Computational Methods for Riemann Surfaces and Helicoids with Handles

vorgelegt von
Diplom-Mathematiker
Markus Schmies
Berlin

Fakultät II - Mathematik und Naturwissenschaften
der Technischen Universität Berlin
zur Erlangung des akademischen Grades
Doktor der Naturwissenschaften
- Dr. rer. nat. -

genehmigte Dissertation

Promotionsausschuss:

Vorsitzender:	Prof. Dr. Ruedi Seiler
Gutachter/Berichter:	Prof. Dr. Alexander I. Bobenko
	Prof. Dr. Ulrich Pinkall

Tag der wissenschaftlichen Aussprache: 16. September 2005

Berlin 2005
D 83

Acknowledgments

There are some people that I would like to thank for their contribution to this thesis.

First of all I want to express my gratitude and my appreciation to my advisor Prof. Alexander I. Bobenko, who introduced me to the field of Riemann surfaces and integrable systems, the theoretical foundations of this thesis. I am also very grateful to co-advisor Prof. Ulrich Pinkall for many deep insights, especially about myself. I would like to thank Prof. John Sullivan for many fruitful discussions during the last 18 months. I am especially grateful to Peter Brinkmann for editing and proofreading the first four chapters. Furthermore, I want to thank Charles Gunn for helping me to overcome many problems I had in understanding the English language. I also like to thank Uli Heller for proofreading the final chapter, and for his contribution to the jtem-project, which provides the basis for the numerics presented here. A special word of thanks goes to Holger Pietsch for his inspiring collaboration in many software projects, especially in the Oorange-project which enabled many of my numerical experiments. I thank Tim Hoffman for providing the software-render backend for the jReality-viewer which I used to render all 3d-images. I would also like to express my gratitude to my colleagues Christoph Bohle, Paul Peters, and Boris Springborn for their generous support. Last but not least I like to thank my friends and family for their love, support, and patience.

Abstract

We develop methods for the numerical treatment of Riemann surfaces and apply them to the problem of computing and visualizing helicoids with handles.

Using Schottky uniformization to represent Riemann surfaces, we explicitly express differentials and integrals of the Riemann surfaces as series, and we find a priori criteria for their convergence that can be evaluated efficiently. We also develop algorithms for the evaluation of these series within prescribed error bounds and study their efficiency.

Riemann theta functions are a basic technical tool for the description of finite gap solutions of integrable systems. We give a survey of computational methods for their evaluation and present new implementations as well as new algorithms and compare them with current methods.

As an application we perform numerical experiments resulting in images of helicoids with up to six handles. Our numerical experiments strongly suggest that there exists exactly one embedded helicoid of a given genus. We also obtain immersed examples of genus-one helicoids with less symmetry than previously known examples.

We close with a description of the software libraries that have been created in the context of this thesis.

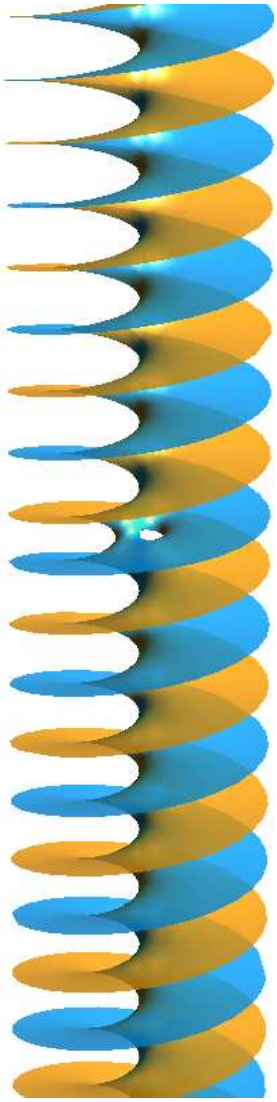
Contents

CHAPTER 1

Introduction and Summary

1.1. Introduction

The purpose of this thesis is a numerical investigation of helicoids with handles. Helicoids with handles are complete minimal surfaces of finite topology, i.e., they have finite genus and finitely many ends. In fact, helicoids have just one end, which is asymptotic to the simple helicoid.



Among all minimal immersions the embedded examples are the most interesting. Embedded minimal surface of infinite topology abound. In the 19th century Scherk discovered his famous families of singly and doubly periodic embedded examples. Since then, many more examples of infinite topology have been constructed, but until 1984 the only known embedded examples of finite topology were the plane, the catenoid (Euler, c. 1744), and the helicoid (Meusnier, 1776).

In 1984 Costa [**Cos84**] wrote down the Weierstrass data for his famous minimal torus having one planar and two catenoidal ends that are all disjoint and separately embedded. The following year David Hoffman and William Meeks, working with Jim Hoffman, were able to generate computer pictures of the Costa surface indicating that it is embedded, which they proved soon afterward [**HM85**].

The family of complete minimal surfaces has an important subclass of surfaces with finite total curvature given by all surfaces M for which the integral of the absolute Gauss curvature $\int_M |K|\, dA$ is finite. Costa's surface has finite total curvature, and many more examples of finite total curvature have been generated since. The minimal surfaces that we are concerned with have infinite

total curvature because the helicoidal end has infinite total curvature.

In 1993, Hoffman, Karcher, and Wei added a single handle to the helicoid, creating the first infinite total curvature example of finite positive genus [**HWK93**]. This helicoid with one handle ($\mathcal{H}e_1$) inspired the theory of infinite total curvature minimal surfaces much like Costa's surface did the finite curvature case. Once again, plots of the surface strongly suggest that it is embedded, and Weber, Hoffmann, and Wolf recently proved this by exposing $\mathcal{H}e_1$ as the limit of a family of embedded minimal surfaces that are invariant under screw-motion [**WHW03**].

In 1997 Collin proved a generalization of Nitsche's Conjecture that implies the following result.

THEOREM 1.1.1. [**Col97**] *A properly embedded minimal surface of finite topology and infinite total curvature has at most one end.*

This result is intuitively clear for surfaces with helicoidal ends. It complements a theorem of Osserman's that states that finite total curvature implies finite genus [**Oss64, Oss86**].

Since all known examples of properly embedded infinite total curvature are asymptotic to the helicoid, it is natural to ask whether this is true for all such surfaces. Increasingly more general results toward this end appeared in Hoffman and McCuan [**HM03**] and Hauswirth, Perez and Romon [**HPR01**]. Finally Meeks and Rosenberg introduced the important class of Weierstrass data of finite type (see (1.1.1) below) and established the following remarkable theorem in 2001.

THEOREM 1.1.2. [**MR01**] *If M is a properly embedded nonplanar minimal surface with finite genus g and one end, then M is asymptotic to a helicoid and of finite type.*

In 1997 Bobenko gave explicit formulas for global conformal parameterizations of all infinite total curvature minimal immersions of finite topology with one helicoidal end of finite type using the spinor Weierstrass representation [**Bob98**]. Because of Theorem 1.1.2 his formulas describe in particular all properly embedded nonplanar minimal surface of finite genus.

The aim of this thesis is a systematic numerical exploration of this class of minimal immersions using analytic methods of the theory of integrable systems as described in [**Bob98**]. We hope that the insights gained from our computational analysis will contribute to the theory of minimal surface and inspire new results for higher genus infinite total curvature surfaces, like earlier experiments did for the genus-one case and the finite total curvature case.

The numerics we developed for this task is not specific to this problem, but applicable to the computation of all finite genus minimal surfaces of finite type. Our starting point is the spinor Weierstrass representation for minimal surfaces, which first appeared in notes of Dennis P. Sullivan and later was reinvented in [**Bob94**] and [**KS93**].

THEOREM 1.1.3. [**Bob94, KS93**] *Let a,b be two holomorphic sections (spinors) of a spin bundle S over a Riemann surface $\mathcal{R}$ that do not vanish simultaneously. Then*

$$F(P) = Re \int_Q^P \left(-a^2 + b^2, i\left(a^2 + b^2\right), 2ab\right)$$

defines a conformal minimal immersion $F : \mathcal{R} \to \mathbb{R}^3$. All conformal minimal immersions are described this way.

The parameterization F is of finite type if $\mathcal{R}$ is a compact Riemann surface C of genus g with a finite number of punctures $\{P_0, \ldots, P_n\}$, and the 1-Forms

$$\frac{b}{a} d\frac{a}{b} \quad \text{and} \quad ab \tag{1.1.1}$$

extend meromorphically to the punctures. This is a proper generalization of meromorphic spinors a, b. The latter class has been shown by Ossermann to describe all finite total curvature surfaces [**Oss64**].

To specify F one needs to prescribe the compact Riemann surface C, the punctures $\{P_0, \ldots, P_n\}$, and the spin structure S. Additional data describes the specific properties of the spinors. For helicoids of finite genus the spinors have special essential singularity at the puncture (end) and are uniquely determined by an element v of the tangent space at the puncture P_0 and by the spin structure S. The quadruple (C, P_0, v, S) depends on $6g - 2$ real parameters. A priori, F given by such spinors is multivalued, and it only defines a closed surface if the Weierstrass integrals vanish along all $2g$ cycles of a holonomy basis of $\Re$. This yields $6g$ constraints, which shows that the period problem is not generically solvable, i.e., there exists no helicoid with generic conformal structure. An analysis of the $\mathcal{H}e_1$ shows that its symmetries can be exploited to reduce the number of constraints, but only one of the symmetries – a 180° rotation about a line orthogonal to the surface – actually reduces this number to match the number of free parameters. The same phenomenon can be observed in the higher genus case also. This motivates the following conjecture:

CONJECTURE 1.1.4. [**Bob98**] *Any immersed minimal surface of finite topology with one helicoidal end is invariant under a* 180° *rotation about a line orthogonal to the surface (normal symmetry).*

An immediate consequence of this conjecture and Theorem 1.1.2 is that any properly embedded minimal surface of finite genus is invariant under a 180° rotation about a line orthogonal to the surface. The class of surfaces satisfying this property contains the surfaces that have been under consideration so far: the simple helicoid, $\mathcal{H}e_1$, and the experimental genus-two and -three examples of Traizet (see [**WHW03, HW02**]).

Efficient numerical methods for generic Riemann surfaces of higher genus are the key ingredient of our computational analysis. Until recently general numerical tools were limited to elliptic functions. Heil developed algorithms for hyperelliptic

Riemann surfaces [**Hei95**], and Deconinck and Hoeij offer Maple packages for computations of algebraic curves [**DvH01**].

Common numerical methods represent Riemann surface through algebraic curves. This approach has serious disadvantages: if one is interested in the corresponding Riemann surface only one needs to factorize algebraic curves with respect to birational maps. This complicates the corresponding parameterization of Riemann surfaces. Moreover, a representation of a Riemann surfaces as a ramified multisheet covering complicates the description of the homology and integration paths, which yields complex algorithms. Schottky uniformization is an attractive alternative to describing Riemann surfaces in terms of algebraic curves.

A Schottky group is a free, finitely generated, discontinuous group G that is purely loxodromic, i.e., a Schottky group of rank N can always be generated by N loxodromic transformations $\sigma_1, \ldots, \sigma_N$. A classical theorem states that for any Riemann surface $\mathcal{R}$ there exists a Schottky group G such that $\mathcal{R}$ is conformally equivalent to the quotient Ω/G, where Ω denotes the set of discontinuity of G [**For29**]. The number N of generators of the Schottky group equals the genus of the associated Riemann surface. A loxodromic transformation σ_i is determined by its fixed points A_i and B_i and the loxodromic factor μ_i. In other words, a Schottky group can be parameterized by the data

$$S = \{A_1, B_1, \mu_1, \ldots, A_N, B_N, \mu_N\},$$

which are fixed points and loxodromic factors of their generators $\sigma_1, \ldots, \sigma_N$. This provides a canonical way of parameterizing Riemann surfaces.

Functions and differentials of the Riemann surface Ω/G are automorphic on Ω. This yields explicit representations for functions, differentials, and integrals in terms of Poincare theta series. For example, normalized differentials of the first kind can be expressed as (-2)-dimensional Poincare theta series:

$$\omega_n(z) = \sum_{\sigma \in G_n \backslash G} \left(\frac{1}{\sigma(z) - B_n} - \frac{1}{\sigma(z) - A_n} \right) d\sigma(z).$$

These series do not always converge. Some sufficient convergence criteria have been developed and numerically tested (see [**BBE$^+$94**]). We developed sufficient a priori convergence criteria that can be evaluated efficiently as well as algorithms for the evaluation of these series in prescribed error bounds. Finding general convergence criteria remains an interesting open problem.

The holomorphic spinors are given in terms of Riemann theta functions, which are the basic technical tool for the description of finite gap solutions of integrable systems. The Riemann theta function is a complex-valued function of g complex variables and is defined by

$$\theta(z\,|B) = \sum_{n \in \mathbb{Z}^g} e^{\frac{1}{2}\langle n, B \cdot n \rangle + \langle z, n \rangle},$$

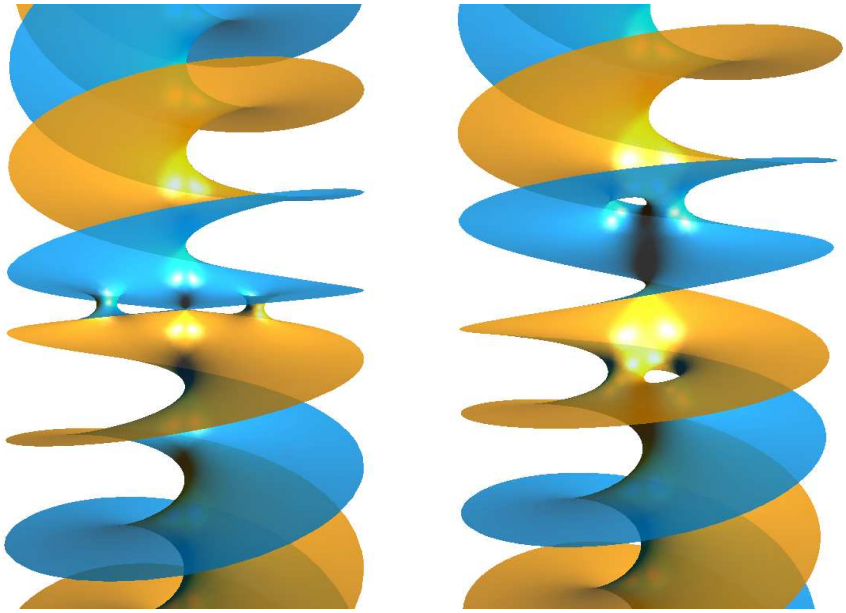
where $z \in \mathbb{C}^g$ and B is a symmetric g-dimensional matrix with strictly negative definite real part. In 1995, M. Heil developed algorithms for evaluating Riemann theta function in higher dimensions [**Hei95**] and established the foundations of current methods [**DHB$^+$04**]. Although the series of the Riemann theta function converges exponentially fast, its evaluation usually accounts for a large portion of the total computational performance because in higher dimensions the sum may consists of tens of thousands of terms. We present refinements of existing methods that accelerate the evaluation process for higher genus helicoids by a factor of ten. This improvement shifts the computational bottleneck from the evaluation of Riemann theta functions to the evaluation of the Schottky series and enables the treatment of examples beyond genus 10.

