\cos(q\,s)) \\ \sin(p\,s)(R + r\cos(q\,s)) \\ r\sin(p\,s) \end{pmatrix}, \qquad s \in [0, 2\pi].$$

We used the previous equation to generate initial conditions for the ideal knots computations. All of the ideal torus knots we studied show a double valley in the *pt* plots just like the trefoil. The minima in the valleys can again be parameterized by the periodic contact functions $\sigma(s)$ and $\tau(s)$. However, for torus knots other than the trefoil, additional isolated contacts are also present in the contact set. To construct a smooth contact surface they have to be discarded in the numerical process. Figure 6.12 shows the complete contact set and the contact surface for the torus knots $T_{2,5}, T_{2,7}, T_{2,9}$ and $T_{2,11}$. Notice that the last knot is still quite far from ideal. The isolated contact chords in the pt plots seem to form crosses. At first, this indicated, that from a given arclength parameter s, we have two or more isolated contacts. Then we numerically verified, that the contact chords do not start at exactly the same point, when they seem to lie on horizontal or vertical lines.

The $T_{2,5}$ knot computations are not as well converged as the trefoil and we could not identify a billiard in this knot. We suspect that a knot's symmetry might imply a billiard, which would mean that the $T_{2,5}$ knot could have one, but did not attempt

anything in that direction. Neither did we investigate billiards on other torus knots.

The contact surfaces in Figure 6.12 were actually 3D printed as physical objects. We will present images of the shapes in Chapter 7.

6.5 Contact sets in $\mathbb{S}^3$

Torus knots can also be constructed in $\mathbb{S}^3$. A torus is a mapping of the unit square $\mathbb{S}\times\mathbb{S}$. If we consider $\mathbb{S}\subset\mathbb{C}$, then the torus is embedded in a four dimensional space.

Definition 6.7. *A Clifford torus is a torus embedded in* $\mathbb{S}^3$ *[5]*

$$\mathbb{S}\times\mathbb{S}\cong\{(R\,e^{i\theta},r\,e^{i\phi})\in\mathbb{S}^3|\theta,\phi\in[0,2\pi),\,r^2+R^2=1,\,r,R\in\mathbb{R}\}.$$

The major and minor radii of the torus (R and r) are treated the same way as in $\mathbb{R}^3$ and a (p,q)-torus knot on a Clifford torus in $\mathbb{S}^3$ is given by the curve

$$\gamma(s)=(R\,e^{p\,i\theta},r\,e^{q\,i\phi}),\qquad\theta,\phi\in[0,2\pi).$$

H. Gerlach proposed [19] to consider the family of $(2,3)$-torus knots on a Clifford torus with radii $R=\sin\eta$, $r=\cos\eta$. He then computed the optimal value for η where this knot has maximal thickness. This value is $\eta=\arctan(\sqrt{3/2})$. Figure 6.13 (a) shows the unit square for the torus on which this curve lies. Further we draw a dashed line for contact chords in $\mathbb{S}^3$ which shows that every point along the curve is in contact with exactly two other points. Further the dashed line crosses the curve 5 times. So at every point on the curve there exists a 5-billiard. The fact that every point is in contact with exactly two other points can be compared to critical pitch helices in $\mathbb{R}^3$ [38, 66]. Closed curves lying on a Clifford torus are actually helices in $\mathbb{S}^3$ and the critical pitch analogous to the optimal value for η. There is however the difference that in $\mathbb{R}^3$ that the two non local contacts at a point of the curve extend to infinity in both sides of the helix.

An observation about $(2,2n+1)$-torus knots for $n>0$ is that for $\eta=\arctan\left(\sqrt{\frac{2n+1}{2}}\right)$ we have an ideal helix with a $2n+3$-billiard at every point along the curve. This has only been verified numerically. The contact surface for $n=1,2,3,4$ with the appropriate η is a torus. For a different η, there are only a few contacts and it can therefore not be a critical pitch helix. The relation between η and a $(2,2n+1)$-torus knot has not been proved, it is only numerically motivated.

The trefoil on a Clifford torus as proposed by Gerlach and baptized the $\mathfrak{g}$-Trefoil, is actually a very good candidate for ideality (maximizing thickness) in $\mathbb{S}^3$. Running simulated annealing on this shape did not lead to anything better. On the other hand simulations starting from an arbitrary trefoil ended up close to a $\mathfrak{g}$-Trefoil.

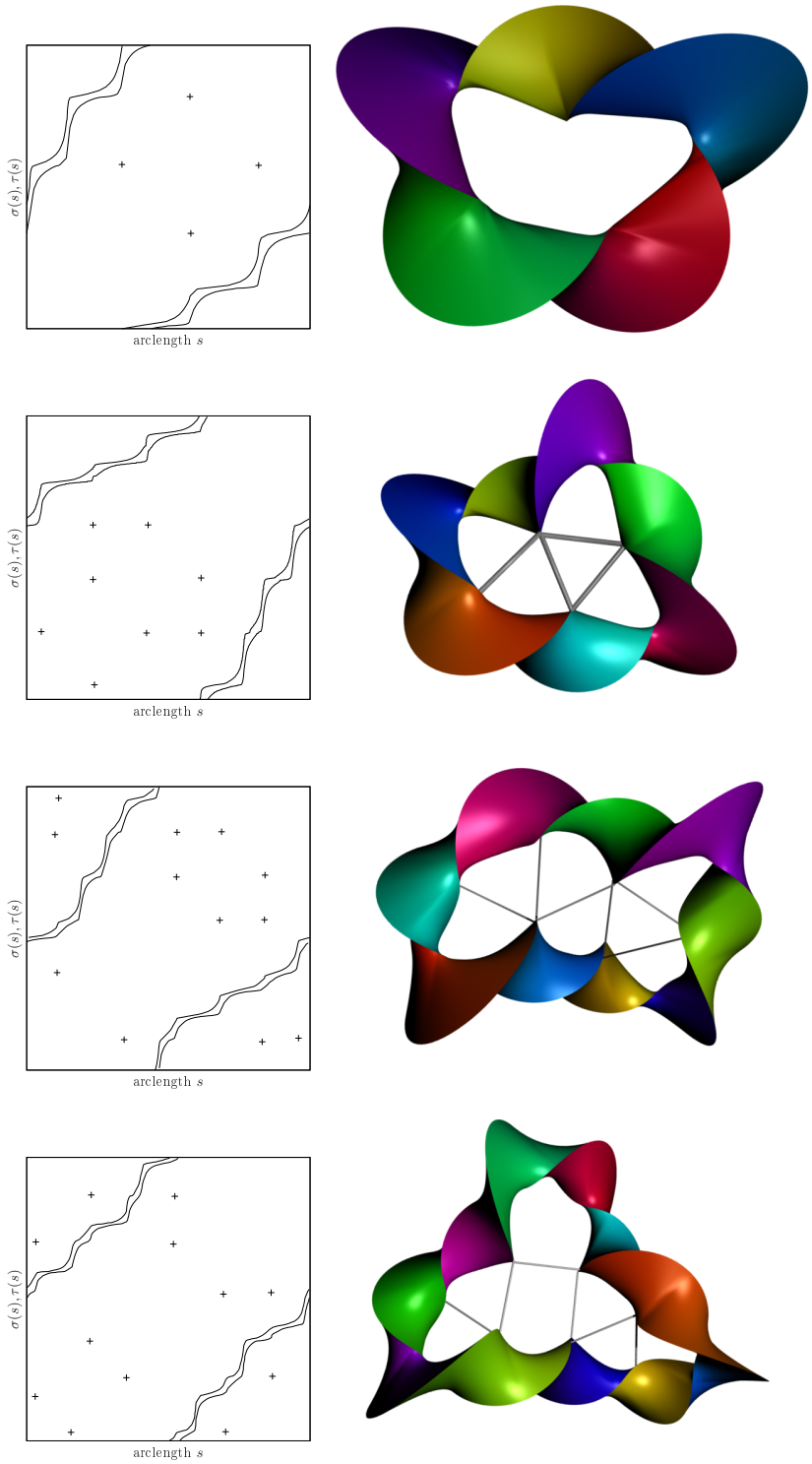
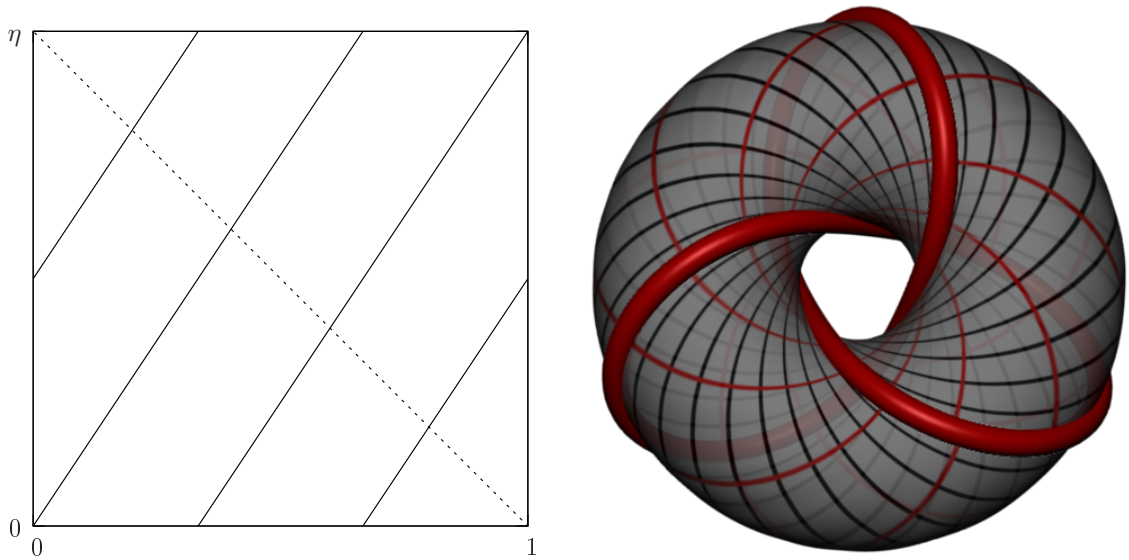


Figure 6.12: From top to bottom : Contact functions σ, τ and isolated contacts are plotted to the left, and on the right the corresponding contact surface for the torus knots $T_{2,5}, T_{2,7}, T_{2,9}$ and $T_{2,11}$.



(a) (b)

Figure 6.13: The unit square in graph (a) depicts the optimal $(2, 3)$-torusknot in $\mathbb{S}^3$. The dashed line is a 5-billiard and a Villarceau circle cutting the trefoil in 5 points. Picture (b) shows the trefoil on a Clifford torus with a few Villarceau circles in red projected to $\mathbb{R}^3$.

The contact chord in $\mathbb{S}^3$ is not a straight line. It lies on the unit sphere and is a geodesic on the sphere, i.e. an arc of a great circle. The previously announced 5-billiard is in fact a Villarceau circle.

Definition 6.8. *[5] The intersection of a bitangent plane with a torus is called Villarceau circles.*

The union of all the contact billiards along the trefoil gives the Clifford torus on which the trefoil lies. Figure 6.13 (b) shows the stereographic projection of the $\mathfrak{g}$-Trefoiland the contact chords to $\mathbb{R}^3$. The red lines are Villarceau circles and the black lines are the Villarceau circles orthogonal to them, which are not contacts. The union of the "curved" contacts to a contact surface indeed produces a torus (Figure 6.13 (b)). We use the inversion in a sphere technique from Section 5.6 (page 55) for the 3D visualization.