Our numerical tools work with Schottky groups that have a canonical classical fundamental domain $F \subset \Omega \subset \mathbb{C}$ with exactly N pairs of boundary circles. Each pair is determined by one of the generators σ_i. Numerical integration in the fundamental domain F is considerably simpler than integration in a branched covering because it obviates the need to keep track of the current sheet. The fundamental domain F yields a canonical homology basis. This is important for the final step of the numerical analysis, i.e., solving the period problem.

Finding parameters satisfying the periodicity conditions is theoretically and numerically a challenging task. For one free real parameter the period problem can be solved by an intermediate-value argument, which is used for most known examples. Two-parameter problems are much more difficult and have only been solved sporadically, e.g., for $\mathcal{H}e_1$. For higher dimensions direct methods for solving the period problem have not been successful as of yet. Wolf and Weber recently developed a completely different method using techniques from Teichmüller theory, which is not currently applicable to our situation [**WW02, Web**]. For the time being, we are left with computational methods.

Root finding is a standard task in numerics and there exist a great variety of methods, but in a high dimensional space it remains the proverbial search for a needle in a haystack. Even if the numerical root finder succeeds, how do we know that its result is really a root and not just a numerical artifact? Without a formal proof we never can be sure, but this does not mean that we have to rely on images showing convincingly closed surfaces. Our numerical analysis allows us to distinguish between artifacts, like local minima, and roots, and it also provides estimates of numerical errors.

In our constructions, we only consider surfaces that possess the normal symmetry (Conjecture 1.1.4). Our systems of equations would be overdetermined without the conjectured holomorphic involution $\pi : C \to C$ of the Riemann surface. There are two kinds of extremal examples related to the two-sheeted ramified covering $C \to C_0 = C/\pi$ associated to this involution. One extreme consists of hyperelliptic examples for which the genus of C_0 is zero ($g_0 = 0$). Here one expects the handles to be located along the normal line of symmetry. The other extreme contains coverings with only two branch points. In this case, the



Two minimal immersions with helicoidal end with associated Riemann surface C of genus 2. In the genus 2 case, there exist only two kinds of two-sheeted covers $C \to C_0$. On the left, C_0 is the complex plane and the handles are horizontally aligned along the normal line of symmetry. On the right, C_0 is a torus and the handles are displaced vertically. Although the periods of both immersions appear to be closed, this is only true for the helicoid on the left, the $\mathcal{H}e_2$. Numerical analysis suggests that there are no closed examples with vertically displaced handles.

genus of C equals twice the genus of C_0 ($g = 2g_0$). The normal line of symmetry intersects the surface only at the two branch points and one expects vertically displaced handles. Our numerical analysis only produced hyperelliptic examples, and it suggests that the other cases are impossible.

Up to genus 10 we have found numerical evidence of hyperelliptic examples; up to genus 6, we were able to give error estimates. All these examples have additional symmetries: 180° rotations about a vertical and a horizontal line contained in the helicoid. Some of the results presented here took weeks of computation on a cluster of twenty-five 2GHz PCs. Using this combination of efficient numerical tools and raw computational power we were able to scan the parameter space for genus one and two, eventually finding a unique solution of each genus. This corroborates a conjecture of Weber, Hoffmann, and Wolf stating that for genus-one, there is only one properly embedded minimal surface of infinite total curvature [**WHW03**]. All this evidence supports the following generalization of their conjecture.

CONJECTURE 1.1.5. *For each genus g there exists a unique properly embedded minimal surface M with infinite total curvature. M is hyperelliptic and contains a vertical and a horizontal axis and is normal to a third. All three axes intersect orthogonally in one point. M is invariant under* 180° *rotations about any of the three axes.*

We note that this conjecture is a stronger version of a conjecture of Meeks and Rosenberg that states that the moduli space of properly embedded one-ended minimal surfaces of genus g consists of a finite number of points [**MR01**].

The situation is different if we consider general minimal immersions. For the genus-one case we found sequences of several hundred, non-embedded rhombic tori with the same symmetries as $\mathcal{H}e_1$. We also found a couple of examples of non-rhombic conformal type. In particular these immersions of genus-one with a helicoidal end do not have all the symmetries of $\mathcal{H}e_1$. It appears that embeddedness has far-reaching consequences, for symmetries as well as uniqueness.

1.2. Summary

In Chapter 2 we introduce numerical methods for Schottky uniformization. After a short introduction to this uniformization theory we prove a priori criteria for the convergence of series that describe differentials and integrals on Riemann surfaces in this uniformization picture. We present algorithms allowing an efficient evaluation of these series within prescribed error bounds and study their efficiency by means of selected examples.

In Chapter 3 we give a survey of computational methods for evaluating Riemann theta functions, and we present refinements of existing methods as well as new algorithms that significantly improve upon current methods.

In Chapter 4 we apply our numerical methods to the problem of computing and visualizing helicoids with handles. After introducing the theory of helicoids with handles we express the problem in terms of Schottky uniformization. We discuss those numerical methods that have not been the subject of previous chapters, but are necessary for the computation of helicoids. In particular we focus on different integration methods and root finding algorithms for functions of several variables and analyze the stability of solutions. We report on our numerical experiments and their results including pictures and data tables of the surfaces we found.

Chapter 5 deals with the software implementation of our numerical methods. This code was the origin of the core libraries of the *jtem* project. We give an introduction to these libraries including example implementations for the most important numerical tasks.

CHAPTER 2

Numerical Methods for Schottky Uniformization

Helicoids with handles are defined by Weierstrass data on Riemann surfaces. For computational purposes we know two suitable representations for Riemann surfaces: algebraic curves and Schottky groups. Historically, algebraic cuves can be considered the origin of Riemann surfaces and may therefore be a natrual representation. Nevertheless, the representation by Schottky groups, also known as Schottky uniformization, is often more useful.

The space of solutions of a given problem is usually (atleast partially) parametrized by Riemann surfaces. Thus if the solution is supposed to meet additional constrains, e.g. periodicity conditions, it is crucial that the moduli space of Riemann surfaces associated to the problem can be explicitly parametrized. All Schottky groups representing a Riemann surface of genus g can be described by a set of $3g$ complex variables, the Schottky data, which encodes the generators of the group.

Schottky uniformization also provides generic closed formulas for differentials and integrals of different kinds, including period matrices, in terms of infinite series[1]. Unfortunately these series do not allways converge and, even worse, there are no general convergence criteria. The aim of this chapter is to develop sufficient convergence critera for the various Schottky series as well as efficient evaluation algorithms for those series when practicable.

2.1. Introduction to Schottky Uniformization

Loxodromic Transformations.

Definition 2.1.1. Let

$$M_\sigma = \begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix} \in PSL(2, \mathbb{C})$$

be the matrix representation of the linear fractional transformation $\sigma(z) = \frac{\alpha z+\beta}{\gamma z+\delta}$ and $\sqrt{\mu}, 1/\sqrt{\mu}$ its eigenvalues. σ is called loxodromic iff $|\mu| \neq 1$.

Transformations with real trace are classified according to the following definition:

[1]The normalized differentials of first kind are given as (-2)-dimensional Poincare theta series.

DEFINITION. A Möbius transformation σ represented by $M_\sigma \in PSL(2,\mathbb{C})$ with $\kappa = \mathrm{Tr}M_\sigma \in \mathbb{R}$ is called *elliptic, parabolic*, or *hyperbolic* respectively if $|\kappa| < 2$, $|\kappa| = 2$,or $|\kappa| > 2$.

Usually loxodromic transformations are defined to be those with nonreal trace. However, Definition 2.1.1 only requires that they be nonelliptic and nonparabolic, therefore they may be hyperbolic.

A nonparabolic transformation σ has exactly two fixed points A and B. If its fixed points are finite, then:

$$\text{(2.1.1)}\quad \begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix} = \frac{1}{A-B} \begin{pmatrix} A\sqrt{\mu} - B\frac{1}{\sqrt{\mu}} & AB\left(\frac{1}{\sqrt{\mu}} - \sqrt{\mu}\right) \\ \sqrt{\mu} - \frac{1}{\sqrt{\mu}} & A\frac{1}{\sqrt{\mu}} - B\sqrt{\mu} \end{pmatrix} \in PSL(2,\mathbb{C})$$

is its matrix representation with eigenvalues $\sqrt{\mu}$ and $1/\sqrt{\mu}$. The circle C of radius $1/|\gamma|$ centered at

$$\sigma^{-1}(\infty) = -\frac{\delta}{\gamma} = \frac{A\frac{1}{\sqrt{\mu}} - B\sqrt{\mu}}{\frac{1}{\sqrt{\mu}} - \sqrt{\mu}}$$

is mapped onto a circle C' of the same radius centered at $\sigma(\infty) = \alpha/\gamma$. C and C' are called the *isometric* circles of the transformation σ.

Schottky Uniformization. A free, finitely generated, discontinuous, purely loxodromic group G is called Schottky group. A more explicit, but equivalent [**Mas67**], approach to characterize Schottky groups is the following:

DEFINITION 2.1.2. Let $C_1, C_1', \ldots, C_N, C_N'$ be a set of $2N$ mutually disjoint Jordan curves on $\mathbb{C}$ forming the boundary of a $2N$-connected domain F (Figure 2.1.1). The loxodromic transformations σ_n, $n = 1, \ldots, N$, defined by

$$\frac{\sigma_n(z) - B_n}{\sigma_n(z) - A_n} = \mu\frac{z - B_n}{z - A_n}, \qquad |\mu| < 1$$

transform the outside of the boundary curves C_n onto the inside of the boundary curves C_n'. Therefore their fixed points A_n and B_n lie inside C_n respectively C_n'. They generate a group G that is called a *Schottky group*. Such a representation is called *classical* if all the boundary curves are circles.

Denote by Ω the set of discontinuity of the Schottky group G, generated by $\sigma_1, \ldots, \sigma_N$, then Ω/G is a Riemann surface $\mathcal{R}$ of genus N [**For29**]. If we choose N homologically independent simple disjoint loops $v_1 \ldots, v_N$ on $\mathcal{R}$, and cut $\mathcal{R}$ along these loops, then we obtain a plane region that can be mapped conformally to the fundamental domain F of G. The loops v_k are exactly mapped onto the curves C_k and C_k'. Thus the boundary of F is the image of the loops $v_1 \ldots, v_N$.

Schottky uniformication states that all Riemann surfaces can be represented in this way [**For29**]. However, it is unknown whether all Riemann surfaces can be uniformized using classical Schottky groups. The task of explicitly determining

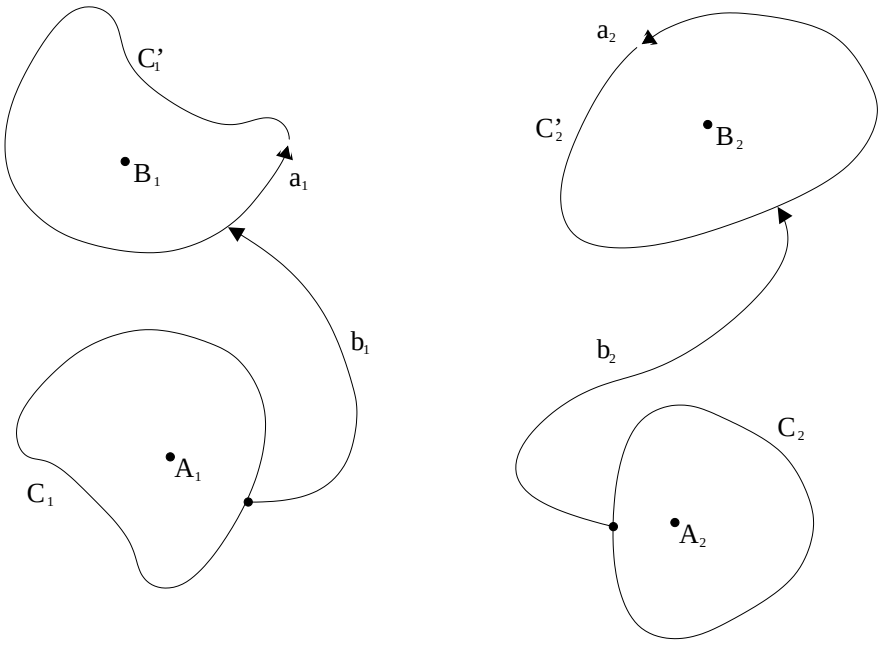


FIGURE 2.1.1. The fundamental domain F. a_n coincides with the positively oriented C_n' ; b_n runs on F between the points $z_n \in C_n$ and $\sigma_n z_n \in C_n'$; b-cycles do not mutually intersect. a_n and b_n form a Schottky generic basis of cycles.

the set $S = \{A_1, B_1, \mu_1, \ldots, A_N, B_N, \mu_N\}$ of uniformization parameters representing a given Riemann surface is also unsolved. Nevertheless, we can use the data to parameterize the moduli space of compact Riemann surfaces of genus N. The complex dimension of the moduli space of compact Riemann surfaces is 1 for genus $N = 1$ and $3N - 3$ for genus $N \geq 2$, hence the S set of parameters is dependent: Conjugating the Schottky group G with a linear fractional transformation τ yields to a Riemann surface $\tau(\Omega)/\tau G\tau^{-1}$ that is mapped conformally by τ^{-1}onto Ω/G. The Schottky data transforms to $\{\tau \cdot A_1, \tau \cdot B_1, \mu_1, \ldots, \tau \cdot A_N, \tau \cdot B_N, \mu_N\}$, which reduces the number of free complex parameter to the dimension of the moduli space of compact Riemann surfaces.

To compute the period matrix we need to fix a canonical basis of cycles.

DEFINITION 2.1.3. Let C_n, C_n', σ_n, G and F be as in Definition 2.1.2. A canonical Basis of cycles is called *Schottky generic* if the a-cycles coincide with positively oriented C_n', and the b-cycles b_n connect $z_n \in C_n$ with $\sigma_n(z_n) \in C_n'$ in F such that they do not mutually intersect.

Figure 2.1.1 shows a Schottky generic basis of cycles. Its definition does not uniquely determine the b-cycles and we have to be aware of this when we calculate period matrices.

Let G be the free group generated by $\sigma_1, \ldots, \sigma_N \in G$. Each $\sigma \in G$ has a unique representation

$$\sigma = \sigma_{i_1}^{j_1} \ldots \sigma_{i_k}^{j_k}$$

with $j_l \in \{-1, 1\}, i_l \in \{1, \ldots, N\}$, and $j_l i_l \neq -j_{l+1} i_{j+1}$. Denote by

$$[\sigma] = [j_1 i_1, \ldots, j_k i_k]$$

the word of σ associated to the generators $\sigma_1, \ldots, \sigma_N$ and by $|\sigma| = k$ its length. Denote further by G_n the subgroup generated by σ_n and define its cosets as:

$$\begin{aligned} G/G_n &= \{\sigma \,|[\sigma] = [\ldots, r], |r| \neq n\} \\ G_n \setminus G &= \{\sigma \,|[\sigma] = [s, \ldots], |s| \neq n\} \\ G_m \setminus G/G_n &= \{\sigma \,|[\sigma] = [s, \ldots r], |s| \neq m \wedge |r| \neq n\}. \end{aligned}$$

The following lemma is one of the major ingredients of the numerical solution of our problem. It was adapted in [**BBE$^+$94**] from the classical papers [**Bak97, Bur92**].

LEMMA 2.1.4. *Let $\sigma_n, A_n, B_n, \mu_n$ and G be as in Definition 2.1.2. If the series*

$$\omega_n(z) = \sum_{\sigma \in G/G_n} \left(\frac{1}{z - \sigma(B_n)} - \frac{1}{z - \sigma(A_n)} \right) dz \tag{2.1.2}$$

are absolutely convergent, then they define holomorphic differentials normalized over a Schottky generic basis. Their integrals are then given by

$$\Omega_n(z) = \int_\infty^z \omega_n = \sum_{\sigma \in G/G_n} \log \frac{z - \sigma(B_n)}{z - \sigma(A_n)} \tag{2.1.3}$$

and the period matrix by

$$B_{nm} = \delta_{nm} \log \mu_n + \sum_{\sigma \in G_m \setminus G/G_n, \sigma \neq id} \log \{B_m, A_m, \sigma(B_n), \sigma(A_n)\}, \tag{2.1.4}$$

where the curly brackets indicate the cross-ratio

$$\{a, b, c, d\} = \frac{a - c}{a - d} \cdot \frac{b - d}{b - c}.$$

The Schottky series (2.1.2), (5.6.2), and (5.6.3) do not necessarily converge in the general case. In fact it is unknown whether an absolute convergent series exists for a given Riemann surface. Nevertheless, in certain situations absolute convergence can be guaranteed.

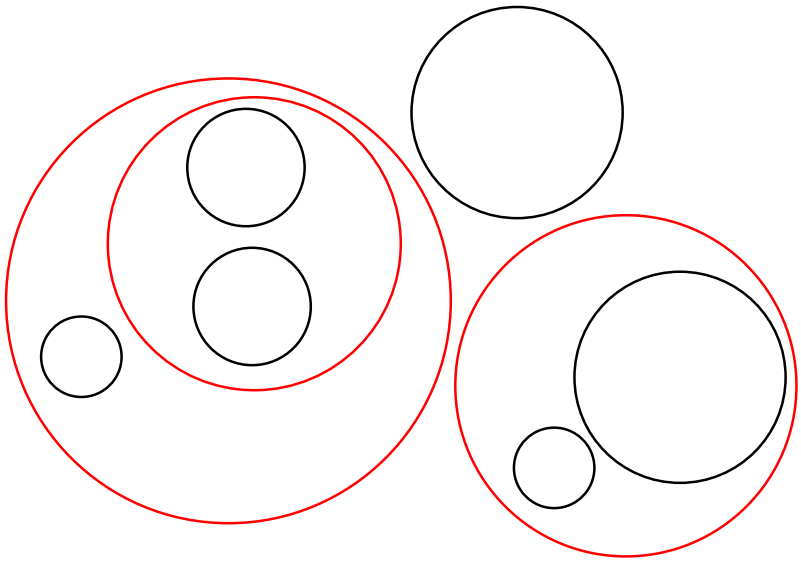


FIGURE 2.2.1. Isometric circles of a decomposable classical Schottky group (black) with a system of decomposing circles (red).

2.2. Convergence of Schottky Series

Before we analyze Schottky series, we want to introduce a very geometric, and therefore quite elegant, criterion for their convergence.

Let $C_n, C_n', \sigma_n, N, G$ and F be as in Definition 2.1.2. A classical Schottky group G is called *circle decomposable* [**BBE+94**] if there exists a system of circles $L_1, \ldots, L_{2N-3}$ satisfying the following conditions:

(1) The circles $L_1, \ldots, L_{2N-3}, C_1, C_1', \ldots, C_N, C_N'$ are mutually disjoint.
(2) The circles $L_1, \ldots, L_{2N-3}$ divide F into $2N-2$ regions $T_1, \ldots, T_{2N-2}$.
(3) Each T_i has exactly three boundary circles.

THEOREM 2.2.1. [**FK65**] *For circle decomposable Schottky groups the Schottky series (2.1.2), (5.6.2), and (5.6.3) are absolutely convergent.*

Figure 2.2.1 shows an example of a circle decomposable Schottky group. It is not trivial to determine the system of $2N-3$ circles decomposing the fundamental domain by an algorithm. This makes it hard to use this concept to estimate the numerical error, but it guarantees the convergence for a large number of surfaces.

We seek criteria that allow us to control the errors in terms of the Schottky uniformization data alone, and guarantees convergence for sufficiently small isometric circles C and C'. Such estimates have been given first by Burnside [**Bur92**], see also [**FK65, Bak97**]. However, these authors considered only

principle mathematical questions about the convergence. They used rather primitive estimates to show that for sufficiently small cirlces the series converge. For numerics (error estimates) we obtain much more improved versions of these classical results.

We call a classical Schottky group *iso-classical* if all the boundary circles are isometric. Let C_n (C_n') be isometric circles of the iso-classical Schottky group G as in Definition 2.1.2 and denote C_n (C_n') by C_n^1 (C_n^{-1}) and its open interior by D_n or D_n^1 (D_n' or D_n^{-1}). Note that

$$\sigma_m^i\left(\mathbb{C}\setminus\overline{D_m^i}\right)=D_m^{-i}\,,$$

because Schottky groups map the outside of C_n onto the inside of C_n'. This yields

LEMMA 2.2.2. *For $\sigma_{(l)}=\sigma_{m_1}^{i_1}\cdot\ldots\cdot\sigma_{m_l}^{i_l}\in G$ we have*

$$\sigma_{(l+1)}\left(\mathbb{C}\setminus\overline{D_{m_{l+1}}^{i_{l+1}}}\right)=\sigma_{(l)}\left(D_{m_{l+1}}^{-i_{l+1}}\right)\subset\sigma_{(l)}\left(\mathbb{C}\setminus\overline{D_{m_l}^{i_l}}\right)\,.$$

For σ with $[\sigma]=[im,\ldots]$ we have $\sigma(F)\subseteq D_m^{-i}$ which implies that for all $\sigma\in G_n\setminus G$ holds $\sigma(F)\cap D_n^{\pm1}=\emptyset$, and $dist(\sigma(F),F)>0$ for all $\sigma\in G,|\sigma|>1$. We now define constants measuring the displacement of the fundamental domain F under an element of the Schottky group. Let $\sigma\in G$ and define

$$\begin{array}{rcl}k(P;\sigma) & = & \min_{z\in F}|P-\sigma(z)|\\ K(P;\sigma) & = & \max_{z\in F}|P-\sigma(z)|\,.\end{array}\tag{2.2.1}$$

Let $\sigma_{(l)}=\sigma_{m_1}^{i_1}\cdot\ldots\cdot\sigma_{m_l}^{i_l}\in G$ and $P\notin D_{m_1}^{-i_1}$. We conclude that

$$\begin{array}{ccccccccc}0 & \leq & k\left(P;\sigma_{(1)}\right) & < & k\left(P;\sigma_{(2)}\right) & <\ldots< & \lim_{l\to\infty} & k\left(P;\sigma_{(l)}\right)\\ \infty & > & K\left(P;\sigma_{(1)}\right) & > & K\left(P;\sigma_{(2)}\right) & >\ldots> & \lim_{l\to\infty} & K\left(P;\sigma_{(l)}\right)\end{array}\,.\tag{2.2.2}$$

For iso-classical Schottky groups, the limits coincide, i.e.,

$$\lim_{l\to\infty}k\left(P;\sigma_{(l)}\right)=\lim_{l\to\infty}K\left(P;\sigma_{(l)}\right)\,,\tag{2.2.3}$$

which follows from the next lemma.

LEMMA 2.2.3. *Let G be a iso-classical Schottky group and $\sigma_{(l)}=\sigma_{m_1}^{i_1}\cdot\ldots\cdot\sigma_{m_l}^{i_l}\in G$ be a infinite sequence. Then the depending sequence of sets $\sigma_{(l)}(F)$ contracts to a point.*

PROOF. We will show that

$$\left|\sigma_{(l)}^{\sigma}(z)\right|=\left|\left(\sigma_{m_1}^{i_1}\right)'\left(\sigma_{m_2}^{i_2}\cdot\ldots\cdot\sigma_{m_l}^{i_l}(z)\right)\right|\cdot\ldots\cdot\left|\left(\sigma_{m_{l-1}}^{i_{l-1}}\right)'\left(\sigma_{m_l}^{i_l}(z)\right)\right|\cdot\left|\left(\sigma_{m_l}^{i_l}\right)'(z)\right|$$

vanishes in the limit. A computation shows that

$$\left|\left(\sigma_n^i\right)'(z)\right|=\left(\frac{R_n}{|z-P_n^i|}\right)^2\,,$$

where P_n^i is the center and R_n the radius of C_n^i. Therefore $|\sigma_n^i{}'(z)| < 1$ for all $z \notin \overline{D_n^i}$. For $\tau = \sigma_n^i \sigma$ we define:

$$\theta(\tau) = \max_{z \in F} \left| \left(\sigma_n^i\right)'(\sigma(z)) \right| = \max_{z \in F} \frac{R_n}{|\sigma(z) - P_n^i|} = \frac{R_n}{k(P_n^i; \sigma)}.$$

Thus we have

$$\left|\sigma'_{(l)}(z)\right| \leq \theta^2\left(\sigma_{(l)}\right) \cdot \ldots \cdot \theta^2\left(\sigma_{(1)}\right).$$

which proves the claim because Lemma 2.2.2 guarantees that

$$\theta\left(\sigma_{(l)}\right) < \theta\left(\sigma_{(l-1)}\right) < \ldots < \theta\left(\sigma_{(1)}\right) = 1\,. \tag{2.2.4}$$

□

We are now ready to give a criterion for the absolute convergence of Series (5.6.2), and (5.6.3).

THEOREM 2.2.4. *Let G be iso-classical Schottky group of rank N. If there exists an index l with*

$$2N - 1 < \theta_l^{-2}, \tag{2.2.5}$$

where $\theta_l = \max_{\sigma \in G, |\sigma| = l+1} \theta(\sigma)$, then its corresponding Series (5.6.2), and (5.6.3) giving the normalized integrals of first kind and the corresponding period matrix are absolutely convergent.

PROOF. Inequality (2.2.4) implies that the sequence of θ_l is strictly monotone, i.e.,

$$0 < \theta_\infty < \ldots < \theta_{l+1} < \theta_l < \ldots < \theta_0 = 1\,.$$

If (2.2.5) holds for one index, it will also be true for all larger indices and for the limit as well. Let

$$G_n(\sigma) = \log \frac{z - \sigma(B_n)}{z - \sigma(A_n)}$$

denote a term of the infinite Sum (5.6.2). Substituting $\log(z - w) = f(w)$ in $G_n(\sigma)$ yields

$$\begin{aligned} |G_n(\sigma)| &= |f(\sigma(B_n)) - f(\sigma(A_n))| \\ &\leq \max_{w \in \sigma(F)} |f\prime(w)| \cdot |\sigma(B_n) - \sigma(A_n)| \\ &= \frac{1}{k(z;\sigma)} |\sigma(B_n) - \sigma(A_n)|\,. \end{aligned}$$

Let $\sigma_{(l)} = \sigma_{m_l}^{i_l} \cdot \ldots \cdot \sigma_{m_1}^{i_1} \in G$ be an infinite sequence growing this time to the left. Then we have

$$\left|\sigma_{(l+1)}(z) - \sigma_{(l+1)}(w)\right| \leq \theta^2\left(\sigma_{(l+1)}\right) \left|\sigma_{(l)}(z) - \sigma_{(l)}(w)\right|\,,$$

so that

$$\left|G_n\left(\sigma_{(l+k)}\right)\right| \leq \frac{\left|\sigma_{(l)}(B_n) - \sigma_{(l)}(A_n)\right|}{k\left(z; \sigma_{(l+k)}\right)} \theta_l^{2k}\,.$$

We would like to bound $k\left(z;\sigma_{(l+k)}\right)$ (from below) by $k\left(z;\sigma_{(l)}\right)$, but we cannot directly take advantage of (2.2.2) because $\sigma_{(l)}$ had to grow to the right, but it grows to the left. To fix this we introduce

$$k\left(z;l\right)=\min_{\sigma\in G,|\sigma|=l}k\left(z;\sigma\right)\,,$$

for which holds $k\left(z;l\right)<k\left(z;l+1\right)$. Thus we have $k\left(z;\sigma_{(l+k)}\right)<k\left(z;l+k\right)$, which implies

$$\left|G_n\left(\sigma_{(l+k)}\right)\right|\leq\frac{\left|\sigma_{(l+k)}\left(B_n\right)-\sigma_{(l+k)}\left(A_n\right)\right|}{k\left(z;l+k\right)}\leq\frac{\left|\sigma_{(l)}\left(B_n\right)-\sigma_{(l)}\left(A_n\right)\right|}{k\left(z;l\right)}\theta_l^{2k}\,. \tag{2.2.6}$$

This proves the convergence of Series (5.6.2), because G/G_n includes exactly $\left(2N-2\right)\left(2N-1\right)^l$ elements of length $l+1$. The convergence of the Series (5.6.3) for the period matrix is a consequence of the convergence of (5.6.2) because

$$B_{nm}\left(\sigma\right)=\log\left\{B_m,A_m,\sigma\left(B_n\right),\sigma\left(A_n\right)\right\}=G_m\left(A_n\right)+G_m\left(B_n\right)\,.$$

□

Inequality (2.2.6) will be essential when we develop apostiori criteria for an evaluation algorithm in Section 2.4.