6.6 Homotopy

The contact set of a trefoil as discussed in the first part of Section 6.2 has been observed by [66, 11, 54] and confirmed by [2]. The following is joint work with H. Gerlach. In this section we would like to define a set of hypotheses from which it can be proven that the contact set for the ideal trefoil is itself a trefoil. Numerics suggest that the hypotheses are satisfied on the ideal trefoil as well as on subsets of the contact set of other $T_{2,2n+1}$

ideal torus knots in $\mathbb{R}^3$, so that their contact sets would also contain $T_{2,2n+1}$ knots.

We will now rephrase the properties of ideal knots explained in Chapter 2 and proved in [24]. Assuming C^2-smoothness it was proved [24] that an ideal knot γ at any parameter s is either locally

(1) straight, i.e. $\gamma''(s) = 0$,

or thickness is attained by at least one of the following

(2a) locally, i.e. $|\gamma''(s)| = 1/\Delta[\gamma]$,

(2b) globally, i.e. there exists $t \neq s$, such that the contact chord $c(s,t) := \gamma(t) - \gamma(s)$ has length 2Δ and is orthogonal to γ at $\gamma(s)$ and $\gamma(t)$.

The discussion about the contact set of the ideal trefoil γ in Section 6.2 suggest that :

(H1) in the sense of (2b), every parameter s of the trefoil is globally in contact with precisely two distinct parameters that we denote by $\sigma(s)$ and $\tau(s)$. The functions σ and τ can be chosen such that the contact chords $c(s,\sigma(s))$ and $c(s,\tau(s))$ are continuous functions in s,

(H2) there is some $s \in I$, such that $\sigma(\sigma(s)) \neq s$,

(H3) the angle between the contact chords $\angle\left(c(s,\sigma(s)), c(s,\tau(s))\right)$ is uniformly bounded away from zero.

Since γ is a homeomorphism on its image, (H1) implies that $\sigma(s) = \gamma^{-1}\big[c(s,\sigma(s)) + \gamma(s)\big]$ and τ are both continuous as well.

We claim that σ (and likewise τ) is (locally) strictly monotone. Assume it is not monotone, so that it has some local extremum. Without loss of generality we assume that σ has a local maximum in s^*. Then there exist two sequences a_i, b_i close to s^* with $a_i < s^* < b_i$, such that $\sigma(a_i) = \sigma(b_i) \to \sigma(s^*)$ for $i \to \infty$. This implies, that the two different contact chords $c(a_i,\sigma(a_i)), c(b_i,\sigma(b_i))$ converge to the single chord $c(s^*,\sigma(s^*))$ contradicting (H3). If σ is not strictly monotone, then $\sigma(s) = c \in I$ has more than one solution, which would imply more than two contact chords from c, contradicting (H1).

Since I is a circle and σ is monotone, it is also surjective. By construction σ must be fixed-point free since $|c(s,s)| = 0 \neq 2\Delta$. This together with the monotonicity implies that σ is indeed injective and orientation preserving.

We claim $\sigma(s) \neq \sigma^{-1}(s)$ for all $s \in I$. First notice that $S = \{s \in I | \sigma(s) \neq \sigma^{-1}(s)\}$ is open and by (H2) non-empty. Now let $s_i \in S$ be some sequence with $s_i \to s$ and consider the contact chords $c(s_i,\sigma(s_i))$ and $c(s_i,\sigma^{-1}(s_i))$. By (H3) we deduce $\sigma(s) \neq \sigma^{-1}(s)$, i.e. S is closed as well, and therefore $I = S$.

Finally we claim $\sigma^{-1} = \tau$. At any given $s \in I$ consider the three potential contact chords $c(s, \sigma(s)), c(s, \tau(s)), c(s, \sigma^{-1}(s))$, The first two are different by (H1) and the case $c(s, \sigma(s)) = c(s, \sigma^{-1}(s))$ was excluded in the previous paragraph. It remains that $c(s, \sigma(s)) = c(s, \tau^{-1}(s))$, which resolves to $\tau^{-1} = \sigma$.

Recall that we may parameterize the union of all contact chords as the contact surface

$$\Sigma(s, h) = \gamma(s) + h\, c(s, \sigma(s)) = \gamma(s) + h(\gamma(\sigma(s)) - \gamma(s)),$$

for $s \in I, h \in [0, 1]$ and the contact curve [54]

$$\gamma_{1/2}(s) = \Sigma(s, 1/2) = \frac{\gamma(s) + \gamma(\sigma(s))}{2},$$

which is where the tubular neighborhood of γ touches itself.

Now we have to make a stronger assumption on the smoothness of σ and on the angles between the tangents along a contact chord :

(H4) $\sigma \in C^1(I, I)$,

(H5) $\gamma'(s) \cdot \gamma'(\sigma(s)) \geq 0 \quad \forall s \in I$.

Fix $h \in [0, 1)$ and consider the curve $\gamma_h(s) = \Sigma(s, h)$. By (H4) and (H5), we infer that it is a regular C^1-curve.

Next, we claim that γ_h is simple. We have to distinguish between the two cases $h \neq 1/2$ and $h = 1/2$. For the former, assume that there were distinct $s_1, s_2 \in I$ with $\Sigma(s_1, h) = \Sigma(s_2, h)$, i.e. $\gamma(s_1) + h\, c(s_1, \sigma(s_1)) = \gamma(s_2) + h\, c(s_2, \sigma(s_2))$. This would imply that the normal disks [25] of radius Δ around $\gamma(s_1)$ and $\gamma(s_2)$ ($h < 1/2$) or $\gamma(\sigma(s_1))$ and $\gamma(\sigma(s_2))$ ($h > 1/2$) would intersect, contradicting the assumed thickness. A double point of $\gamma_{1/2}$ would imply that the original γ touches a ball of radius Δ in four coplanar points from which we infer by Theorem C.1 in Appendix C that it is a circle.

We conclude that $\Sigma(s, h), h \in [0, 1)$ is a C^1-connected family of regular, simple curves, so they are all ambient isotopic (Apply Lemma C.1 in Appendix C to each fiber, use compactness and connectedness of $[0, 1-\varepsilon]$ and transitivity of isotopy). In particular $\gamma_{1/2}$ has the same knot type as γ.

Examples

In the previous section we derived an ambient homotopy between the contact curve and the knot itself using hypotheses (H1)-(H5). We now want to illustrate the homotopy using the approximate ideal Fourier trefoil $\widetilde{\gamma}$ from Section 5.5 of length $L[\widetilde{\gamma}] = 1$, since (H1) can only be approximately satisfied, and similar to [11, 66] we approximate σ by a function $\widetilde{\sigma}$ computed as follows : fix $s = 0$ and consider $f_s(t) = |\gamma(t) - \gamma(s)|$. Note that

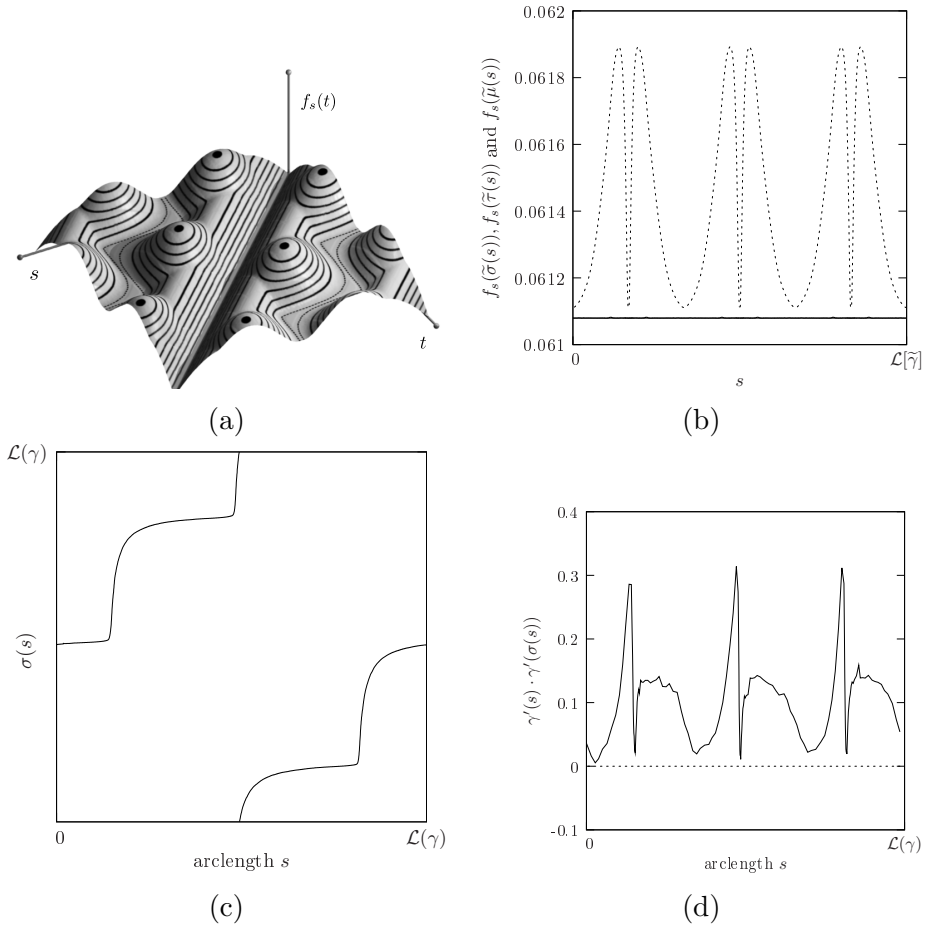


Figure 6.14: (a) The distance function $f_s(t)$ for $s, t \in I$. The dashed lines in the valleys indicate the two local minima, i.e. $\widetilde{\sigma}(s)$ and $\widetilde{\tau}(s)$. (b) Graph of the local minima $f_s(\widetilde{\sigma}(s))$ and $f_s(\widetilde{\tau}(s))$ (solid lines), which are smaller than the local maximum $f_s(\widetilde{\mu}(s))$ (dashed line) that separates them in the valley. (c) The periodic function $\widetilde{\sigma}(s)$ for an approximate ideal trefoil $\widetilde{\gamma}$. (d) A graph of the scalar product $\widetilde{\gamma}'(s) \cdot \widetilde{\gamma}'(\widetilde{\sigma}(s))$ between the tangents at s and $\widetilde{\sigma}(s)$.

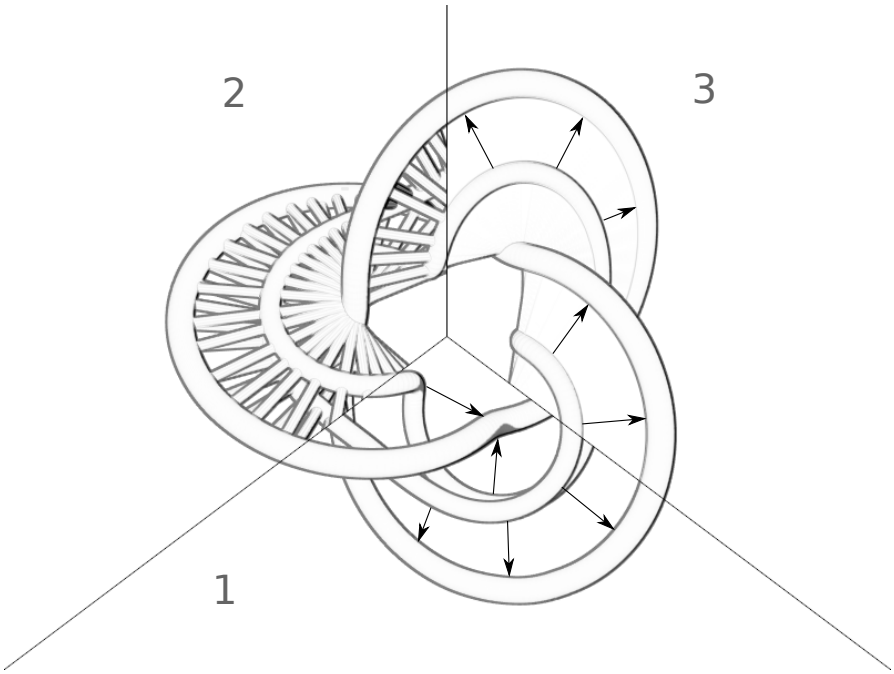


Figure 6.15: The trefoil homotopy visualized with the contact curve $\gamma_{1/2}$ (visible in 1-3); the chords $c(s, \sigma(s))$ are shown in (2) and the surface $\Sigma(s, h)$ in (3). Note that the tube radius of both the trefoil and the contact curve are much smaller than Δ, in order to reveal the contact set.