The proof of the convergence of the Poincare theta series is done with a slightly different representation of the Series (2.1.2). Let

$$H_n\left(z;\sigma\right)=\left(\frac{1}{\sigma\left(z\right)-B_n}-\frac{1}{\sigma\left(z\right)-A_n}\right)\left(\gamma_\sigma z+\delta_\sigma\right)^{-2}$$

where

$$\left(\begin{array}{cc}\alpha_\sigma & \beta_\sigma\\ \gamma_\sigma & \delta_\sigma\end{array}\right)\in PSL\left(2,\mathbb{C}\right)$$

denotes the matrix representation of a linear fractional transformation σ. A short computation shows that for all $\sigma\in G$ we have

$$H\left(z;\sigma\right)=\frac{1}{z-\sigma^{-1}\left(B_n\right)}-\frac{1}{z-\sigma^{-1}\left(A_n\right)}\,, \tag{2.2.7}$$

so that

$$\omega_n\left(z\right)=\sum_{\sigma\in G_n\backslash G}H_n\left(z;\sigma\right)dz\,, \tag{2.2.8}$$

which is a (-2)-dimensional Poincare theta series.

THEOREM 2.2.5. *The theta series (2.2.8) corresponding to a Schottky group G are absolutely convergent iff*

$$\Gamma_n=\sum_{\sigma\in G_n\backslash G,\sigma\neq id}\frac{1}{\left|\gamma_\sigma\right|^2}<\infty\,. \tag{2.2.9}$$

PROOF. For genus $N = 1$ the coset $G_n \backslash G$ includes the identity only, therefore there is nothing to show. For $N > 1$ let $\sigma \in G_n \setminus G$. Since $\sigma^{-1}(\infty) = -\delta_\sigma/\gamma_\sigma$ we have

$$|\gamma_\sigma z + \delta_\sigma| = \left| z + \frac{\delta_\sigma}{\gamma_\sigma} \right| = \left| z - \sigma^{-1}(\infty) \right| |\gamma_\sigma| \ ,$$

so that

$$\frac{l(\sigma,n)}{K(z;\sigma^{-1})^2} \frac{1}{|\gamma_\sigma|^2} \leq |H_n(z;\sigma)| \leq \frac{L(\sigma,n)}{k(z;\sigma^{-1})^2} \frac{1}{|\gamma_\sigma|^2} \ , \tag{2.2.10}$$

where

$$l(\sigma,n) = \frac{|A_n - B_n|}{K(A_n;\sigma)\, K(B_n;\sigma)} \quad \text{and} \quad L(\sigma,n) = \frac{|A_n - B_n|}{k(A_n;\sigma)\, k(B_n;\sigma)} \ .$$

Because of (2.2.3) we can guarantee positive upper and lower bounds for all $|\sigma| > 1$, which proves the claim. □

For $N > 1$, we have $G_n \setminus G \subset G = \cup_{n=1,\ldots,N} G_n \setminus G$, so that

$$\Gamma_n < \Gamma < \sum_{n=1}^{N} \Gamma_n \qquad \text{with} \qquad \Gamma = \sum_{\sigma \in G, \sigma \neq id} \frac{1}{|\gamma_\sigma|^2} \ ,$$

which implies that all ω_n, $n = 1, \ldots, N$ converge absolutely iff $\Gamma < \infty$. This also holds for $N = 1$. A short computation shows that

$$\frac{1}{|\gamma_{\sigma^k}|\, 2} = \frac{|A-B|^2}{\left|1 - \sqrt{\mu^k}\right|^2} |\mu|^k \leq \left(\frac{|A-B|}{1 - \left|\sqrt{\mu}\right|} \right)^2 |\mu|^k$$

and therefore

$$\Gamma < \left(\frac{|A-B|}{1 - \left|\sqrt{\mu}\right|} \right)^2 \frac{2\,|\mu|}{1 - |\mu|} < \infty$$

for genus 1.

COROLLARY 2.2.6. *All ω_n, $n = 1, \ldots, N$ converge absolutely iff $\Gamma < \infty$.*

Another consequence of Theorem 2.2.5 is that if ω_n converges absolutely for one $z_0 \in F$, then it also converges for all $z \in F$. None of these results allow us to estimate any error or deliver a criterion that can be evaluated computationally. This is also true for the next lemma, but it shows the right direction.

LEMMA 2.2.7. *For $\sigma \in G$ with $[\sigma] = [il, \ldots, jr]$, let*

$$\kappa_L(\sigma) = \left| \frac{\gamma_\sigma}{\gamma_{\sigma_l^{-j}\sigma}} \right| \quad \text{and} \quad \kappa_R(\sigma) = \left| \frac{\gamma_\sigma}{\gamma_{\sigma\sigma_r^{-i}}} \right| \ .$$

Then $\kappa_L(\sigma) = \kappa_R(\sigma^{-1})$, and Γ converges absolutely if

$$2N - 1 < \kappa^2 \qquad \text{with} \qquad \kappa = \inf_{\sigma \in G, |\sigma| > 1} \kappa_{L/R}(\sigma) \ .$$

PROOF. For any $\tilde{\kappa} < \kappa$, and hence also for all $\tilde{\kappa} \in]2N-1, \kappa[$ there exists a k_0 such that $\tilde{\kappa} < \kappa_{L/R}(\sigma)$ for all $\sigma \in G$ with $|\sigma| \geq k_0$ and thus for $\sigma_{(k)} = \sigma_{s_1}^{i_1} \cdot \ldots \cdot \sigma_{s_k}^{i_k}$ we have

$$\left|\gamma_{\sigma_{(k)}\sigma}\right| = \left|\frac{\gamma_{\sigma_{(k)}\sigma}}{\gamma_{\sigma_{(k-1)}\sigma}}\right| \ldots \left|\frac{\gamma_{\sigma_{(k_0+1)}\sigma}}{\gamma_{\sigma_{(k_0)}}}\right| \left|\gamma_{\sigma_{(k_0)}}\right| > \kappa^{k-k_0} \left|\gamma_{\sigma_{(k_0)}}\right| .$$

Therefore, and because of the fact that there are exactly $2N(2N-1)^{k-1}$ transformations of length k, we have

$$\sum_{|\sigma|\geq k_0} \frac{1}{|\gamma_\sigma|^2} \leq \sum_{|\sigma|\geq k_0} \frac{\tilde{\kappa}^{2k_0-2|\sigma|}}{\left|\gamma_{\sigma_{(k_0)}}\right|^2} \leq \frac{\tilde{\kappa}^{2k_0}}{\gamma_{k_0}^2} \sum_{|\sigma|\geq k_0} \frac{1}{\tilde{\kappa}^{2|\sigma|}} \leq \frac{\tilde{\kappa}^{2k_0}}{\gamma_{k_0}^2} \frac{2N}{2N-1} \sum_{i=k_0}^{k} \left(\frac{2N-1}{\tilde{\kappa}^2}\right)^k$$

where $\gamma_n = \min_{|\sigma|=n} |\gamma_\sigma|$. □

To obtain error estimates one has to control $\kappa_{L/R}$.

LEMMA 2.2.8. *Let G be a Schottky group generated by $\sigma_1, \ldots, \sigma_N$ with fixed points A_n, B_n, $n = 1, \ldots, N$. For $\sigma \in G$ with $[\sigma] = [\ldots, ir]$ we have*
(2.2.11)

$$\kappa_R(\sigma) = \frac{1}{|A_r - B_r|} \max_s \left|C_r^s + \frac{\delta_\tau}{\gamma_\tau}\right| \sqrt{|\mu_r^{is}|} - \left|C_r^{-s} + \frac{\delta_\tau}{\gamma_\tau}\right| \sqrt{|\mu_r^{-is}|} = \kappa_L\left(\sigma^{-1}\right)$$

with $C_r^1 = A_r, C_r^{-1} = B_r$ and $\tau = \sigma\sigma_r^{-i}$.

PROOF. We have $\gamma_\sigma = \gamma_\tau \alpha_{\sigma_r^i} + \delta_\tau \gamma_{\sigma_r^i}$. Using Representation (2.1.1) one sees that

$$\begin{aligned}
\frac{\gamma_\sigma}{\gamma_\tau} &= \alpha_{\sigma_r^i} + \gamma_{\sigma_r^i} \frac{\delta_\tau}{\gamma_\tau} \\
&= \frac{1}{A_r - B_r} \left(A_r \sqrt{\mu_r^i} - B_r / \sqrt{\mu_r^i} + \left(\sqrt{\mu_r^i} - 1/\sqrt{\mu_r^i}\right) \frac{\delta_\tau}{\gamma_\tau} \right) \\
&= \frac{1}{A_r - B_r} \left(\left(A_r + \frac{\delta_\tau}{\gamma_\tau}\right) \sqrt{\mu_r^i} - \left(B_r + \frac{\delta_\tau}{\gamma_\tau}\right) \sqrt{\mu_r^{-i}} \right) .
\end{aligned}$$

□

The fact that $\tau^{-1}(\infty) = -\delta_\tau/\gamma_\tau$ yields

$$k\left(C_r^s; \tau^{-1}\right) \leq \left|C_r^s + \frac{\delta_\tau}{\gamma_\tau}\right| \leq K\left(C_r^s; \tau^{-1}\right) .$$

Let $\sigma = \tau\sigma_r^i$ with $|\sigma| > 0$. We let

$$\overline{\kappa_R}(\sigma) = \frac{1}{|A_r - B_r|} \left(k\left(C_r^{\tilde{s}}; \tau^{-1}\right) \sqrt{|\mu_r^{i\tilde{s}}|} - K\left(C_r^{-\tilde{s}}; \tau^{-1}\right) \sqrt{|\mu_r^{-i\tilde{s}}|} \right) ,$$

where $\tilde{s}$ realizes the maximum in (2.2.11), e.g.

$$\kappa_R(\sigma) = \frac{1}{|A_r - B_r|} \left(\left|C_r^{\tilde{s}} + \frac{\delta_\tau}{\gamma_\tau}\right| \sqrt{|\mu_r^{i\tilde{s}}|} - \left|C_r^{-\tilde{s}} + \frac{\delta_\tau}{\gamma_\tau}\right| \sqrt{|\mu_r^{-i\tilde{s}}|} \right) .$$

Note that $\overline{\kappa_R}(\sigma) < \kappa_R(\sigma)$.

LEMMA 2.2.9. *Let G be a iso-classical Schottky group. Then for all $\epsilon > 0$ there exists an n_ϵ such that*

$$\kappa_R(\sigma) - \overline{\kappa_R}(\sigma) < \epsilon \qquad \forall\, |\sigma| \geq n_\epsilon\,.$$

PROOF. A computation yields

$$\kappa_R(\sigma) - \overline{\kappa_R}(\sigma) \;\leq\; \frac{\sqrt{|\mu_r^{i\tilde{s}}|} + \sqrt{|\mu_r^{-i\tilde{s}}|}}{|A_r - B_r|} Diam\left(\tau^{-1}(F)\right)\,,$$

which implies the claim along the lines of the proof of Lemma 2.2.3. □

This leads to a result that allows us to check for $2N-1 < \kappa^2$ by evaluating $\overline{\kappa_R}(\sigma)$ for finitely many transformations σ.

LEMMA 2.2.10. *Let G be a iso-classical Schottky group and*

$$\kappa_n = \min_{|\sigma|=n} \overline{\kappa_R}(\sigma)\,.$$

Then $\kappa_n \leq \kappa_{|\sigma|} \leq \overline{\kappa_R}(\sigma) < \kappa_R(\sigma)$ for all $n \leq |\sigma|$, and we have

$$\kappa_1 < \ldots < \kappa_n < \ldots < \kappa_\infty = \kappa\,.$$

PROOF. Let $\sigma = \sigma_l^j \tau \sigma_r^i$ with $|\sigma| = n+1$ and $\kappa_{n+1} = \overline{\kappa_R}(\sigma)$. Inequality (2.2.2) guarantees

$$k\left(C_r^{\tilde{s}};\sigma\right) > k\left(C_r^{\tilde{s}};\tau\sigma_r^i\right) \qquad \text{and} \qquad K\left(C_r^{\tilde{s}};\sigma\right) < K\left(C_r^{\tilde{s}};\tau\sigma_r^i\right)\,,$$

so that

$$\kappa_n \leq \overline{\kappa_R}\left(\tau\sigma_r^i\right) < \overline{\kappa_R}(\sigma) = \kappa_{n+1}\,.$$

To prove that $\kappa_\infty = \kappa$, let $\sigma_{(n)}$ be a sequence fulfilling $\kappa_n = \overline{\kappa_R}\left(\sigma_{(n)}\right)$ and $\upsilon_{(n)}$ one with the property

$$\kappa_R\left(\upsilon_{(n)}\right) = \min_{|\sigma|=n} \kappa_R(\sigma)\,.$$

Moreover, is $\lim_{n\to\infty} \kappa_R\left(\upsilon_{(n)}\right) = \kappa$ implies

$$\kappa_n = \overline{\kappa_R}\left(\sigma_{(n)}\right) \leq \overline{\kappa_R}\left(\upsilon_{(n)}\right) < \kappa_R\left(\upsilon_{(n)}\right) < \kappa + \epsilon\,,$$

so that $\kappa_\infty \leq \kappa$. On the other hand, Lemma 2.2.9 guarantees that $\kappa_R\left(\sigma_{(l)}\right)$ and $\overline{\kappa_R}\left(\sigma_{(l)}\right)$ have the same limit and thus $\kappa \geq \lim_{n\to\infty} \kappa_R\left(\sigma_{(n)}\right) = \kappa$. □

The following theorem summerizes the last result, Theorem 2.2.5, and Lemma 2.2.7 as an evaluable criterion for the convergence of the Poincare theta series giving normalized differentials of first kind.

THEOREM 2.2.11. *Let G be a iso-classical Schottky group of rank N. If there exists an index l with*

$$2N - 1 < \kappa_l^2,$$

then the series (2) giving normalized differentials of first kind are absolute convergent.

The latter is analogous to Theorem 2.2.4, which handles the case of the normalized integrals.

2.3. Error Estimates for Schottky Series

Throughout this section, let G be always a iso-classical Schottky group. For a subset S of G/G_n, we approximate Ω_n by

$$\hat{\Omega}_n(z;S) = \sum_{\sigma\in S} G(\sigma)$$

with associated error

$$\varepsilon_{\Omega_n}(z;S) = \left|\Omega_n(z) - \hat{\Omega}_n(z;S)\right| .$$

Let $\sigma_{(l)} = \sigma_{n_l}^{i_l}\cdot\ldots\cdot\sigma_{n_1}^{i_1}$ be a leftward growing sequence in G/G_n i.e., $n_1 \neq n$. Then the proof of Theorem 2.2.4 provides us with estimates for G_n (Inequality 2.2.6)

$$\left|G_n\left(\sigma_{(l+k)}\right)\right| \leq C_{\Omega_n}\left(z;\sigma_{(l+k)}\right) \leq C_{\Omega_n}\left(z;\sigma_{(l)}\right)\theta_l^{2k}, \tag{2.3.1}$$

with

$$C_{\Omega_n}(z;\sigma) = \frac{|\sigma(B_n) - \sigma(A_n)|}{k(z;|\sigma|)} .$$

Let $\sigma = \sigma_{n_k}^{i_k}\cdot\ldots\cdot\sigma_{n_1}^{i_1} \in G$. Then $\sigma_{(l)} = \sigma_{n_l}^{i_l}\cdot\ldots\cdot\sigma_{n_1}^{i_1}$, $l = 1,\ldots,k$ are called *suffixes* of σ. The id is a suffix of any group element. If a set S includes all the suffixes of its elements it is called *suffix colsed*. The set of suffixes of a subset S of G is called *suffix closure* and denoted by $\overline{S}^s$. Relation (2.3.1) shows that the subset S of $G/G_n \subset G$ which is used for approximation should be suffix closed.

The *boundary* $\partial^s S$ of a suffix closed subset S, given by

$$\partial^s S = \left\{\sigma \in S^{-1} \left| \overline{\{\sigma\}}^s \cap S^{-1} = \{\sigma\}\right.\right\} ,$$

has the property that for all $\sigma \in S^{-1}$ it includes exactly one element that is a suffix of σ. This enables a natural projection

$$\begin{array}{rccl} \pi_S: & S^{-1} & \to & \partial^s S \\ & \sigma & \mapsto & \partial^s S \cap \overline{\{\sigma\}}^s . \end{array}$$

The set $\pi_S^{-1}(\tau)$ consists of all $\sigma \in S^{-1}$ that have τ as suffix. The set $\pi_S^{-1}(\tau)$ is the *suffix cone* $Cone^s(\tau)$, i.e.

$$Cone^s(\tau) = \left\{\sigma \in G \left| \tau \in \overline{\{\sigma\}}^s \right.\right\} ,$$

which yields

Corollary 2.3.1. *Let S be a suffix closed subset. Then we have*

$$S^{-1} = \dot{\cup}_{\sigma\in\partial^s S} Cone^s(\sigma) .$$

LEMMA 2.3.2. *Let S be a suffix closed subset of G/G_n. If $2N-1<\theta_{|\tau|}^{-2}$ for all $\tau\in\partial^s S$, then*

$$\varepsilon_{\Omega_n}(z;S)=\left|\Omega_n(z)-\hat{\Omega}_n(z;S)\right|<\sum_{\tau\in\partial^s S}\hat{\varepsilon}_{\Omega_n}(z;\tau)=\hat{\varepsilon}_{\Omega_n}(z;S)\ ,$$

with $\hat{\varepsilon}_{\Omega_n}(z;\tau)=C_{\Omega_n}(z;\tau)\,R\left(\theta_{|\tau|}^2\right)$ and

$$R(q)=\sum_{\sigma\in Cone^s(\tau)}q^{|\sigma|-|\tau|}=(1-(2N-1)\,q)^{-1}\ .$$

PROOF. We have

$$\left|\Omega_n(z)-\hat{\Omega}_n(z;S)\right|<\sum_{\sigma\in S^{-1}}|G_n(z;\sigma)|\ ,$$

so that in light of Corollary 2.3.1 we need to show

$$\sum_{\sigma\in Cone^s(\tau)}|G_n(z;\sigma)|<\hat{\varepsilon}_{\Omega_n}(z;\tau)\ .$$

Relation (2.3.1) yields

$$\sum_{\sigma\in Cone^s(\tau)}C_{\Omega_n}(z;\sigma)<C_{\Omega_n}(z;\tau)\sum_{\sigma\in Cone^s(\tau)}\theta_{|\tau|}^{2(|\sigma|-|\tau|)}=\hat{\varepsilon}_{\Omega_n}(z;\tau)\ ,$$

because there are exactly $(2N-1)^k$ transformations in $Cone^s(\tau)$ of length $|\tau|+k$. □

To approximate the elements of the period matrix B_{nm} by

$$\hat{B}_{nm}=\sum_{\sigma\in S}B_{nm}(\sigma)\ ,$$

let now S be a suffix closed subset of $G_m\setminus G/G_n$. The associated error will be defined like the one above and denoted by $\varepsilon_{B_{nm}}(S)$. We have already seen that $B_{nm}(\sigma)=G_m(A_n)+G_m(B_n)$, thus in analogy to (2.2.5) we have

$$\left|B_{nm}\left(\sigma_{(l+k)}\right)\right|\leq C_{B_{nm}}\left(\sigma_{(l+k)}\right)\leq C_{B_{nm}}\left(\sigma_{(l)}\right)\theta_l^{2k}\,,$$

with

$$C_{B_{nm}}(\sigma)=C_{\Omega_m}(A_n)+C_{\Omega_m}(B_n)\ .$$

Summing up the error is more difficult then before. We have

$$\varepsilon_{B_{nm}}(S)=\left|B_{nm}-\hat{B}_{nm}(S)\right|<\sum_{\sigma\in G_m\backslash G/G_n\backslash S}|B_{nm}(\sigma)|\ .$$

Since $S \subset G_m \setminus G/G_n = (G_m \setminus G) \cap (G/G_n)$, we get a simular result to Corollary 2.3.1:

$$\begin{aligned}(G_m \setminus G/G_n) \setminus S &= (G_m \setminus G) \cap ((G/G_n) \setminus S) \\ &= (G_m \setminus G) \cap \dot{\cup}_{\sigma \in \partial^s S} Cone^s(\sigma) \\ &= \dot{\cup}_{\sigma \in \partial^s S} Left_m(\sigma) \ ,\end{aligned}$$

where $Cone^s_m(\sigma) = Cone^s(\sigma) \cap G_m \setminus G$. This yields further

$$\varepsilon_{B_{nm}}(S) < \sum_{\tau \in \partial^s S} \sum_{\sigma \in Cone^s_m(\tau)} |B_{nm}(\sigma)| < \sum_{\tau \in \partial^s S} C_{B_{nm}}(\tau) \sum_{\sigma \in Cone^s_m(\tau)} \theta^{2(|\sigma|-|\tau|)}_{|\tau|} \ ,$$

which leads to

LEMMA 2.3.3. *Let S be a suffix closed subset of $G_m \setminus G/G_n$. If $2N-1 < \theta^{-2}_{|\tau|}$ for all $\tau \in \partial^s S$, then*

$$\varepsilon_{B_{nm}}(S) = \left| B_{nm}(z) - \hat{B}_{nm}(S) \right| < \sum_{\tau \in \partial^s S} \hat{\varepsilon}_{\hat{B}_{nm}}(\tau) = \hat{\varepsilon}_{B_{nm}}(S)$$

with $\hat{\varepsilon}_{B_{nm}}(\tau) = C_{B_{nm}}(\tau) R_m(\tau; q)$ and

$$R_m(\sigma; q) = \sum_{\sigma \in Cone^s_m(\tau)} q^{|\sigma|-|\tau|} = (2N-2) \frac{q}{1+q} R(q) + \begin{cases} \frac{1}{1+q} & \sigma \in G_m \setminus G \\ 0 & otherwise \, . \end{cases}$$

We could have estimated $R_m(\tau; q)$ by $R(q)$, but we need the accuracy of Lemma 2.3.3. We derive the expressions for $R_m(\tau; q)$ in the Appendix. Figure 2.3.1 shows the two ratios $\phi_N(q) = R_m(\sigma; q)/R(q)$ with $\sigma \in G_m \setminus G$ and $\varphi_N(q) = R_m(\sigma; q)/R(q)$ with $\sigma \notin G_m \setminus G$ for positive $q < 1/(2N-1)$ and different ranks $(N = 2, 3, 4, 5)$ of the Schottky group G. For slowly converging series where q almost equals $1/(2N-1)$ we get an improvement by a factor of $1-q$. For small q, i.e., for fast converging series, $R_m(\sigma; q)$ and $R(q)$ almost coincide for $\sigma \in G_m \setminus G$, but for $\sigma \notin G_m \setminus G$ $R_m(\sigma; q)$ gets small compared to $R(q)$.

Analogous to Ω_n we approximate ω_n by

$$\hat{\omega}_n(z; S) = \sum_{\sigma \in S} H_n\left(z; \sigma^{-1}\right) dz$$

and denote the associated error by $\varepsilon_{\omega_n}(z; S)$. The proof of Theorem 2.2.5 provides estimates for H_n. Relation (2.2.10) guarantees:

$$\left| H_n\left(z; \sigma^{-1}\right) \right| \leq \frac{L(\sigma^{-1}, n)}{k(z; \sigma)^2} \frac{1}{|\gamma_\sigma| \, 2} \, ,$$

Let $\sigma_{(l)} = \sigma^{i_l}_{n_l} \cdot \ldots \cdot \sigma^{i_1}_{n_1}$ be a leftward growing sequence in G/G_n, i.e., $n_1 \neq n$. The definition of L and Inequality (2.2.2) imply that $L\left(\sigma^{-1}_{(l+k)}\right) < L\left(\sigma^{-1}_{(l)}\right)$, which yields in combination with $k\left(z; \sigma_{(l+k)}\right) < k(z; l)$ and Lemma 2.2.10

$$\left| H_n\left(z; \sigma^{-1}_{(l+k)}\right) \right| \leq C_{\omega_n}\left(z; \sigma_{(l+k)}\right) \leq C_{\omega_n}\left(z; \sigma_{(l)}\right) \kappa_l^{-2k} \, , \tag{2.3.2}$$

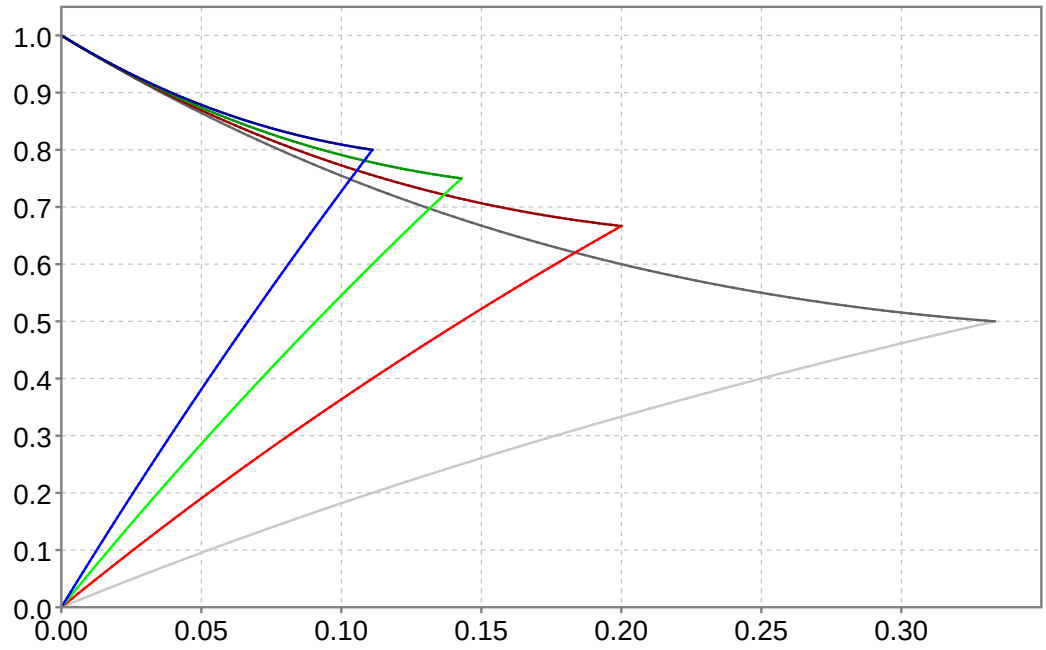


FIGURE 2.3.1. The improvement of the estimates achieved by using $R_m(\sigma; q)$ instead of $R(q)$ expressed by their ratio given as ϕ_N(dark) or φ_N(light) for $N = 2$(gray), 3(red), 4(green), 5(blue).

where $l > 1$,

$$C_{\omega_n}(z;\sigma) = \frac{L(\sigma^{-1}, n)}{k(z;|\sigma|)^2} \frac{1}{|\gamma_\sigma|^2}.$$

Analogous to Lemma 2.3.2, we conclude

LEMMA 2.3.4. *Let S be a suffix closed a subset of G/G_n. If $2N-1 < \kappa^2_{|\tau|}$ for all $\tau \in \partial^s S$, then*

$$\varepsilon_{\omega_n}(z;S) = |\omega_n(z) - \hat{\omega}_n(z;S)| < \sum_{\tau\in\partial^s S} \hat{\varepsilon}_{\omega_n}(z;\tau) = \hat{\varepsilon}_{\omega_n}(z;S)$$

with $\hat{\varepsilon}_{\omega_n}(z;\tau) = C_{\omega_n}(z;\tau) R\left(\tau; \kappa^{-2}_{|\tau|}\right)$.

The weak spot of our estimates is that we majorize $1/k(z;\sigma^{-1})$ with $1/k(z;|\sigma|)$. This can be rather bad if the radii of the isometric circles differ significantly. The Appendix deals with this problem and improves the estimates at the cost of technical complications.

2.4. Evaluation Methods for Schottky Series

Theorems 2.2.4 and 2.2.11 provide us with criteria for the convergence of the Schottky series. Ω_n and B_{nm} converge if

$$q_\infty^\Omega = (2N-1)\,\theta_\infty^2 < 1\,, \tag{2.4.1}$$

and ω_n converges if

$$q_\infty^\omega = (2N-1)\,\kappa_{\tilde{l}}^{-2} < 1\,. \tag{2.4.2}$$

The monotone series $(q_l^\star)$ converges very fast and experiments show that three terms suffice to extrapolate the limit with a high accuracy. Figure 2.4.1 shows the analysis of the convergence Criteria (2.4.1) and (2.4.2) for several examples taken from the family of Riemann surfaces that are associated with constant mean curvature tori in R^3, S^3, H^3 [**Bob91**]. These Schottky groups allow an anti-holomorphic involution in addition to their hyper-elliptic and are Fuchsian groups of the second kind, for which the Schottky series converges [**BBE$^+$94**]. For our examples of genus 2 this is obvious because they are circle decomposable. The examples show that our criteria can guarantee convergence for integrals, and simultaneously fail for differentials. In all cases we have examined the integrals converge faster than the differentials. It is astonishing that the criteria can still guarantee convergence when the circles almost touche. The $q_\infty^\star$ are good indicators for the speed of convergence. For good numerical results these values should not be too close to 1. The first example in Figure 2.4.1, which belongs to the famous Wente torus having a threefold symmetry [**Hei95**, page 38], is already at the limit of what we can handle with this method. It is a good candidate for test cases.

We need to determine a suffix closed subset $S_\star(z;\varepsilon)$ of G/G_n for a given accuracy $\varepsilon > 0$ such that $\varepsilon_\star(z; S_\star(z;\varepsilon)) < \varepsilon$. For efficiency reasons and numerical stability as well the subset should be as small as possible.