$f_s(t)$ is twice the *pp* function for fixed s. Numerics indicate that $f_s(t)$ has three local minima, in particular $f_s(s) = 0$ (see Figure 6.14 (a)). Pick the local minimum of $f_s(t)$ closest to 2Δ and set $\widetilde{\sigma}(s) = t$ for the corresponding t value. Sampling s up to $L[\widetilde{\gamma}]$ and restricting the minimization to a small neighborhood close to the previously found minimum yields $\widetilde{\sigma}(s)$. If the numerical shape satisfies (H1)-(H5), then $\widetilde{\sigma}(s) = \sigma(s)$. By picking the other minimum close to 2Δ we can extract $\widetilde{\tau}$ and compare it to $\widetilde{\sigma}^{-1}$ using linear interpolation. They coincide up to an error of 10^{-3}. Further we have that for all s, the local minima $f_s(\widetilde{\sigma}(s))$ and $f_s(\widetilde{\tau}(s))$ in the valley are separated by a local maximum $f_s(\widetilde{\mu}(s))$, which is always larger by at least $2 \cdot 10^{-5}$ than either of the local minima as depicted in Figure 6.14 (b). The way we extract $\tilde{\sigma}$ here is just another method compared to previous sections, and could instead be done based on the *pt* or *tt* functions.

We take this as a hint that (H1)-(H5) are reasonable. In particular for (H1), $|c(s, \widetilde{\sigma}(s))|$ deviates from 2Δ by less than $3 \cdot 10^{-6}$, while the dot product between the contact chords and the tangents is less than $4 \cdot 10^{-5}$. Assumption (H4) is suggested by Figure 6.14 (c), and (H5) by Figure 6.14 (d).

Figure 6.15 illustrates the homotopy. The different components of the image are the center curve of the trefoil - thickened to a fraction of the thickness Δ and the contact curve $\gamma_{1/2}$ in view (1). In the upper left part (2) a set of contact chords $c(s, \sigma(s))$ are

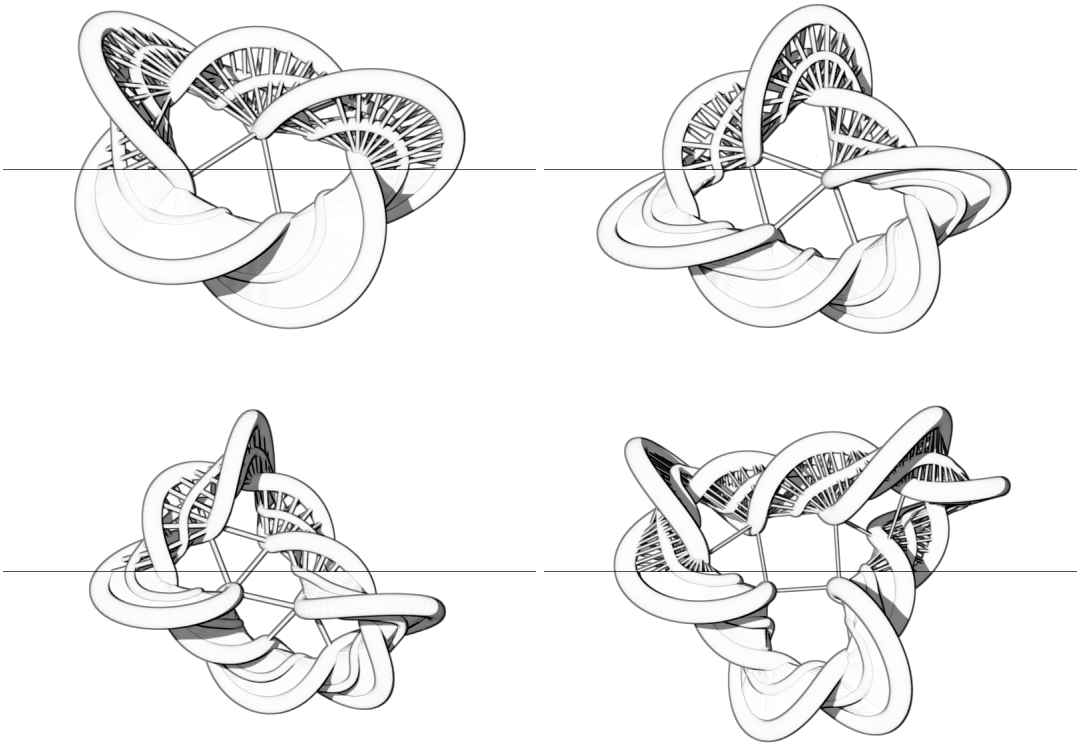


Figure 6.16: Contact chords and the contact surface visualized in the same image for the knots $T_{2,5}$, $T_{2,7}$, $T_{2,9}$ and $T_{2,11}$ (upper left to lower right). Note the isolated contact chords for these higher knots that are in addition to the continuous contact surface.

visualized as thin cylinders and in the upper right (3) we show the ruled surface, which is the previously defined $\Sigma(s,h)$, for $s \in I, h \in [0,1]$. The black arrows indicate the direction of the homotopy as h varies. This takes the trefoil center curve $\gamma(s)$ to the contact curve $\gamma_{1/2}(s)$.

The discussion about torus knots in Section 6.4 suggests that the construction carries over to these knots after filtering a few isolated contact chords. They appear in the center of the knot and are elements of the contact set in addition to the smooth contact surface and associated contact curve. In the same way, and also for the trefoil there may well be isolated points where local curvature is active. Figure 6.16 visualizes the contact contact chords, curve, surface for the knots $T_{2,5}$, $T_{2,7}$, $T_{2,9}$ and $T_{2,11}$. Note that they all belong to the class of torus knots $T_{2,2n+1}$.

6.7 On the angle condition for ideal knots

In previous sections we used numerical evidence to formulate mathematical claims such as the homotopy argument. The ultimate challenge is still to describe an ideal knot either by a closed form expression, or some differential equation that would allow efficient numerics. For the trefoil we have the following constraints: the symmetry group suggested in Section 3.3, the contact properties of the fundamental domain as discussed in 6.2, and assumptions (H1)-(H5) in the previous section. However, all this is still not sufficient to formulate a system of differential equations for the trefoil. This section is motivated by the fact that given the fundamental group α, β of a trefoil we can construct the rest of the knot and the pieces are related through contact. We would like to formulate a new set of constraints relating α and β, perhaps leading to a closed and consistent system of differential equations. The approach taken here is motivated by the mechanics of knots [39].

Let $T(s) \in \mathbb{R}$ be a tension along a curve $\gamma(s)$ in $\mathbb{R}^3$. The curve is unit speed parameterized, i.e. $|\gamma'(s)| = 1$, such that the tangent field is given by $\gamma'(s) = t(s)$. We use the change of the tension in the tangent direction along the knot and an external force density $F(s)$ to write the balance laws

$$\begin{aligned} & (T(s)\gamma'(s))' + F(s) = 0 \\ \Longrightarrow \quad & T'(s)\gamma'(s) + T(s)\gamma''(s) + F(s) = 0. \end{aligned}$$

Suppose that the force density $F(s)$, which is a force per unit arc length, is in the normal plane of $\gamma'(s)$ for each s, as is the case for double critical contact chords. Computing the dot product with $\gamma'(s)$ yields

$$T'(s) = 0.$$

Thus the tension T is constant along the curve $\gamma(s)$, i.e. $T(s) = T$. The equations are now

$$T\gamma''(s) + F(s) = 0,$$

or

$$T\kappa(s)n(s) + F(s) = 0,$$

where $n(s)$ is the principal normal and $\kappa(s)$ the curvature. In some sense the curvature force density $T\kappa(s)$ in the normal direction $n(s)$ wants the knot to become shorter. One can set the tension $T = 1$, this only rescales the force density F. In order for the curve to stay in an equilibrium state, we have to identify the force density $F(s)$ acting against the local curvature. For that we thicken the knot to the radius of its thickness Δ. Since the tube then starts touching itself, we relate the force density $F(s)$ to self contact.

For the shapes arising from one of the numerical methods we need to sample the curve $\gamma(s)$ with N points $\gamma(s_i)$. Figure 6.17 shows three of these points on γ, the principal normal n_i at $\gamma(s_i)$ and the angles θ_i and ψ_i between the principal normal and

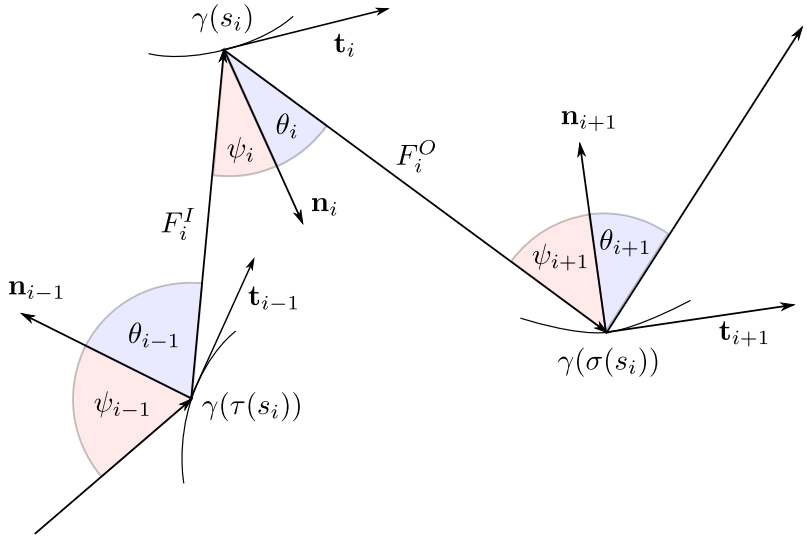


Figure 6.17: Illustrates the angle θ_i between the principal normal n_i at $\gamma(s_i)$ and contact chord $c(s_i, \sigma(s_i))$. The same holds for the angle ψ_i and chord $c(s_i, \tau(s_i))$.

the contact chord. We also define the forces per unit arc length F_i^O, the "outbound" force at s_i, and the "inbound" force F_i^I.