Let G be a iso-classical Schottky group and l the smallest integer such that $q_l^\star < 1$. In order to use one of Lemmas 2.3.2, 2.3.3, and 2.3.4 any suffix closed subset must contain all elements with a word length less than or equal to l. We denote the set of those elements with

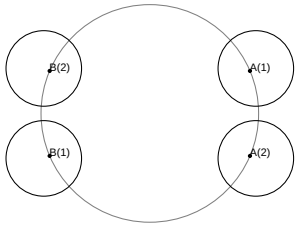
$$S_n(l) = \{\sigma \in G/G_n \,||\sigma| \le l\}\,,$$

which is also a suffix closed subset.

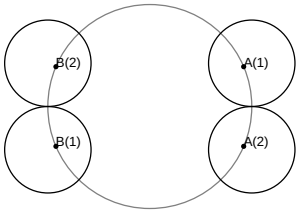
A straightforward approach to determine a smallest suffix closed subset is the following

Algorithm 2.4.1. *Let $\varepsilon > 0$, G a iso-classical Schottky group and l the smallest integer such that $q_l^\star < 1$:*

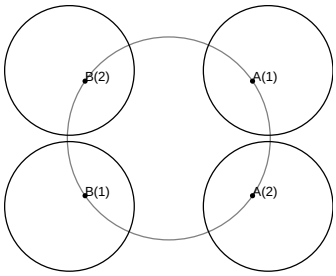
1. *Let $S_{(i=1)} = S_n(l)$.*
2. *If $\hat{\varepsilon}\left(z; S_{(i)}\right) < \varepsilon$, let $S_\star(z;\varepsilon) = S_{(i)}$ and finish.*
3. *Seek $\sigma_{(i)} \in \partial^s S_{(i)}$ with $C_\star\left(z;\sigma_{(i)}\right) = \max_{\sigma\in\overline{S_{(i)}}} C_\star(z;\sigma)$.*



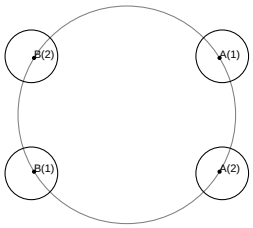
l	q_l^ω	q_l^Ω
1	-	1.5729726
2	2.1919203	0.6089423
3	0.7095799	0.6026912
4	0.7008699	0.5972790
5	0.6937671	0.5972347
6	0.6937057	0.5971962



l	q_l^ω	q_l^Ω
1	-	2.9883689
2	5.5728355	0.9239982
3	1.1721161	0.9228728
4	1.1678997	0.8975555
5	1.1295999	0.8975374
6	1.1295356	0.8971324



l	q_l^ω	q_l^Ω
1	-	2.4897787
2	6.5114680	1.1106899
3	1.6407322	1.0831381
4	1.5694645	1.0162737
5	1.4367818	1.0148276
6	1.4333759	1.0113023



l	q_l^ω	q_l^Ω
1	-	0.2075865
2	0.2221489	0.1390086
3	0.1452401	0.1379418
4	0.1440661	0.1378807
5	0.1439990	0.1378796
6	0.1439978	0.1378795

FIGURE 2.4.1. Analysis of the convergence Criteria (2.4.1) and (2.4.2) for several examples taken from the family of Riemann surfaces delivering all constant mean curvature tori in R^3, S^3, H^3. The first example in Figure 2.4.1 is the Riemann surface of the famous Wente torus having threefold symmetry.

4. Let $S_{(i+1)} = S_{(i)} \cup \left\{\sigma_{(i)}\right\}$.
5. Let $i = i + 1$ *and continue with Step 2.*

The algorithm above is simple and gives the optimal result, but the search in Step 3. makes it rather expensive. A different approach is inspired by the following property of the boundary of a suffix closed subset:

COROLLARY 2.4.2. *Let* S *be a suffix closed subset* G/G_n. *Then*

$$1 = \frac{2N-1}{2N-2} \sum_{\sigma \in \partial^s S} (2N-1)^{-|\sigma|} .$$

PROOF. Let $l = \max_{\sigma \in S} |\sigma|$. Then the boundary $\partial^s S_n(l) \subset S^{-1}$. $\pi_S^{-1}(\sigma)$ contains exactly $(2N-1)^{l+1-|\sigma|}$ transformations of word length $l+1$, i.e.,

$$\# \left(\pi^{-1}_{S|\partial^s S_n(l)} \right) = (2N-1)^{l+1-|\sigma|} ,$$

which implies

$$\begin{aligned} 1 &= \frac{1}{(2N-2)(2N-1)^l} \sum_{\sigma \in \partial^s S_n(l)} 1 \\ &= \frac{1}{(2N-2)(2N-1)^l} \sum_{\sigma \in \partial^s S} \sum_{\hat{\sigma} \in \pi^{-1}_{S|\partial^s S_n(l)}} 1 \\ &= \frac{1}{(2N-2)(2N-1)^l} \sum_{\sigma \in \partial^s S} (2N-1)^{l+1-|\sigma|} . \end{aligned}$$

□

This partition of 1 like property yields

ALGORITHM 2.4.3. *Let* $\varepsilon > 0$, G *a iso-classical Schottky group and* l *the smallest integer such that* $q_l^\star < 1$*:*

1. Let $S_{(i=1)} = S_n(l)$.
2. Let $\tilde{S}_{(i)} = \left\{ \sigma \in \partial^s S_{(i)} \,\middle|\, \hat{\varepsilon}_\star(z;\sigma) > \frac{2N-1}{2N-2} (2N-1)^{-|\sigma|} \varepsilon \right\}$.
3. If $\tilde{S}_{(i)} = \{\emptyset\}$, *let* $S_\star(z;\varepsilon) = S_{(i)}$ *and finish.*
4. Let $S_{(i+1)} = S_{(i)} \cup \tilde{S}_{(i)}$.
5. Let $i = i + 1$ *and continue with Step 2.*

This method offers a good trade-off between the size of $S_\star(z;\varepsilon)$ on one hand and computational cost on the other. Figure 2.4.2 shows the results of Algorithm 2.4.3 for the differential of first kind ω_1 for the example of the Wente tours with threefold symmetry. The algorithm manages to keep the error $\hat{\varepsilon}_{\omega_1}$ in the selected clipping within a range of $[3.3, 5.3] \cdot 10^{-4}$. Simultaneously the number of terms used in the approximation of $\#S_{\omega_1}(z; 10^{-3})$ varies in the range of $[113, 962]$. Great values occur only at "hot spots", where fixed points of elements of the Schottky

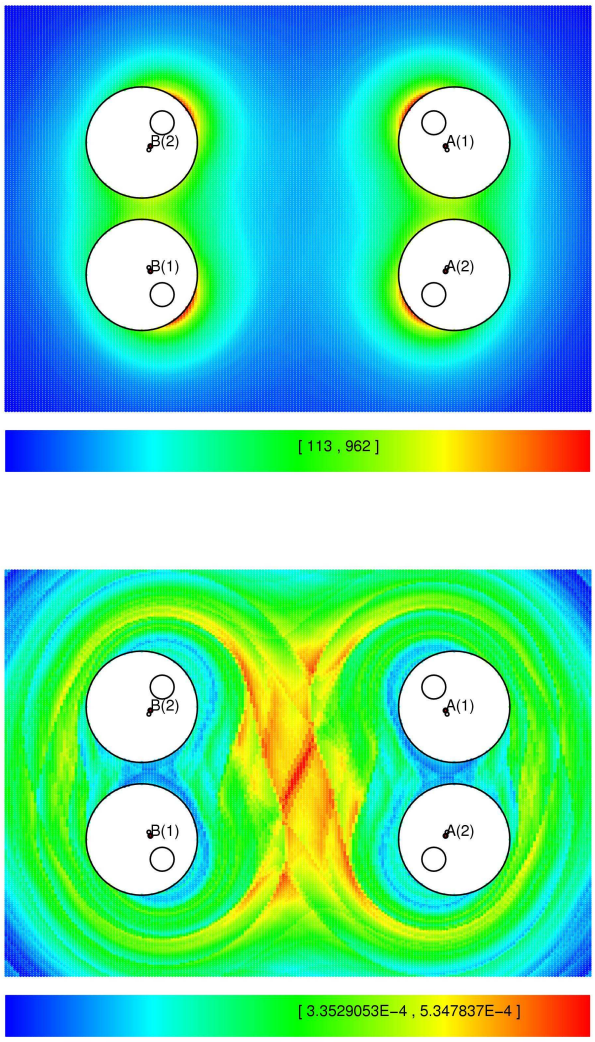


FIGURE 2.4.2. Results of Algorithm 2.4.3: $\#S_{\omega_1}(z;10^{-3})$ (top) and $\hat{\varepsilon}_{\omega_1}(z;S_{\omega_1}(z;10^{-3}))$ for the Wente torus with threefold symmetry.

group get close to the boundary of the fundamental domain F. However, most of the fundamental domain is covered by values at the lower bound of the range, from which the numerical performance benefits.

In Step 2 of Algorithm 2.4.3 we stay for all $\sigma \in \partial^s S_{(i)} \backslash \tilde{S}_{(i)}$ below the threshhold $\hat{\varepsilon}_\star(z;\sigma)$. We can therefore increase the ε by

$$\frac{2N-1}{2N-2}(2N-1)^{-|\sigma|}\varepsilon - \hat{\varepsilon}_\star(z;\sigma)$$

after each test. We will refer to this optimized version as Algorithm 2.4.3*. The optimized algorithm reduces the range of $\hat{\varepsilon}_{\omega_1}(z; S_{\omega_1}(z; 10^{-3}))$ in the case above to $[4.0, 6.8] \cdot 10^{-4}$ and $\#S_{\omega_1}(z; 10^{-3})$ to $[87, 812]$, an improvement of 20%. Figure 2.4.6 (also) shows analogous plots to those in Figure 2.4.2 for Algorithm 2.4.3*, which lacks symmetry as a result of optimization.

The crucial constants $C_\star(z; \sigma)$ separate into a fairly inexpensive computable z-independent factor and an expensive z-dependent factor $k_{|\sigma|}(z)^{-p_\star}$, with $p = 1$ for Ω_n and B_{nm}, and $p = 2$ for ω_n. The complexity of the computation of $k_l(z)$ grows exponentially in l, thus it is inevitable to replace $k_l(z)$ by $k_{\tilde{l}}(z)$ with a small $\tilde{l} = 1, 2, 3 \leq l$. This is negligible, because $k_l(z)$ converges very fast, particularly with regard to the fact that we earlier replaced $k(z; \sigma)$ by $k_{|\sigma|}(z)$ (Appendix). Just as the computation of $q_l^\star$. We do not determine more of them then we have already computed for checking convergence. At this point we have reduced our evaluation methods to more or less elementary operations. Figures 2.4.4 and 2.4.3 shows plots, again for the Wente torus having threefold symmetry, of the integral Ω_1 respectively for the differential ω_1 of first kind. Even the integral has periods its absolute value is continues. To realize a period of $2\pi i$ its argument values have to switch their sign at the cut. The plots of the differential of first kind nicely show its $2N - 2$ roots. Each corresponds to an isometric circle, only the pair of circles of the generator giving the coset related to that differential lack roots.

After all that effort a natural questions arises: How strict are our estimates? We estimated the absolute series and thus we can only be as good as these. Let us denote them with $|\Omega|_n$ and $|\omega|_n$. We define their approximation for a given suffix closed subset S of G/G_n as

$$\begin{aligned} \left|\hat{\Omega}\right|_n(z; S) &= \sum_{\sigma \in S} |G_n(z; \sigma)| \quad \text{and} \\ |\hat{\omega}|_n(z; S) &= \sum_{\sigma \in S} \left|H_n\left(z; \sigma^{-1}\right)\right| . \end{aligned}$$

Plots of the defects $d_\star(z; \varepsilon) = |\star|_n(z) - |\widehat{\star}|_n(z; S_\star(z; \varepsilon))$ will reveal the results of our effort and show how strict our estimates are. To approximate the defect $d_\star(z; \varepsilon)$ we approximate $|\star|_n(z)$ by $|\widehat{\star}|_n(z; S_\star(z; \tilde{\varepsilon}))$, where $\tilde{\varepsilon}$ is significant by smaller than ε. For a relative error of 10% $\tilde{\varepsilon}$ should be 10 times smaller than the resulting minimum of $d_\star(z; \varepsilon)$. Figures 2.4.5 and 2.4.6 shows $\#S_\star(z; 10^{-3})$, $\hat{\varepsilon}_\star(z; S_\star(z; 10^{-3}))$, and $d_\star(z; 10^{-3})$ for Ω_1 and ω_1 respectively. The algorithm manages to keep the defect $d_{\Omega_1}(z; 10^{-3})$ in the selected clipping in the range of $[1.5, 2.3] \cdot 10^{-4}$ and $d_{\omega_1}(z; 10^{-3})$ in $[0.7, 2.3] \cdot 10^{-4}$.

Caching many of the costly entities is important if one wants to evaluate the Schottky series multiple times for constant Schottky data. For example we can reduce the effort of the test in line 2 of Algorithm 2.4.3 to a single multiplication,

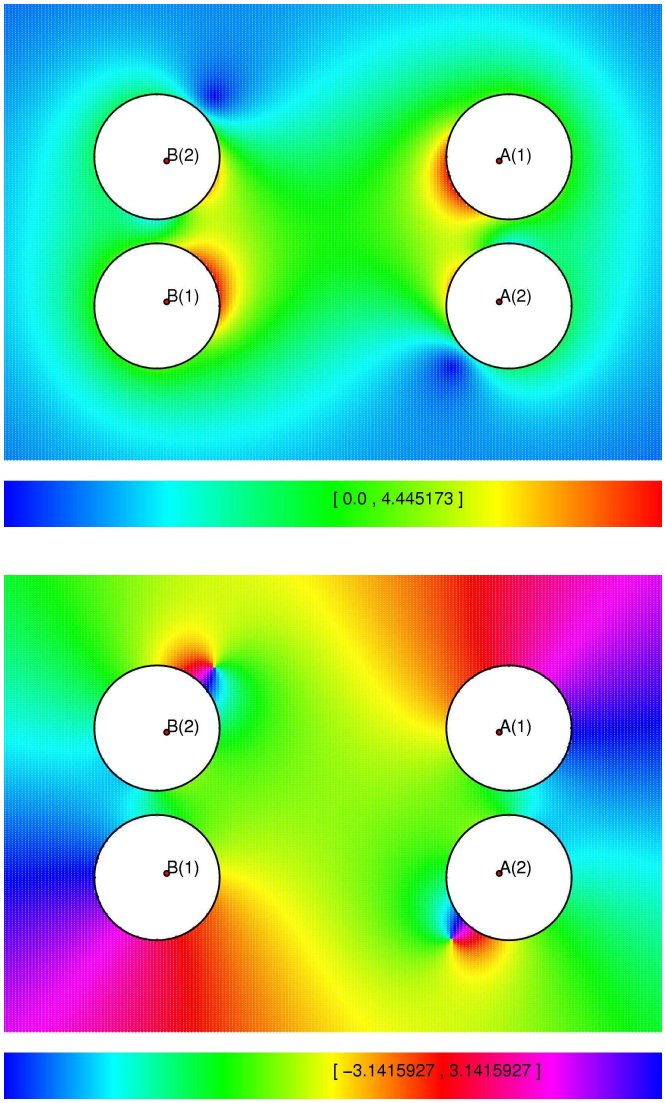


FIGURE 2.4.3. Absolute (top) and argument value (bottom) of ω_1 for the Wente torus with threefold symmetry. Symmetry faces $\overline{\omega_1(z)} = \omega_2(\bar{z})$.

if we cache the values

$$\rho_\star(\sigma) = \hat{\varepsilon}_\star(z;\sigma)\,\frac{2N-2}{2N-1}\,(2N-1)^{|\sigma|}\,k\left(z;\tilde{l}\right)^{p_\star}\ .$$

The value $k\left(z;\tilde{l}\right)$ is independent of σ and is therefore only computed once per evaluation and should be chached, because we can use it for all $\omega_n(z)$, $\Omega_n(z)$, $n = 1,\ldots,N$.