Projecting the force densities to the principal normal and the binormal at a given point $\gamma(s_i)$ we can write

$$\begin{aligned} F_i^I \sin\psi_i + F_i^O \sin\theta_i &= 0 \\ -F_i^I \cos\psi_i + F_i^O \cos\theta_i + \kappa_i &= 0, \end{aligned} \tag{6.3}$$

where we recall that we deal with forces per unit length. Written as N systems this reads

$$M_i y_i = b_i, \quad i = 1, \ldots, N$$

where

$$M_i = \begin{pmatrix} \sin\psi_i & \sin\theta_i \\ -\cos\psi_i & \cos\theta_i \end{pmatrix}, \quad y_i = \begin{pmatrix} F_i^I \\ F_i^O \end{pmatrix}, \quad b_i = \begin{pmatrix} 0 \\ -\kappa_i \end{pmatrix}.$$

By computing the determinants

$$\begin{aligned} \det M_i &= \sin\psi_i \cos\theta_i + \cos\psi_i \sin\theta_i \\ &= \sin(\theta_i + \psi_i) \end{aligned}$$

we can express the force densities as

$$y_i = \frac{1}{\sin(\theta_i + \psi_i)} \begin{pmatrix} \cos\theta_i & -\sin\theta_i \\ \cos\psi_i & \sin\psi_i \end{pmatrix} b_i.$$

At a sample point $\gamma(s_i)$ on the curve we therefore have the force densities

$$\begin{aligned} F_i^I &= \frac{\sin\theta_i}{\sin(\theta_i+\psi_i)}\kappa_i \\ F_i^O &= \frac{-\sin\psi_i}{\sin(\theta_i+\psi_i)}\kappa_i. \end{aligned} \tag{6.4}$$

From this we can derive a compatibility condition. Physically one expects that the outbound force at s is equal to minus the inbound force coming from the curve segment in contact and given by the contact function $\sigma(s)$. In order to derive a compatibility condition between forces, we equate the contributions of the forces per unit length of the intervals $[s, s+\delta s]$ and $[\sigma(s), \sigma(s+\delta s)]$:

$$\int_s^{s+\delta s} F^O(t)dt = -\int_{\sigma(s)}^{\sigma(s+\delta s)} F^I(t)dt.$$

Dividing the last equation by δs and taking the limit $\delta s \to 0$ yields

$$F^O(s) = -F^I(\sigma(s))\sigma'(s). \tag{6.5}$$

An analogous computation for $\tau(s)$ yields

$$F^I(s) = -F^O(\tau(s))\tau'(s).$$

When the force densities are eliminated by use of (6.4), condition (6.5) reduces to a purely geometric relation that should be satisfied on segments of ideal shapes with double contact chords.

We computed these force densities on a numerical approximation of the ideal Fourier trefoil (from Section 3.3) and a 5_1 knot, which is a slightly more annealed version of a 5_1 knot from E. Rawdon. For a partitioning s_i, $i = 1, \ldots, N$ of the interval $[0, 1]$, a discrete second order version of (6.5) is

$$\frac{1}{2}(F^O(s_{i+1}) + F^O(s_i))(s_{i+1} - s_i) = \frac{1}{2}(F^I(\sigma(s_{i+1})) + F^I(\sigma(s_i)))(\sigma(s_{i+1}) - \sigma(s_i)) \tag{6.6}$$

Figure 6.18 shows plots of the angle θ between the principal normal $n(s)$ and the outgoing contact chords $c(s, \sigma(s))$, the bottom row depicts the forces $F^O(s_j)(s_{j+1} - s_j)$ and $F^I(\sigma(s_j))(\sigma(s_{j+1}) - \sigma(s_j))$, with $j = 1, \ldots, N$, N being the number of biarcs, and F^I, F^O evaluated as in (6.4), for the trefoil and 5_1 knot. In Section 6.6 we suppose that every point on a torus knot is in contact with two other points on the curve plus a few isolated contacts. Here we implicitly take the same assumption. The knot 5_1 has two isolated contacts. At these points the curvature is balanced not only by two contact forces, but by three. Numerics makes it difficult to properly treat a point that has more than two contacts, which is why we discard isolated contacts. This introduces errors in the forces at points with isolated contacts, which are very few compared to the number of sample points we choose. The trefoil does not have any isolated contacts (only maybe

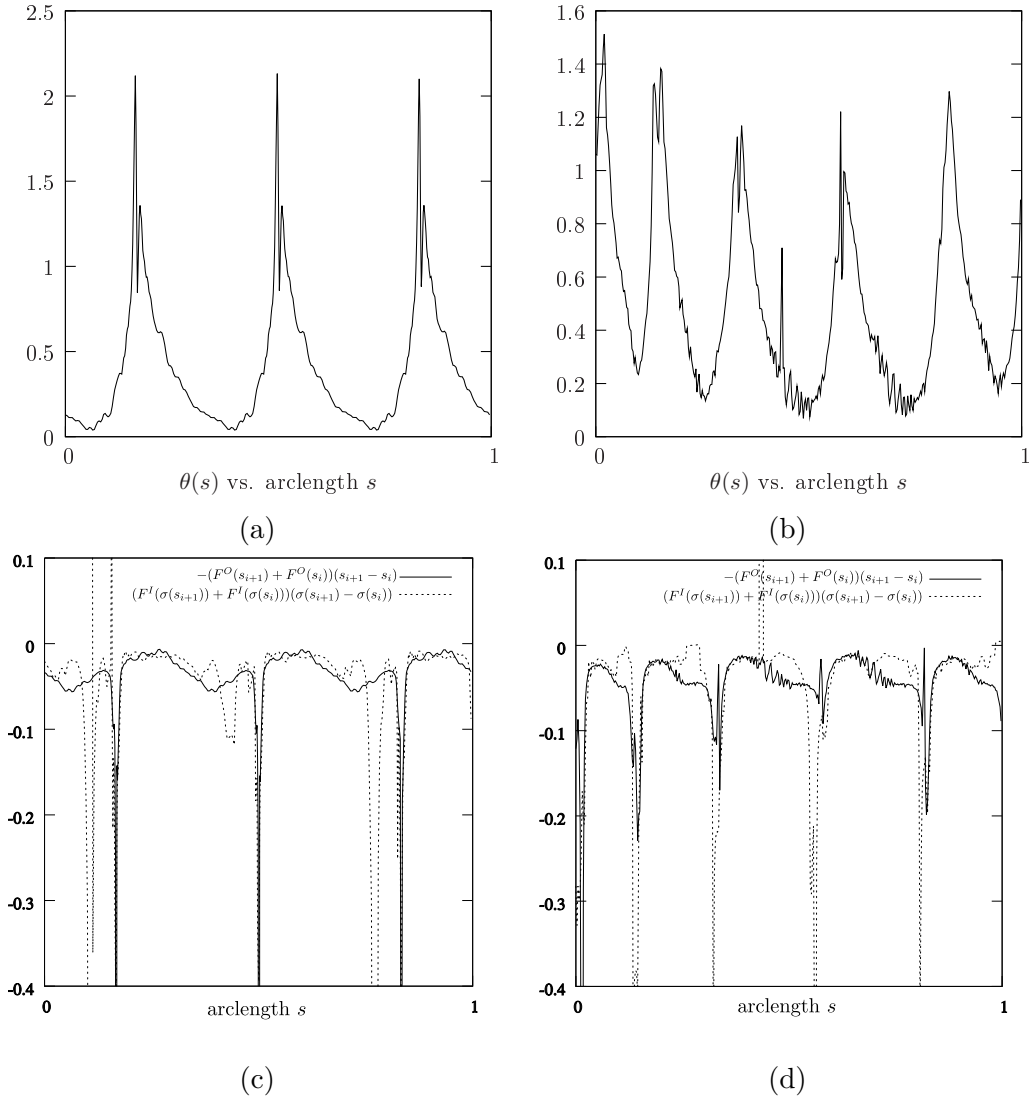


Figure 6.18: The left column corresponds to the Fourier trefoil and the right column to the 5_1 knot. In the upper row the angle θ between the principal normal and contact chord $c(s, \sigma(s))$ is shown. The bottom row is the actual angle condition, where the black line is the outbound force $(F^O(s_{i+1}) + F^O(s_i))(s_{i+1} - s_i)$, $i = 1, \ldots, N$ and the dashed line represents the inbound force $(F^I(\sigma(s_{i+1})) + F^I(\sigma(s_i)))(\sigma(s_{i+1}) - \sigma(s_i))$, $i = 1, \ldots, N$.

local curvature active at six points). The spikes and the noise in these graphs might be due to several things. If we consider the trefoil and its fundamental group as discussed in Section 6.2, then the large segment β is in contact with a small segment α and vice versa. This means that the sampling is always sparse in β and dense in α. Computation of the principal normals and the curvature is a delicate matter especially in the region where local curvature might be active, in particular on the short segment α. The angle condition relates information at α to β, which means that slight errors in normals or curvature values imply altered angles and forces and reflect as noise or even spikes in the graphs (see Figure 6.18).

The plots indicate that condition (6.5) is not satisfied everywhere. A different exaplanation than the one in the previous paragraph is, as mentioned in Section 5.5, that a small local perturbation of an ideal knot can significantly change its curvature profile while still being close to ideal. This argument also applies here since the curvature plays its role and the forces can be perturbed in the same way.

Remark 6.4. *Conditions (6.4) and (6.5) have a priori nothing to do with the minimization process of a knot with ropelength as the energy. A more in depth investigation is necessary to link the two problems. One could argue that the angle condition is in some sense a conditional curvature flow and therefore related to what algorithms like* `SONO` *or* `RidgeRunner` *do. But it is unclear whether they minimize the same energy.*

Chapter 7

Printing a contact surface in 3D

Given an appropriate meshed surface of a 3D object there exist various technologies for making the physical object. Such machines are called 3D printers. A model that can be printed in 3D needs to comply with a few rules. They might vary from one printer to another. The first thing is what is commonly called being "water tight". If we put the object in a water tank, it is not supposed to get filled with liquid. In other words, the surface has an inside and an outside and the mesh cannot have holes. A second condition is the so called manifold property, which, in short, says that every edge is part of exactly two polygons of the mesh and does not intersect another edge (excluding endpoints). And the normals to the polygons all need to point in the correct direction in order for the printer to distinguish the inside and outside of the object. A 3D printer will usually also have size constraints. Objects that are too large or have too fine structures, or smaller than some tolerance, cannot be printed.

7.1 Closing the surface

The triangulated contact surface as explained in Section 6.2 is not water tight. The sharp edge is not connected and we present hereafter two ways to close the surface.

Suppose that we have a set of contact chords c_i, $i = 1, \ldots, n$ for a knot γ and further that we are able to construct a smooth contact surface S_1. We now want to extract its border $\partial S_1 = \beta_1 \cup \beta_2$, where β_1 and β_2 are two curves, and build a triangulated surface S_2 between β_1 and β_2, such that $\partial(S_1 \cup S_2) = \emptyset$. Using the definition of a contact surface we can write $\beta_1(s) = \Sigma(s, \varepsilon)$ and $\beta_2(s) = \Sigma(s, 1 - \varepsilon)$.

The steps involved in extracting two boundary curves β_1 and β_2 are shown in Figure 7.1. First, the contact chords are shortened (b) and translated in the direction normal to the contact surface (c). Since the endpoints of the contact chords lie on the boundary, we are now able to construct β_1 and β_2. Pick a point p in β_1 and the closest point p^* to

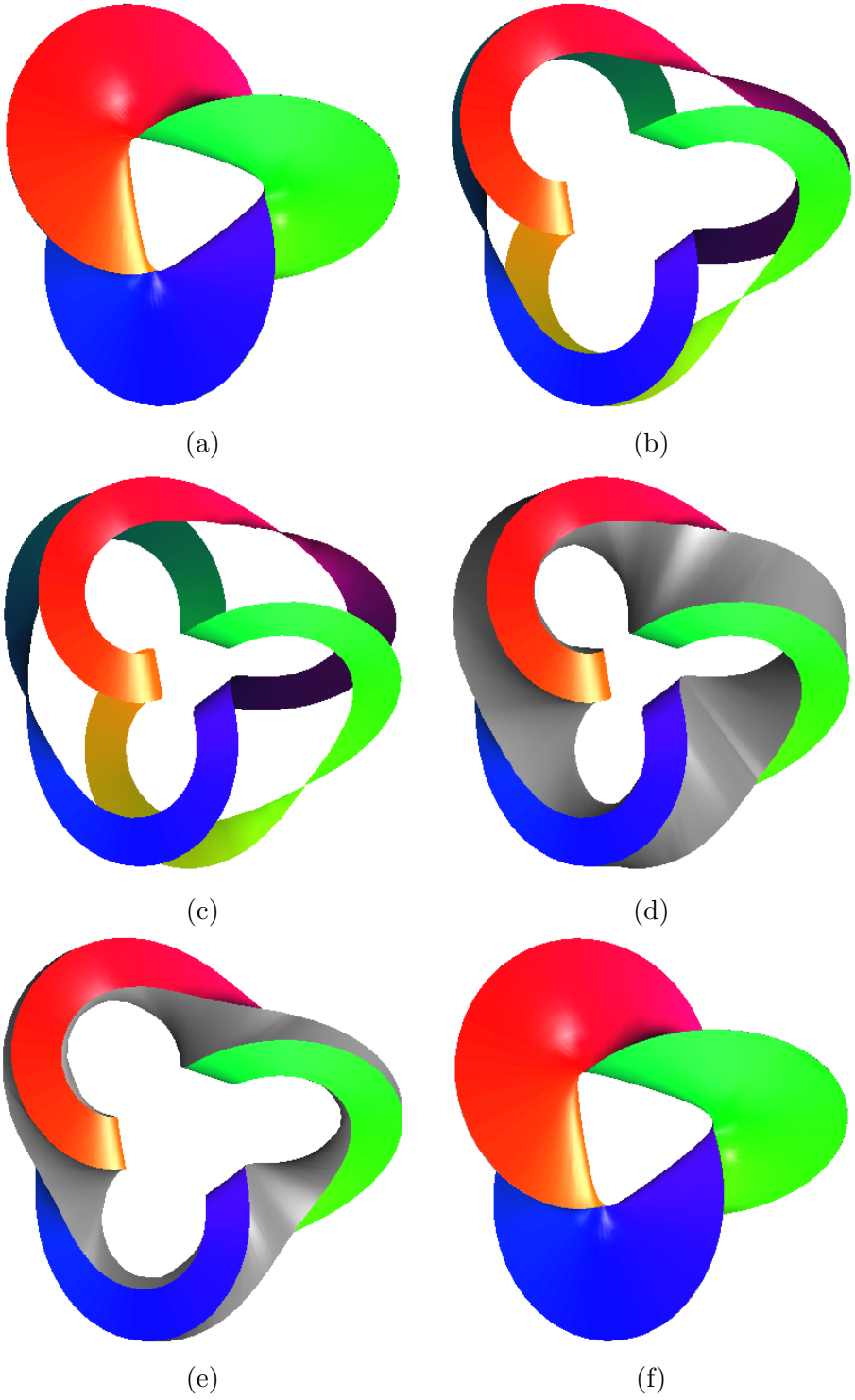


Figure 7.1: Procedure to close the contact surface of a trefoil by extracting two borders and constructing a triangulation between them.

p in β_2. If we follow β_1 and β_2 and triangulate the boundary we obtain the grey surface in (d). Restoring the original points for the contact chords (normal displacement (e) and shrinking (f)) gives a triangle mesh which is water tight and which has successfully been printed.

There are however several problems that can occur during this process. The value for shrinking is limited on both sides, if the contact chords remain almost the same, then if after shrinking the distance between points on the boundary β_1 or β_2 is larger than the distance between points on different boundaries, the boundary extraction algorithm will jump from β_1 to β_2 and vice versa. On the other hand, if the contact chords are too short, then we might jump from one endpoint of the contact chords to the other. A relatively fine uniform sampling can help to avoid both cases.

Another issue is the two starting points for β_1 and β_2. It is not obvious how to guess the first two close points on different edges. If after the double border extraction we realize that the distance between the first and last point of one border is larger than the distance between the first point on this border and either the first or last point on the other border, then we have to redo the process. This second time however we know two points on different edges, since the first and last point of one border is further apart.

The last trouble we might encounter is the triangulation. The direction of β_1 and β_2 could be opposite and the triangulation could be wrong. A simple reordering of either β_1 or β_2 solves this.

This method is a natural way of closing the contact surface but can still construct ill conditioned triangles (long and skinny). We used this algorithm for our first 3D printed models. For newer versions however a simpler way of closing the surface has been chosen. For that we used Blender and its feature “select non-manifold”. This exactly selects the boundary of our surface. Merging vertices with increasing tolerance values will eventually remove all the non-manifold regions and properly close the surface. The mesh loses some detail along the sharp edge when merging vertices, but the 3D printer usually can not print extremely small parts and the borders will lose some detail anyway.

7.2 Printing the surface in 3D

In an early stage of this thesis a first set of trefoil contact sets was printed in the Computer and Visualization Laboratory group of G. Abou-Jaoudé at EPFL. The models were made of paper and then in plaster. At the time, we used the border extraction method explained in the previous section to close the contact set.

Then we discovered the online 3D printing service Shapeways [63]. It is possible to upload 3D meshes to their website. The models are checked and tested to determine if they can be printed (normals pointing outwards, manifold property, size constraints). They propose a variety of materials to choose from. We decided to order a few models

in a material called "White, Strong & Flexible", which is a nylon kind of material, the strongest currently available on their website. It is also possible to print objects in metal, but for that, the wall thickness has to be at least 1mm. We did not order an object in metal yet.

The objects printed by Shapeways are a simple trefoil, which is naturally closed and with correctly oriented normals. We can now produce contact surface meshes for other knots than just the trefoil. The approach to close the surface for Shapeways the same as with Blender, where we lose some fine structure along the sharp edges, which is not problematic since the printer has a hard time creating the sharp edge anyway. In Figure 7.2 we show photos by B. Favre of a Shapeways printed trefoil knot, and the contact surfaces for the trefoil, 5_1, 7_1 and 9_1.

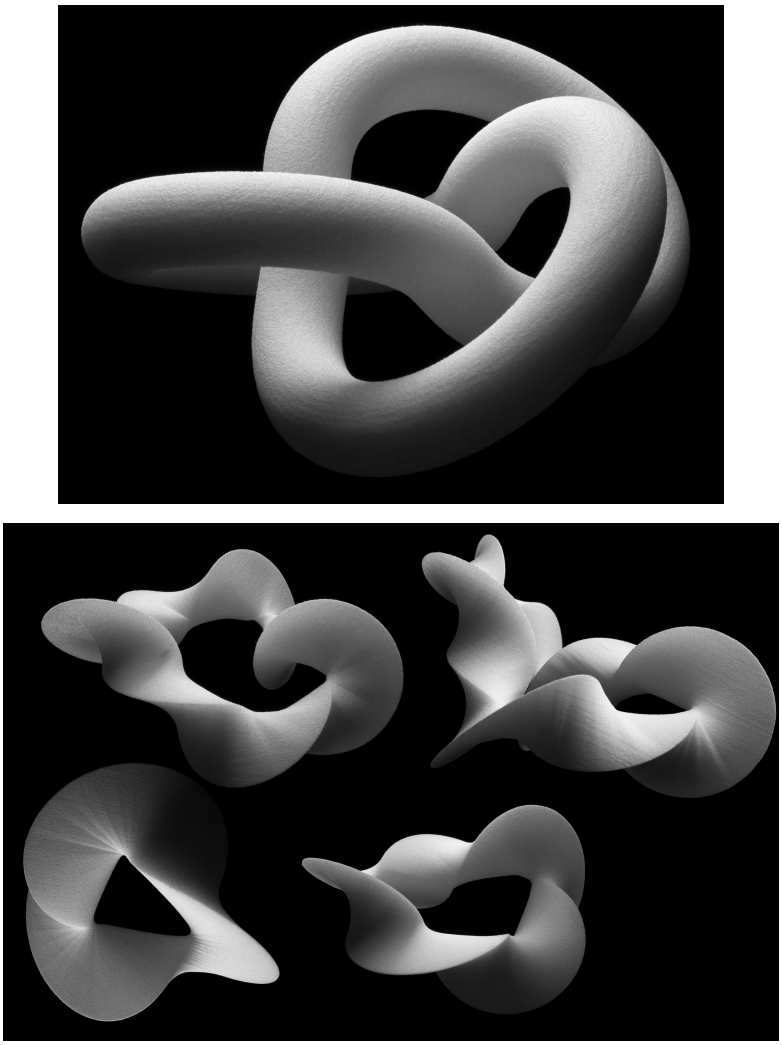


Figure 7.2: Trefoil model (top) and contact surfaces for 3_1,5_1,7_1 and 9_1 (bottom).

Chapter 8

Conclusion

In this thesis we exploit the interplay between scientific computation and visualization to enhance the understanding of ideal knot shapes.

Chapter 2 introduced the necessary background material, including the notion of a space curve, a thick curve and the concept of ideal knot shapes. We first defined a tubular neighborhood of radius $r > 0$ around a closed, smooth curve, which is a thick curve or tube. The largest radius r for which such a neighborhood is injective is called the thickness Δ of the curve. We relate the thickness Δ to the smallest radius of a circle through three points on the curve (the *pt* function). Ideal knots are ropelength (L/Δ) minimizers of a given isotopy class. Only a few ideal shapes are known analytically, such as the circle and the Hopf link. So we approach this question numerically using the biarc discretization (Section 2.2), mainly following what has been developed in [66] but extended to $\mathbb{R}^N$ so as to be able to compute in $\mathbb{S}^3$, i.e. the three sphere embedded in $\mathbb{R}^4$. A closed curve approximated by biarcs is in $C^{1,1}(\mathbb{S}, \mathbb{R}^3)$, which is known to be the appropriate regularity of ideal knots.

In Chapter 3 we discuss Fourier knots, which are simply a different parameterization of a closed space curve. An appealing property of this representation is that we can enforce symmetries on knot shapes. Starting from prior computations of various types, we conjecture the symmetries of the trefoil, the figure-eight and the 5_1 knot and show that a symmetrization of the knots with respect to these symmetries is visible in the Fourier coefficients. In particular these coefficients are no longer independent, and several vanish. This property proved to be helpful in the numerical part of this thesis, where we could speed up the computation of approximately ideal knot shapes.

Before the presentation of the numerical computations of ideal knots in Chapter 5, a discussion of visualization in Chapter 4 explained the design of a specialized color gradient, which is used to emphasize detailed structure in the *pp*, *pt* and *tt* plots of ideal knots. Then `curview` is described, which is a curve viewer program included in the `libbiarc` library. A simple example is presented in order to familiarize the reader with

the 2D plots *pp*, *pt* and *tt* that arise in the Chapters 5 and 6 about computations and contact, respectively. The last part of Chapter 4 describes a possible way to visualize curves in $\mathbb{S}^3$.

Chapter 5 deals with numerical algorithms to approach ideal knot shapes. The algorithms mentioned are SONO, `RidgeRunner` and simulated annealing. Simulated annealing on knots was first done by B. Laurie [36]. Our simulated annealing code, based on the biarc discretization, is part of the `libbiarc` and has been augmented beyond previous biarc simulations by a coordinated non local move set based on the Fourier representation of the knot. Since we know how to efficiently evaluate thickness on a biarc approximated curve, we switch between the two representations when annealing Fourier coefficients. In this sense, annealing a Fourier knot is just a particular, global, annealing move, since we always compute the objective function (the ropelength) on a biarc curve. The thickness evaluation algorithm and a parallel implementation concludes this chapter. Thickness evaluation seems to be the bottleneck in our computations, so parallelization of this part is worthwhile.