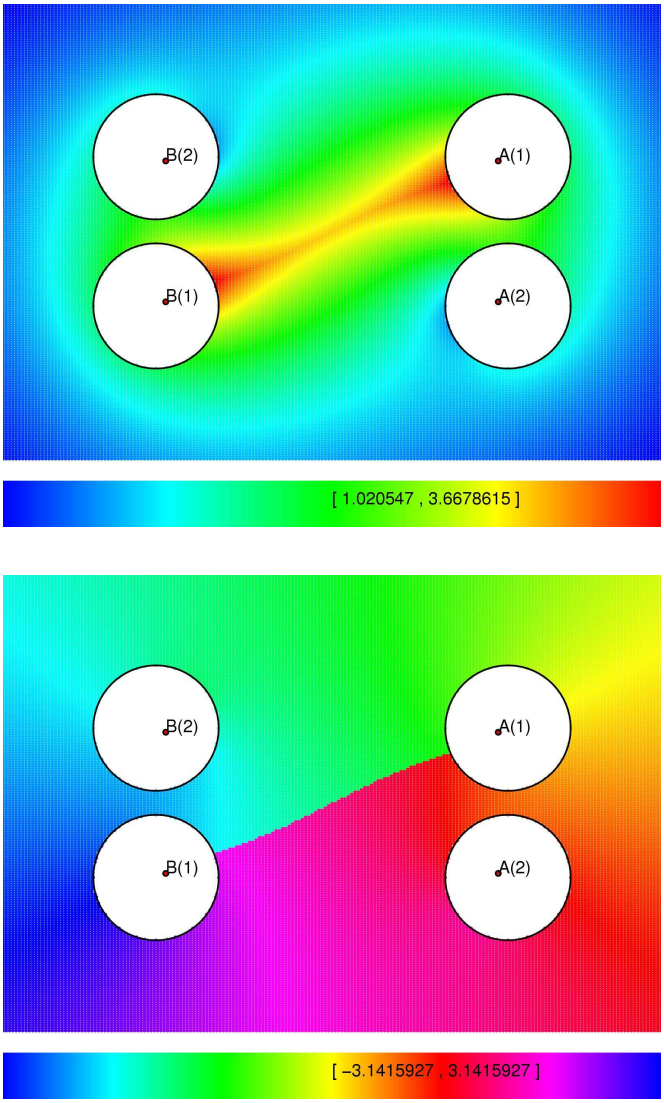


FIGURE 2.4.4. Absolute (top) and argument value (bottom) of Ω_1 for the Wente torus with threefold symmetry. Symmetry faces $\overline{\Omega_1(z)} = \Omega_2(\bar{z})$.

Appendix: Improving Estimates of Schottky Series

The disadvantage of the estimates yielding Lemma 2.3.2, 2.3.3, and 2.3.4, is that we replace $1/k(z;\sigma)$ by $1/k(z;|\sigma|)$ with

$$k(z;s) = \min_{\sigma \in G, |\sigma| = s} k(z;\sigma).$$

A poorly conditioned scenario arrises when we evaluate ω_n or Ω_n at points that are close or even on one of the isometric circles. Suppose $z \in C_n^{-i}$. Then $k(z;\sigma)$

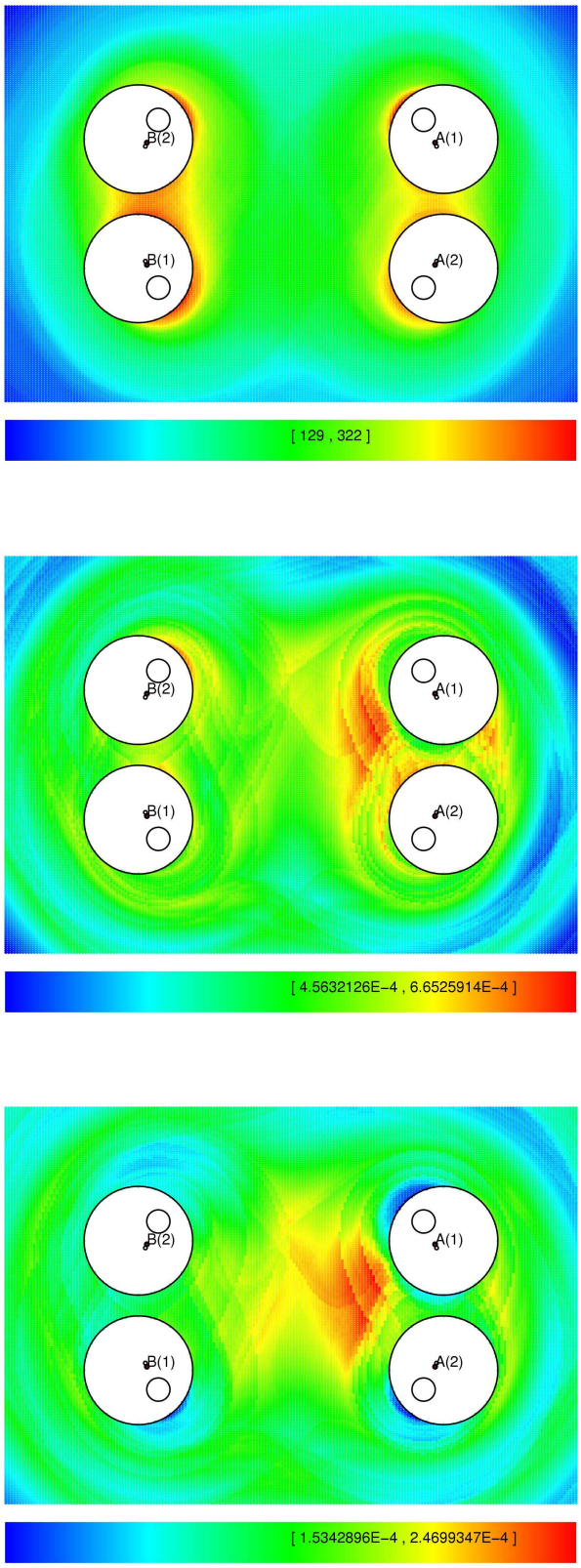


FIGURE 2.4.5. Results of Algorithm 2.4.3* and analysis of strictness of estimates: $\#S_{\Omega_1}(z;10^{-3})$ (top), $\hat{\varepsilon}_{\Omega_1}(z;S_{\Omega_1}(z;10^{-3}))$(center), and $d_{\Omega_1}(z;10^{-3}) \approx \left|\hat{\Omega}\right|_1(z;S_{\Omega_1}(z;10^{-6})) - \left|\hat{\Omega}\right|_1(z;S_{\Omega_1}(z;10^{-3}))$(bottom) for the Wente torus with threefold symmetry.

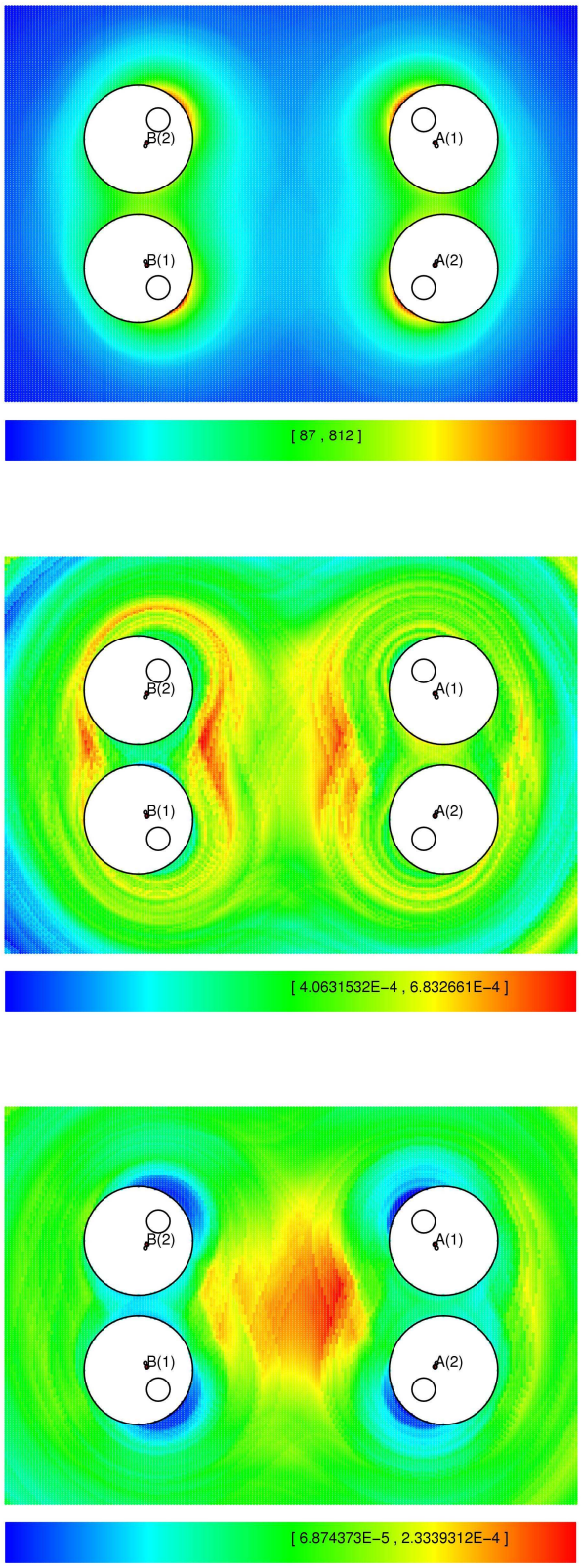


FIGURE 2.4.6. Results of Algorithm 2.4.3* and analysis of strictness of estimates: $\#S_{\omega_1}(z;10^{-3})$ (top), $\hat{\varepsilon}_{\omega_1}(z;S_{\omega_1}(z;10^{-3}))$(center), and $d_{\omega_1}(z;10^{-3}) \approx |\hat{\omega}|_1(z;S_{\omega_1}(z;10^{-6})) - |\hat{\omega}|_1(z;S_{\omega_1}(z;10^{-3}))$(bottom) for the Wente torus with threefold symmetry.

will be relatively small for all σ having σ_n^i as a suffix, i.e., $[\sigma] = [in, \ldots]$, and relatively large for all others. This is significant if the iso-classical Schottky group has small isometric circles compared to their distance. In such a case we almost lose a factor of $2N-1$ in the accuracy of our estimates. A straightforward approach to fix this is to disassemble $Cone^s(\sigma)$ into appropriate subsets, such as

$$\Lambda_m^j(\tau) = \left\{ \sigma \in Cone^s(\tau) \left| [\sigma] = \left[\sigma_m^j, \ldots\right] \right. \right\}.$$

Note that $Cone^s(\tau) = \dot{\cup}_{j,m} \Lambda_m^j(\tau)$. We use this to improve the estimates for ω_n:

$$\begin{aligned}
\sum_{\sigma \in Cone^s(\tau)} \left| H_n\left(z; \sigma^{-1}\right) \right| &\leq \sum_{\sigma \in Cone^s(\tau)} \frac{L\left(\sigma^{-1}, n\right)}{\left|\gamma_\sigma\right|^2} k(z;\sigma)^{-2} \\
&\leq \frac{L\left(\tau^{-1}, n\right)}{\left|\gamma_\tau\right|^2} \sum_{\sigma \in Cone^s(\tau)} \kappa_{|\tau|}^{-2(|\sigma|-|\tau|)} k(z;\sigma)^{-2} \\
&= \frac{L\left(\tau^{-1}, n\right)}{\left|\gamma_\tau\right|^2} \sum_{j,m} \sum_{\sigma \in \Lambda_m^j(\tau)} \kappa_{|\tau|}^{-2(|\sigma|-|\tau|)} k(z;\sigma)^{-2} \\
&\leq \frac{L\left(\tau^{-1}, n\right)}{\left|\gamma_\tau\right|^2} \sum_{j,m} k_m^j(z;|\sigma|)^{-2} \sum_{\sigma \in \Lambda_m^j(\tau)} \kappa_{|\tau|}^{-2(|\sigma|-|\tau|)},
\end{aligned}$$

where

$$k_m^j(z;l) = \min_{\sigma \in G, |\sigma| = l, [\sigma] = \left[\sigma_m^j, \ldots\right]} k(z;\sigma).$$

Obvious that $k_m^j(z;l) < k_m^j(z;l+k)$ just like $k(z,l) < k(z,l+k)$. We need to compute the series

$$Q_m^j(\sigma;q) = \sum_{\tilde{\sigma} \in \Lambda_m^j(\sigma)} q^{|\tilde{\sigma}|-|\sigma|}. \tag{2.4.3}$$

To count the number of elements of $\Lambda_m^j(\sigma)$ of a certain word length we denote these subsets by

$$\Lambda_m^j(\sigma, k) = \left\{ \tilde{\sigma} \in \Lambda_m^j(\sigma) \left| |\tilde{\sigma}| - |\sigma| = k \right. \right\}.$$

Now let $[\sigma] = [\ldots, in]$. We have

$$\begin{array}{rclrcll}
\#\Lambda_n^i(\sigma, k+1) &=& (2N-1)^k - \#\Lambda_n^{-i}(\sigma, k) & \#\Lambda_n^i(\sigma, 0) &=& 1 & \\
\#\Lambda_n^{-i}(\sigma, k+1) &=& (2N-1)^k - \#\Lambda_n^{i}(\sigma, k) & \#\Lambda_n^{-i}(\sigma, 0) &=& 0 & \\
\#\Lambda_m^j(\sigma, k+1) &=& (2N-1)^k - \#\Lambda_m^{j}(\sigma, k) & \#\Lambda_m^{j}(\sigma, 0) &=& 0 & m \neq n.
\end{array}$$