Now that we possess a set of reasonably converged ideal knot shapes of various isotopy classes – we usually computed with all the knots up to nine crossings – we investigate their contact set properties. Chapter 6 has three substantial parts. First we defined the notion of a contact set CS, which informally is where the tubular neighborhood touches itself. This is also based on the *pt* function, where a pair $(s, \sigma) \in CS$ if $pt(s, \sigma) = \Delta[\gamma]$. We then discuss the contact sets of the trefoil and a few other torus knots, where the midpoints of the contact chords form a curve. In the second part, Section 6.6, we show under a few assumptions, that the contact curve of the trefoil is ambient isotopic to the knot, and is therefore in the same knot class. The contact set of the figure-eight knot is fundamentally different, since there are two disjoint components. For torus knots we believe that for fixed s every $pt(s, \cdot)$ cut has exactly two global minima plus a few isolated contacts. The figure-eight knot does not have this simple structure and there are regions that do not seem to have contact at all. This does not contradict known results provided that the curve segments that are not in contact are straight.

In the investigation of the trefoil contact properties we discovered a closed trajectory when following iterations of the contact function $\sigma(s)$, starting from $s = 0$, which is on the π rotation symmetry axis. This trajectory is a closed 9-billiard and partitions the trefoil into nine curve segments, where only two (called α and β) are independent, the remainder being constructed by simple symmetries. This partitioning of the knot is significant because it also partitions the contact set. We have not carefully checked whether such billiards exist on other $\mathbb{R}^3$ knots.

In the third and final part of Chapter 6 we discussed an angle condition for ideal knots. In order to derive a set of differential equations for the ideal trefoil the 9-billiard narrows the problem down to parameterizing α and β, and relating them with the

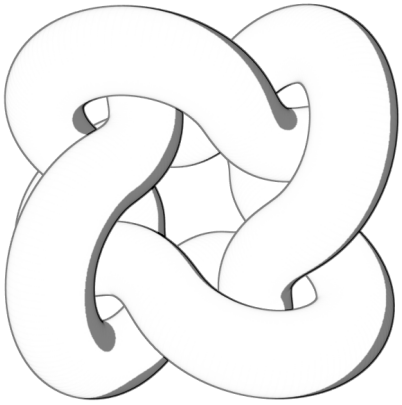


Figure 8.1: Knot 8_{18} seen from a possible symmetry axis.

contact function σ and the angle condition presented in Section 6.7. At every point $s \in \mathbb{S}$ along the knot, curvature, inbound and outbound forces balance in a certain explicit sense, which involves the local curvature, the incoming $c(s, \tau(s))$ and outgoing $c(s, \sigma(s))$ contact chords, and the angle between the chords and the principal normals at $s, \sigma(s)$ and $\tau(s)$. We have a compatibility condition, since there are two different ways of computing the same force: first as the outbound force at s and second as the inbound force at $\sigma(s)$. As observed, comparatively large changes in curvature can be made locally with only very small associated changes in thickness. As these balance laws involve the curvature, they might offer a smoothing algorithm for ideal knots already close to ideal.

In the final Chapter 7 we describe how the contact surface meshes obtained in Chapter 6 can be 3D printed. Surprisingly, a physical model of the contact surfaces showed aspects about the surface that we had not realized by looking at them on a computer screen. For example, the contact curve of the more complicated torus knot $T_{2,11}$ starts becoming less and less planar, while we wrongly assumed the contrary when inspecting them via visualization on a computer screen.

Several directions of future work are opened up by this thesis. Analytically, the ingredients for a closed form expression of the trefoil might now be close to being found. We want to find two curves $\alpha(s), \beta(t) \in C^\infty(I, \mathbb{R}^3)$ such that we have C^1 boundary matching conditions at the endpoints. Is curvature active at the endpoints of α and β? Do we have zero curvature at the midpoint of α? The angle condition implies a second copy of β, but for that we need the contact function $\sigma(s)$. An analytic expression for σ seems elusive.

From a numerical point of view, futher computations on higher order knots using biarcs and the `libbiarc` for thickness evaluation are still desirable. In addition to what we have learned from the trefoil, better contact sets of higher order knots could lead to

a generic understanding of how to partition a knot's contact set, considering its regions of possibly active or vanishing curvature, straight pairs of double critical chords and symmetries, which, at least to our knowledge, are less present in more complicated knots. Nevertheless symmetries do arise, for example the knot 8_{18} shown in Figure 8.1 is a very good candidate for Fourier annealing, since the symmetries visible in Figure 8.1 are clear. The principal drag axis as computed in [10, 23] indicate projections along which the knot shapes might show some symmetry.

Appendix A

libbiarc

The library is available as a static Mercurial [50] repository at

```
hg clone static-http://lcvmwww.epfl.ch/libbiarc/
```

or as a tar ball at `lcvmwww.epfl.ch/libbiarc/tars/libbiarc-0.1.tgz`. There is an extensive documentation at `lcvmwww.epfl.ch/libbiarc/doc/html` explaining the different API calls and tools included. How to install and generate the documentation is also explained on that web site.

A.1 Command line options

The viewer `curview` in the `inventor` directory is started with at least one filename of a PKF file and the following optional arguments

```
-N=NODES              number of points on the curve
-S=SEGMENTS           cross sectional segmentation
-R=RADIUS             tube radius
-Tol=TOLERANCE        mesh adjustment tolerance
-closed               closes all the curve
-iv-scene <file.iv>   add a .iv scene-graph to the scene.
-texture <img_file>   texture map the pkf curve
-whitebg              white background color
```

where in order of appearance we have the number of data nodes, the cross sectional segments, the radius of the tube, and a tolerance value used for the mesh creation. Further the flag `-closed` is used to treat all the curves as closed. Right now it is not possible to mix open and closed curves in the same scene. A very handy option is the

`-iv-scene` argument. We can ask the viewer to load an inventor scene file (.iv) to add supplementary objects like spheres, line sets or surfaces. This has often been used to analyze specific knot shapes, and for example their contact properties. The next option permits the user to map a texture image on the tube, and the last option switches the background color from default black to white, which has been used to produce the images for the thesis.

A.2 Key bindings

Here we describe the key bindings for `curview`, and their actions where letters `A-Z` are case insensitive.

Keys `F`, `V` : Increase (`F`) and decrease (`V`) transparency of the tube.

Keys `A`, `Y` or `Z` : Increase (`A`) and decrease (`Y/Z`) the radius of the tube by 10%.

Keys `S`, `X` : Change the number of circular segments.

Keys `D`, `C` : Resample the curve with more (`D`) or less (`C`) data points.

Key `1` : Cycle through different curve framings (Frenet frame, parallel transport or writhe framing).

Key `L` : Open or close a curve. This is only available in `BIARC` view mode.

Key `Return` : For this we need to be in `BIARC` mode. Creates a new curve if currently there is none. If there is already a curve object, then the viewer adds a new data point in the direction of the last tangent. Only single component curves can be edited.

Key `Delete` : Removes the last picked data point in `BIARC` mode.

Key `Space` : Cycles through three different viewing modes: `SOLID`, `WIRE` and `BIARC`. The first is the standard view, `WIRE` shows the mesh of the tube and the `BIARC` representation draws the curve as a line with little red spheres (actually octahedrons) for the data points, blue spheres for the biarc matching points. The tangents at every point are yellow at data points and green for the matching tangents.

Key `W` : Exports the current state of the curve to the file `curve.pkf`. The curve is correctly oriented with respect to the current camera view.

Key `R` : This function is only available in `BIARC` mode. Resamples the curve between two selected data points with 10 nodes. First hit `R`, then click on the two locations where you need to change the sampling of the curve.

Key `2` : Displays the inertia axes [27] of the curve. If the modifier `CTRL` is pressed, then it orients the curve with respect to these axis. Repeating this, cycles through the three different axes.

Keys `I, O, P` : Opens a different window with either a *pp* (`O`), *pt* (`P`) or *tt* (`I`) plot.

Key `Q` : Close the viewer.

A.3 Annealing

As detailed in Chapter 5, simulated annealing is an algorithm to find a global minimum in a configuration space by a coordinated random search. The principle is simple, change the configuration and check whether the new energy associated to the problem is lower. If it is the case, we accept the change, if not we accept it according to some probability. In simulated annealing this probability is given by a Boltzmann distribution and depends on the current temperature. It is not a very efficient method, but gives good results depending on the underlying problem without having to compute any gradient of the objective function. These annealing base classes are available in the directory `experimental/annealing` in the `libbiarc`.

A.3.1 Implementation

Since we use simulated annealing for different problems, the need of an abstract annealing structure was necessary. All the core routines and parameters are defined in abstract base classes. Every particular problem could then rapidly be implemented deriving from these classes. It is no longer necessary to reprogram the whole simulation machinery for every new problem. The main classes are

```
class BasicMove
class BasicAnneal
```

The changes that can be made during an annealing run need to be derived from `BasicMove`. This class contains the routine `move`, to be reimplemented in the derived classes, `accept` and `reject` which are called by the main annealing class. They adjust the step size associated to every move object and undo rejected moves. A given problem might have various orders of magnitudes when it comes to moves, that is why the step size is very important to control it. The constructor initializes the step sizes and the step change, which is performed depending on whether a move gets accepted or not. Derived classes would contain problem specific data, like floating point values or entire structures that get changed in the annealing process.

The simulation task itself is derived from `BasicAnneal`. The annealing class has an array of possible moves or changes and the interface for that is implemented in `BasicMove` as explained earlier. Once the necessary moves are implemented, the only routines from `BasicAnneal` that have to be reimplemented are `energy` and probably `best_found`, `stop` and the constructor to define new parameters for the problem.

A word about parameter handling. Parameters are given as a string `params` with key=value pairs delimited by a comma. Example : `T=.01,N=10,best_filename=best.txt`. The different key, value pairs get extracted in the initialization with the helpers

```
// Extract strings
#define extract(Var,params)
#define extract2(Var,Str,params)
// Extract integers
#define extract_i(Var,params)
#define extract_i2(Var,Str,params)
// Extract floats
#define extract_f(Var,params)
#define extract_f2(Var,Str,params)
```

There are two versions for extracting strings, floats or integers. The first one assumes that the variable name `Var` is the same as the name in the parameter string, the other version can extract variables where the names differ. It looks for the string `Str` in `params` and initializes `Var` correctly. To complete the presentation of the base annealing class, there are a few other routines. The method `show_config` prints the parameter settings, `accept_curr_moves` and `reject_curr_moves` are called if the current configuration gets accepted or not. `wiggle` performs a random move, `update_minmax_step` controls the move step sizes, `logline` prints simulation information during the run. A call to `do_anneal` starts an actual simulation.

A.3.2 Toy problems

To demonstrate the versatility and ease of use of this abstraction, we implemented three different toy problems. Further investigating these problems would be another research project. The only reason we present these toy problems is to illustrate the way of using the base class structure.

Dot repulsion

In this problem we try for N dots $x_i \in \mathbb{R}^2, i = 1, \ldots, N$, to minimize the energy

$$E(x_1, \ldots, x_N) = \sum_{i,j=1}^{N} (2 - |x_i - x_j|)^2.$$

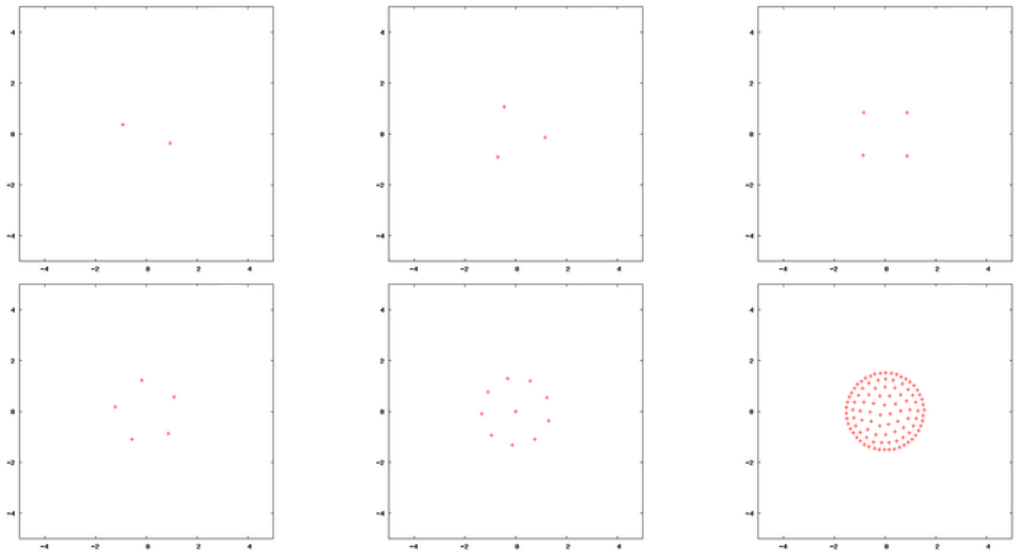


Figure A.1: Annealing results for the dot repulsion problem for $N = 2, 3, 4, 5, 10, 100$.

This means that every point wants to be at Euclidean distance 2 from all other points. A move for a point x_i is implemented as two float moves stored as two `SimpleFloatMove` objects. The class `SimpleFloatMove` is derived from `BasicMove`. The simulated annealing has an additional parameter N, the number of dots and is taken care of in the constructor. The simulation stops if the temperature gets too low. In `best_found` we write the coordinates for all the points to `best_filename`. Figure A.1 shows the minimal state found for $N = 2, 3, 4, 5, 10$ and 100 points.

Square box problem

In the second problem (still in $\mathbb{R}^2$) the goal is to find the smallest square box in which we can fit N smaller square boxes all of the same size. The only additional parameter is the number of boxes N just as in the previous example. Then we define a `Box` class containing the center (2 floating points) and an orientation angle (1 float). The edge size of a box is fixed to one. The moves are implemented as three `SimpleFloatMove`, acting on the center coordinates and the angle of the boxes.

The annealing class initializes the N boxes in a random fashion and sets up the possible moves. In the wiggle routine we move and rotate a box, accepting only non overlapping configurations. The energy is given by the edge length of the bounding square box surrounding all the smaller boxes.

Figure A.2 shows the results for $N = 2, 3, 4, 5, 10, 17$. The results are correct up to $N = 10$, where the optimal edge length has been proven to be $3.707(3 + 2^{-1/2})$ [15]. This is an indication that the energy function used in this problem could further be

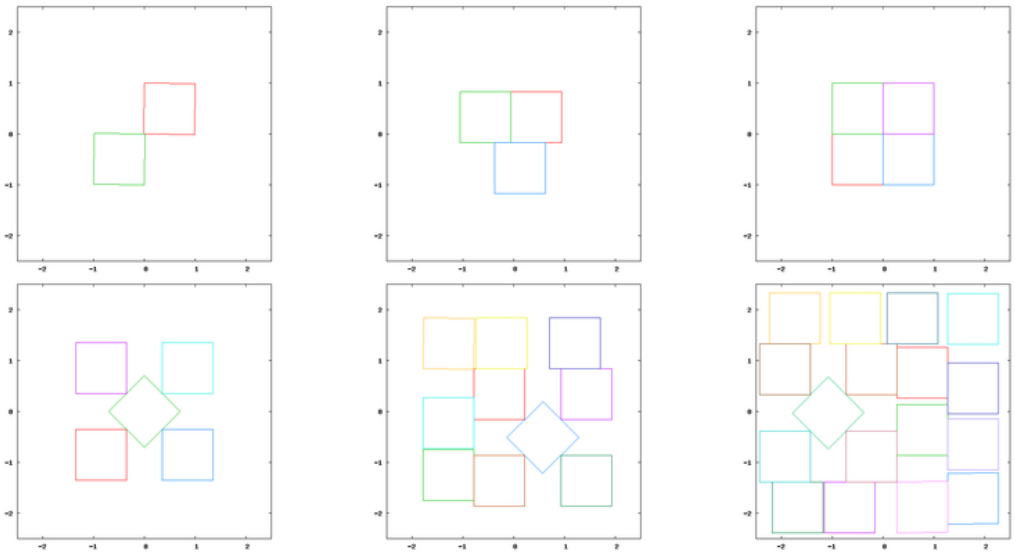


Figure A.2: Annealing results for the box problem with $N = 2, 3, 4, 5, 10, 17$.

improved.

Curve fitting

This last example program has to some extent been used to fit a monotone contact function for the ideal trefoil $\sigma(s)$. Given a list of points $(x_i, y_i) \in \mathbb{S} \times \mathbb{S}, i = 1, \ldots, N$ we try to fit a monotonely increasing function

$$f : \mathbb{S} \longrightarrow, \mathbb{S} \qquad s \mapsto f(s)$$

with periodic boundary conditions in both directions. We try to stay as close as possible to the data points in a least squares sense. The energy functional is therefore

$$E(x_1, \ldots, x_N, y_1, \ldots, y_N, f) = \sum_{i,j=1}^{N} (y_i - f(x_i))^2.$$

The monotone function f is given by M points and linearly interpolated between data points.

For the additional annealing parameters we need a parameter M for the number of data points we use to define f. This time we also have two other parameters, `filename` contains the data we want to fit and `resume_from` can be used to continue a previously stopped run.

Appendix B

Blender Python plugins

A variety of 3D modeler and renderer softwares exist on the market, such as Alias' Maya, 3D Studio Max, and many more [40, 42]. In particular there is Blender, an open source 3D studio [28] with good functionality and tools. Almost all of these programs have some mechanism for extending their functionality either with binary plugins or a scripting language, sometimes proper to each package. In Blender there are two ways of extending its functionality, either with binary plugins or one can simply write plugins in plain Python using the Blender Python API. This API permits the user to easily create and manipulate objects like primitives or meshes or to edit directly material, modifier or animation properties. The Blender documentation states that the preferred way of extending Blender is with scripting, which is what we will present here.

A Blender Python script usually contains a part where one deals with the geometry or the objects to be generated and modified in a scene, then a GUI part responsible for the look and feel of the plugin, and finally an event management system, where one registers event handlers. The scripts are written or loaded in Blender's `Text Editor` window, where they can then be executed.

In the `libbiarc` package, scripts for importing PKF curve files into Blender are located in the `experimental/blender` directory. We now give a brief description of what they do.

pkfanim.py The script loads a sequence of PKF files from a directory. Then, using the radius and the number of nodes along the centerline specified in the GUI, it generates a tubular mesh object for every PKF found in the directory. Finally the meshes are included in the scene as a Blender animation.

pkfcurve.py Here we import a PKF curve as a Blender trajectory or curve object. This object could then be used for beveling or trajectory following.

pkfmesh.py This tool is probably the most useful if one only wants to produce a still

image of a given curve. It produces a Blender mesh from a PKF file similar to the animation script, but only for a single file.

struts.py A very handy tool to import contact chords as small cylinders.

wire.py The purpose of this script is purely artistic. It creates a wire frame like object of the tubular mesh of a curve with cylinders. Then it adds spheres as joints at the data points.

Two rendered examples produced using the above scripts are shown in Figure B.1.

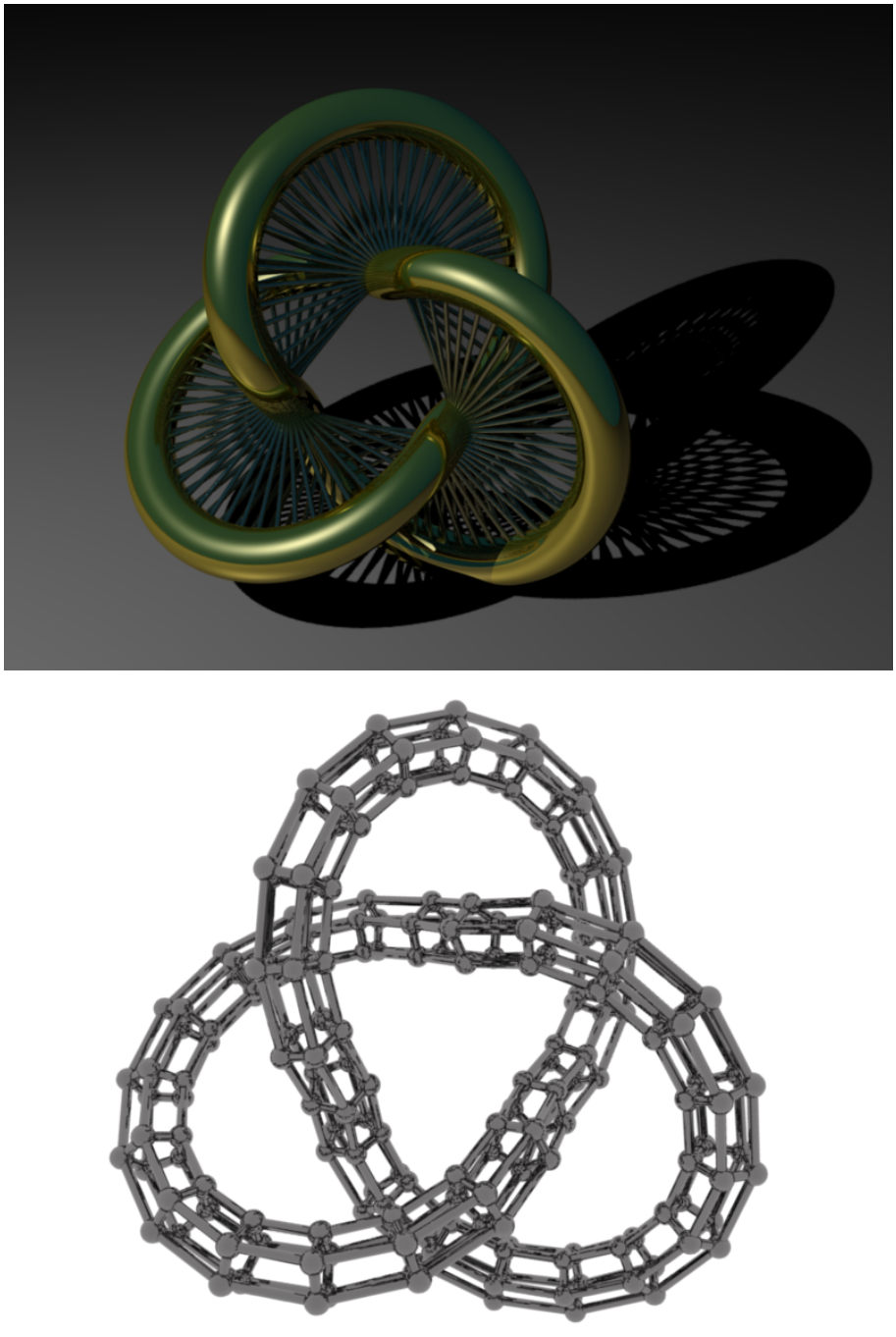


Figure B.1: Two pictures rendered with Blender and constructed with the plugins presented in this section.

Appendix C

Two theorems

For the readers convenience we included the statement of two results used in this thesis, and cited in the references, but which might not otherwise be easily accessible.

Theorem C.1. *[18, Satz 3.27] Let $\gamma \in C^1(\mathbb{S}^1, \mathbb{R}^N)$ be a closed curve of positive thickness $\Theta := \triangle[\gamma] > 0$. And let $B_\Theta(C)$ be an open ball around some center $C \in \mathbb{R}^N$, such that $B_\Theta(C) \cap \gamma(\mathbb{S}^1) = \emptyset$. Now let there be two distinct points $X, Y \in \partial B_\Theta(C) \cap \gamma(\mathbb{S}^1)$, which are not antipodal on $\partial B_\Theta(C)$ (i.e. $|X - Y| < 2\Theta$).*

Then γ joins X and Y by a geodesic arc of radius Θ running on $\partial B_\Theta(C)$.

Lemma C.1. *[58] Let $\eta \in C^1(\mathbb{S}^1, \mathbb{R}^3)$ be a regular simple closed curve. Then there exists a constant $\varepsilon^* > 0$ depending on η such that all $\zeta \in C^1(\mathbb{S}^1, \mathbb{R}^3)$ with $||\zeta' - \eta'||_{C^0(\mathbb{S}^1, \mathbb{R}^3)} \leq \varepsilon^*$ are ambient isotopic to η.*

Bibliography

[1] Adams, C., The Knot Book W. H. Freeman, New York, (1994).

[2] Ashton T., Cantarella J., Piatek M., and Rawdon E., Self-contacts sets for 50 tightly knotted and linked tubes mat.DG/0508248, (2005).

[3] Arikan, O., Pixie OpenSource RenderMan, `www.renderpixie.com`, (21 september 2009).

[4] Barth N., The Gramian and K-Volume in N-Space: Some Classical Results in Linear Algebra JYI Vol. 2, Issue 1, (1999).

[5] Berger M., Geometry I+II, Springer-Verlag, New York (1987).

[6] K. M. Bolton, Biarc curves, Computer-Aided Design **7** (1975) 89-92.

[7] Buck, G.; Simon, J., Energy and length of knots, in Lectures at Knots96, ed. Suzuki, S., World Scientific Publishing, Singapore (1997) 219–234.

[8] Cantarella, J.; Kusner, R.B.; Sullivan, J.M. On the minimum ropelength of knots and links. Inv. Math. **150** (2002), 257–286.

[9] Cantarella J., Piatek M., Rawdon E., Visualizing the tightening of knots In VIS '05: Proceedings of the 16th IEEE Visualization 2005 (VIS'05), pages 575-582, Washington, DC, USA, 2005. IEEE Computer Society.

[10] Carlen, M., Computation and visualization of stokes flow of knotted filaments, Master's thesis, EPFL, Lausanne (2005)

[11] Carlen, M.; Laurie, B.; Maddocks, J.H.; Smutny, J., Biarcs, Global Radius of Curvature, and the Computation of Ideal Knot Shapes, in *Physical Knots*, Eds. J. Calvo, K. Millett, E. Rawdon, and A. Stasiak, World Scientific (2001), 153–162.

[12] Do Carmo, M.P. Differential Geometry of Curves and Surfaces. Prentice Hall, New Jersey (1976).

[13] Folland, G.B., Fourier Analysis and Its Applications, Brooks/Cole Publishing Co. (1992).

[14] Michael H. Freedman, Zheng-Xu He, and Zhenghan Wang., Möbius energy of knots and unknots, Ann. of Math. (2), 139(1):1–50, (1994).

[15] E. Friedman, Packing unit squares in squares: a survey and new results, The Electronic Journal of Combinatorics DS7 (2005).

[16] Galer, M., Horvath, L., Digital imaging : essential skills 3rd ed., Focal Press, Oxford, Boston (2005).

[17] F. R. Gantmacher, Matrix Theory, Vol. 1, Chelsea Publishing Company, New York (1959).

[18] Gerlach, H. Der Globale Krümmungsradius für offene und geschlossene Kurven im $\mathbb{R}^N$, Diploma thesis at Bonn University 2004.

[19] Gerlach, H. Ideal Knots and Other Packing Problems of Tubes, PhD thesis to appear 2010, `http://library.epfl.ch/theses/`, EPFL Lausanne

[20] Gerlach H., von der Mosel, H., What are the longest ropes on the unit sphere?, Preprint Nr. 32 Institut f. Mathematik, RWTH Aachen University (2009).

[21] Golub, G. H.; Van Loan, C. F., Matrix Computations (3rd ed.), Johns Hopkins University Press, Baltimore, MD, USA, (1996).

[22] Gonzalez, O. and De la Llave, R., Existence of ideal knots, Journal of Knot Theory and Its Ramifications **12** (2003) 123–133.

[23] Gonzalez, O., Graf, A.B.A. and Maddocks, J.H., Dynamics of a rigid body in a stokes fluid, J. Fluid Mech **519** (2004), 133–160.

[24] Gonzalez, O., Maddocks, J.H., Global Curvature, Thickness and the Ideal Shapes of Knots, Proc. Natl. Acad. Sci. USA **96** (1999), 4769–4773.

[25] Gonzalez, O.; Maddocks, J.H.; Schuricht, F.; von der Mosel, H. Global curvature and self-contact of nonlinearly elastic curves and rods, Calc. Var. **14** (2002), 29–68.

[26] Gonzalez O., Maddocks J.H., Smutny J., Curves, circles, and spheres, Contemporary Mathematics 304 (2002) 195-215

[27] Gruber, C. Mécanique générale Presses polytechniques et universitaires romandes, Lausanne (1988).

[28] Hess, R., The Essential Blender: Guide to 3D Creation with the Open Source Suite Blender, No Starch Press, San Francisco, CA, USA, (2007).

[29] V. Katritch, J. Bednar, D. Michoud, R. G. Scharein, J. Dubochet and A. Stasiak, Geometry and physics of knots, Nature **384** (1996) 142–145.

[30] Kauffman, L.H., Fourier Knots, Chapter 19 of [67].

[31] Kim M., Wood S., The MPEG-4 Book, Prentice Hall (2002).

[32] Kirkpatrick, S., Gelatt C. D., Vecchi M. P. Optimization by Simulated Annealing, Science, New Series 220 **4598** (1983) 671–680.

[33] Kongsberg SIM AS, `www.coin3d.org`, (21 september 2009).

[34] Krotenheerdt, O. and Veit., S. Zur Theorie massiver Knoten, Beitr. Algebra und Geometrie, **5** (1976), 61–74.

[35] Kylander O.S., Kylander K., GIMP - The Official Handbook, Coriolis Value/November (1999).

[36] Laurie B., Annealing Ideal Knots and Links: Methods and Pitfalls, Chapter 3 of [67].

[37] R.A. Litherland, J. Simon, O. Durumeric, E. Rawdon, Thickness of Knots, *Topology and its Applications* **91(3)** (1999), 233–244.

[38] A. Maritan, C. Micheletti, A. Trovato and J. R. Banavar, Optimal shapes of compact strings, *Nature* **406**(6793) (2000) 287–290.

[39] Maddocks J.H., Keller J.B., Ropes in equilibrium, SIAM J.Appl. Math. **47**, 6 (1987) 1185-1200

[40] Meade, T. and Arima, S., Maya 6: The Complete Reference, McGraw-Hill, Inc., New York, NY, USA, (2004).

[41] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller and E. Teller *Equation of State Calculations by Fast Computing Machines* J. Chem. Phys. 21, 1087 (1953).

[42] Murdock, K.L., 3D Studio MAX R3 Bible, Wiley (January 2000).

[43] Alexander Nabutovsky, Non-recursive functions, knots "with thick ropes", and self-clenching "thick" hyperspheres, Comm. Pure Appl. Math., 48(4):381–428, (1995).

[44] Neider, J. and Davis, T., OpenGL Programming Guide: The Official Guide to Learning OpenGL, Release 1, Addison-Wesley Longman Publishing Co., Inc.,Boston, MA, USA, (1993).

[45] Qt – Cross-platform application and UI framework, qt.nokia.com, (21 september 2009).

[46] O'Hara J., Energy of Knots, Chapter 16 of [67].

[47] O'Hara J., Family of energy functionals of knots, Topology Appl., 48(2):147–161, (1992).

[48] Open Inventor Architecture Group, Open inventor c++ reference manual, Addison Wesley, (1994).

[49] Osher, S. and Sethian, J.A., Fronts Propagating with Curvature Dependent Speed: Algorithms Based on Hamilton-Jacobi Formulations, J. Comp. Phys, **79** (1988) 12–49.

[50] O'Sullivan, B., Mercurial: The Definitive Guide, O'Reilly Media, (2009).

[51] Persistence of Vision Raytracer Pty. Ltd., `www.povray.org`, (21 september 2009).

[52] Piegl, Les and Tiller, Wayne, The NURBS book (2nd ed.), Springer-Verlag, New York, (1997).

[53] Pieranski P., *In Search of Ideal Knots*, Chapter 2 of [67].

[54] Pieranski, P.; Przybyl, S. *In Search of the Ideal Trefoil Knot*, in *Physical Knots*, Eds. J. Calvo, K. Millett, E. Rawdon, and A. Stasiak, World Scientific (2001), 153–162.

[55] Luìs F. Portugal F.L., Júdice J.J., and Vicente L.N., A comparison of block pivoting and interior-point algorithms for linear least squares problems with nonnegative variables, Math. Comp., **63**#208 (1994), 625–643.

[56] Rawdon, E. J., Can computers discover ideal knots?, *Experimental Mathematics*, **12**#3 (2003), 287–302.

[57] Reiter P., Knotenenergien, Diploma thesis, Math. Inst. Univ. Bonn, (2004).

[58] Reiter, P. All regular curves in a C^1-neighborhood are ambient isotopic. Preprint Nr. 4 Institut f. Mathematik, RWTH Aachen University (2005), `http://www.instmath.rwth-aachen.de/preprints`

[59] Rolfsen, D., Knots and Links Publish or Perish, Inc. Houston, Texas (1976).

[60] G. Scharein, R.G., Interactive Topological Drawing, PhD Thesis, Department of Computer Science, The University of British Columbia, (1998).

[61] F. Schuricht and H. von der Mosel, Global curvature for rectifiable loops, *Mathematische Zeitschrift*, **243** (2003) 37-77.

[62] F. Schuricht and H. von der Mosel, Characterization of ideal knots, *Calc. Var.*, **19** (2004) 281–305.

[63] Shapeways passionate about creating, `www.shapeways.com`, (14 december 2009).

[64] T. J. Sharrock, Biarcs in three dimensions, in The Mathematics of Surfaces II, Ed. R. R. Martin, Oxford Univ. Press, New York, (1987) 395-411.

[65] Shreiner, D., OpenGL Reference Manual: The Official Reference Document to OpenGL, Version 1.2, Addison-Wesley Longman Publishing Co., Inc.,Boston, MA, USA, (1999).

[66] Smutny, J. Global radii of curvature and the biarc approximation of spaces curves: In pursuit of ideal knot shapes, PhD. Thesis No. 2981, EPF Lausanne (2004), `http://library.epfl.ch/theses/?display=detail\&nr=2981`.

[67] A. Stasiak, V. Katritch and L. H. Kauffman (Eds), Ideal knots, Ser. Knots Everything **19**, World Sci. Publishing, River Edge, NJ, (1998).

[68] E. M. Stein and G. Weiss, Introduction to Fourier Analysis on Euclidean Spaces, Princeton University Press, (1971).

[69] Trautwein, A.K., An introduction to harmonic knots, Chapter 18 of [67].

[70] Upstill, S., RenderMan Companion: A Programmer's Guide to Realistic Computer Graphics, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, (1989).

[71] Wang, W. and Joe, B., Classification and properties of space biarcs, in SPIE: Curves and surfaces in computer vision and graphics III, Boston, **1830** (1992), 184-195.

[72] Weeks, J., The Shape of Space, M. Dekker, New York (1985).

[73] Wernecke J., The Inventor Mentor: Programming Object-Oriented 3d Graphics with Open Inventor, Release 2, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, (1993)

[74] Wright, W. D., The measurement of colour, 4th Ed., Hilger, London (1969).