Rewriting the recursive formulas yields

$$\begin{array}{rclcll}
\#\Lambda_m^j(\sigma, k) &=& (-1)^{k-1} \sum_{\nu=0}^{k-1} (1-2N)^\nu &=& r_k & m \neq n \\
\#\Lambda_n^{-i}(\sigma, k) &=& r_k + \frac{(-1)^k - 1}{2} & & & \\
\#\Lambda_n^{i}(\sigma, k) &=& r_k + \frac{(-1)^k + 1}{2} & & & .
\end{array}$$

For $m \neq n$ we have:

$$\begin{aligned} Q_m^j(\sigma;q) &= \sum_{\tilde{\sigma}\in\Lambda_m^j(\sigma)} q^{|\tilde{\sigma}|-|\sigma|} \\ &= \sum_{k=0}^{\infty}(-1)^{k-1}\sum_{\nu=0}^{k-1}(1-2N)^{\nu}\, q^{\nu} \\ &= \sum_{\nu=0}^{\infty}(1-2N)^{\nu}\sum_{k=\nu+1}^{\infty}(-1)^{k-1}\, q^{\nu} \\ &= \sum_{\nu=0}^{\infty}(1-2N)^{\nu}\,\frac{-(-q)^{\nu+1}}{1+q} \\ &= \frac{q}{1+q}\frac{1}{1-(2N-1)\,q} \\ &= r(q) \end{aligned}$$

The $Q_m^j(\sigma;q)$ coincide for $m \neq n$. For the two cases with $m = n$ we obtain:

$$\begin{aligned} Q_n^{-i}(\sigma;q) &= r_-(q) &= r(q) - \frac{q}{1-q^2} \\ Q_n^{i}(\sigma;q) &= r_+(q) &= r(q) + \frac{1}{1-q^2}\,. \end{aligned}$$

By the definition of Λ_n^i we have

$$R(q) = \sum_{\sigma\in Cone^s(\tau)} q^{|\sigma|-|\tau|} = \sum_{\iota,\mu} Q_\mu^\iota(\sigma;q)\ .$$

Simularly,

$$R_m(q;\sigma) = \sum_{\sigma\in Cone_m^s(\tau)} q^{|\sigma|-|\tau|} = \sum_{\iota,\mu\neq m} Q_\mu^\iota(\sigma;q)\ ,$$

which yields the expression we have used in Lemma 2.3.3. Finally we have:

$$\sum_{\tilde{\sigma}\in Cone^s(\sigma)} |H_n(z;\sigma)| < \tilde{\varepsilon}_{\omega_n}(z;\sigma) \leq \hat{\varepsilon}_{\omega_n}(z;\sigma)\ ,$$

where

$$\tilde{\varepsilon}_{\omega_n}(z;\sigma) = \frac{L\left(\sigma^{-1},n\right)}{|\gamma_\sigma|^2}\sum_{\iota,\mu} k_\mu^\iota(z;|\sigma|)^{-2}\, Q_\mu^\iota\left(\sigma;\kappa_{|\sigma|}^{-2}\right)\ .$$

Simular results can be obtained for Ω_n and B_{nm}:

$$\begin{aligned}\tilde{\varepsilon}_{\Omega_n}(z;\sigma) &= |\sigma(B_n)-\sigma(A_n)|\sum_{\iota,\mu}\frac{Q_\mu^\iota\left(\sigma;\theta_{|\sigma|}^{-2}\right)}{k_\mu^\iota(z;|\sigma|)}\\ \tilde{\varepsilon}_{B_{nm}}(\sigma) &= |\sigma(B_n)-\sigma(A_n)|\sum_{\iota,\mu\neq m}\frac{Q_\mu^\iota\left(\sigma;\theta_{|\sigma|}^{-2}\right)}{k_\mu^\iota(A_m;|\sigma|)}+\frac{Q_\mu^\iota\left(\sigma;\theta_{|\sigma|}^{-2}\right)}{k_\mu^\iota(B_m;|\sigma|)}.\end{aligned}$$

The constants $\tilde{\varepsilon}_\star$ look much more complicated than the $\hat{\varepsilon}_\star$, but these constants are equally expensive computationally costs because we have to evaluate $k(z;\sigma)$ for all $\tilde{\sigma}\in G$ with $|\tilde{\sigma}|=|\sigma|$ for both. To take full advantage of these estimates we only have to replace each occurrence of $\hat{\varepsilon}_\star$ in Section 2.4 by $\tilde{\varepsilon}_\star$. Figures 2.4.7 and 2.4.8 demonstrate the results of our improvements for the examples of the Schottky data of He_3 and the Wente torus with threefold symmetry [**Hei95**, page 38]. We evaluate an integral of the first kind in the fundamental domain F and plot the size of $S_{\Omega_\star}(z;\star)$ generated by Algorithm 2.4.3 using the *optimized* error $\tilde{\varepsilon}_{\Omega_\star}$ and the *unoptimized* error $\hat{\varepsilon}_{\Omega_\star}$. The analysis shows that the optimized estimates yield significant smaller sets $S_{\Omega_\star}(z;\star)$ for those z close to the hot spots in the Schottky domain.

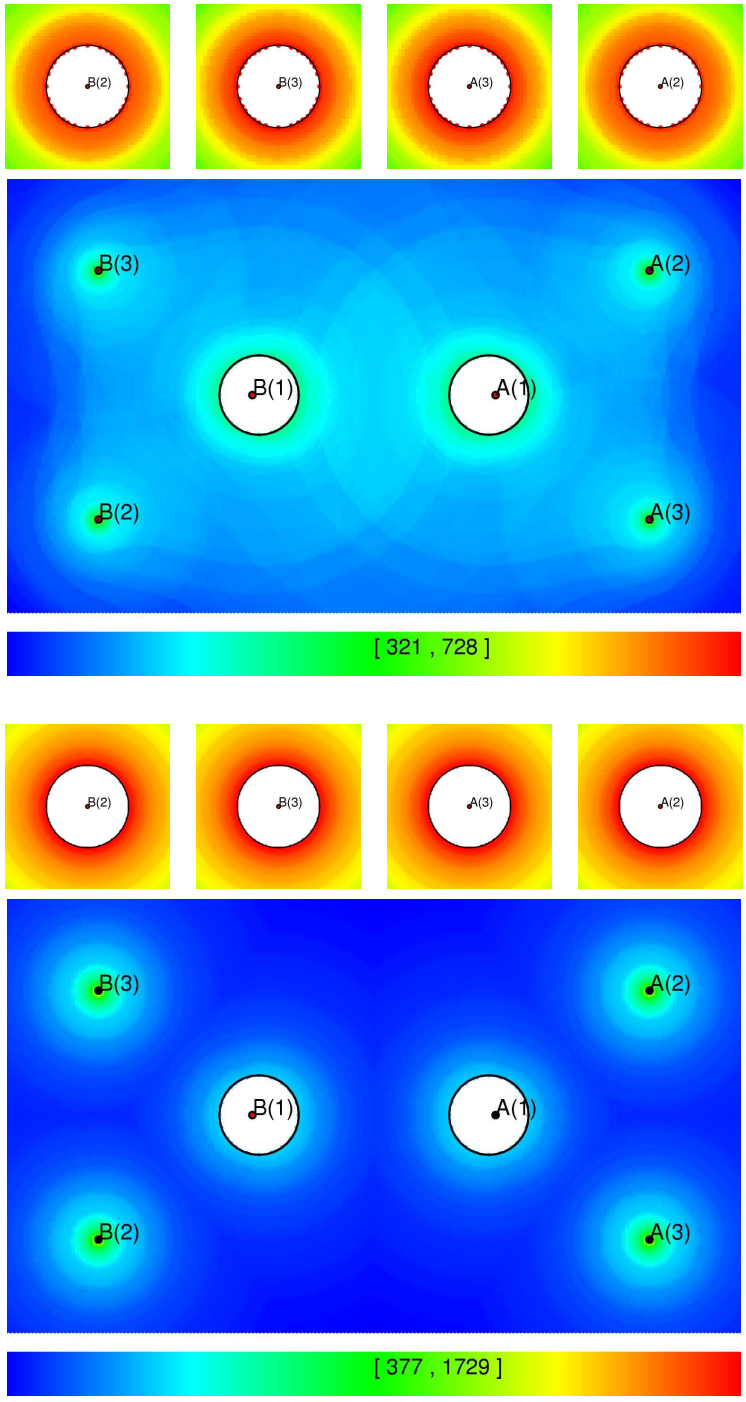


FIGURE 2.4.7. Size of $S_{\Omega_1}(z; 10^{-9})$ generated by Algorithm 2.4.3 using the *optimized* error $\tilde{\varepsilon}_{\Omega_1}$(top) and the *unoptimized* error $\hat{\varepsilon}_{\Omega_1}$(bottom) for the Schottky data of $\mathcal{H}e_3$.

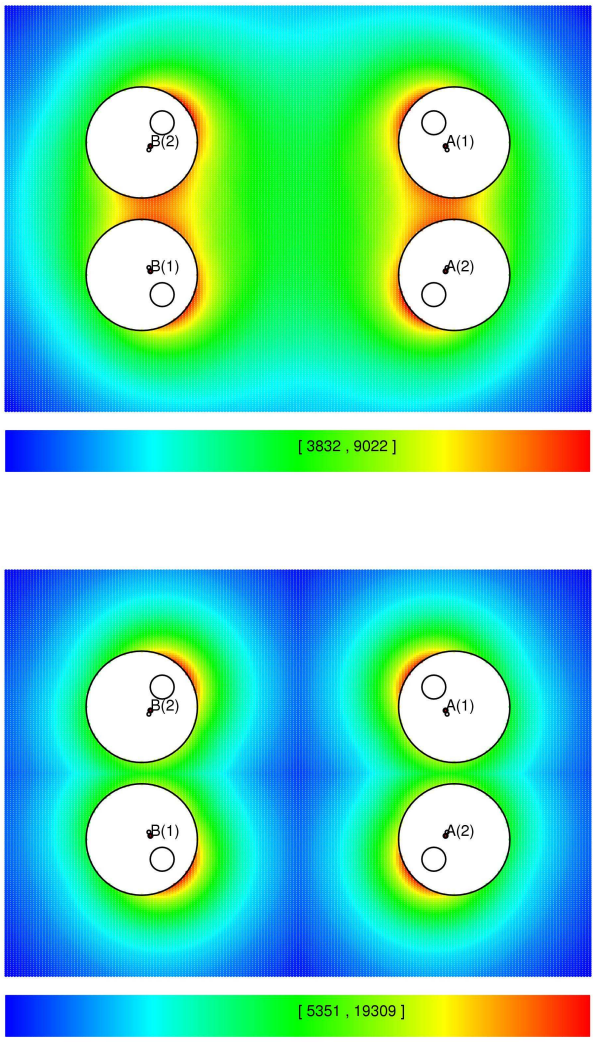


FIGURE 2.4.8. Size of $S_{\Omega_1}(z; 10^{-5})$ generated by Algorithm 2.4.3 using the *optimized* error $\tilde{\varepsilon}_{\Omega_1}$(top) and the *unoptimized* error $\hat{\varepsilon}_{\Omega_1}$(bottom) for the Wente torus with threefold symmetry.

CHAPTER 3

Computing Riemann Theta Functions

This chapter addresses the problem of computing values of Riemann theta functions. We give a brief summery of the approximation estimates and methods we developed in [**DHB$^+$04**], emphasizing the computational aspects. In particular, we compare the performance of pointwise and uniform approximation for different cases. We also present a new algorithm for the problem of selecting summation indices for the uniform approximation of the Riemanns theta function and discuss some improvements in the pointwise case. The algorithms are implemented in Java and belong to the *jtem* package *riemann* (Chapter 5).

3.1. Definition

The Riemann theta function is a complex-valued function of g complex variables and given by

$$\theta(z\,|B) = \sum_{n\in\mathbb{Z}^g} e^{\frac{1}{2}\langle n, B\cdot n\rangle + \langle z, n\rangle}, \tag{3.1.1}$$

where $z \in \mathbb{C}^g$ and B is a symmetric g-dimensional matrix with strictly negative definite real part[1]. The Fourier series representation (3.1.1) converges absolutely in z and B and therefore defines a homomorphic function in both entries. An overview of the theory of theta functions can be found in [**Mum83, Mum84, Mum91, BBE$^+$94**].

[1]There are many conventions for writing the theta functions, but they are the same up to rescaling the arguments.

3.2. Pointwise Approximation

To obtain good approximation formulas for (3.1.1) we have to determine the dominant terms of the infinite sum. For $z = x + iy$ and $B = X + iY$ we have

$$\begin{aligned}
\theta(z\,|B) &= \sum_{n\in\mathbb{Z}^g} e^{\frac{1}{2}\langle n,B\cdot n\rangle+\langle z,n\rangle} \\
&= \sum_{n\in\mathbb{Z}^g} e^{\frac{1}{2}\langle n,X\cdot n\rangle+\langle x,n\rangle}\, e^{\frac{i}{2}\langle n,Y\cdot n\rangle+i\langle y,n\rangle} \\
&= \sum_{n\in\mathbb{Z}^g} e^{\frac{1}{2}\left\langle n+X^{-1}\cdot x,X\cdot\left(n+X^{-1}\cdot x\right)\right\rangle-\frac{1}{2}\left\langle x,X^{-1}\cdot x\right\rangle}\, e^{\frac{i}{2}\langle n,Y\cdot n\rangle+i\langle y,n\rangle} \\
&= e^{-\frac{1}{2}\left\langle x,X^{-1}\cdot x\right\rangle}\,\theta_\Sigma(z\,|B)\ ,
\end{aligned}$$

where

$$\theta_\Sigma(z\,|B) = \sum_{n\in\mathbb{Z}^g} e^{\frac{1}{2}\left\langle n+X^{-1}\cdot x,X\cdot\left(n+X^{-1}\cdot x\right)\right\rangle}\, e^{\frac{i}{2}\langle n,Y\cdot n\rangle+i\langle y,n\rangle}$$

is the oscillating part, and $e^{-\frac{1}{2}\left\langle x,X^{-1}\cdot x\right\rangle}$ is of exponential growth. In most applications the theta functions are used to compute Abelian functions, which can be expressed as ratios of theta functions. The exponential factors usually cancel and it suffices to approximate θ_Σ.

The most dominant term in the oscillating sum is the one with summation index n that is closest to $V = -X^{-1}\cdot x$. To emphasize this, let $[V]$ be the vector with integer component closest to V, and $[[V]] = V - [V]$. By shifting the index n by $[X^{-1}\cdot x]$ we get:

$$\begin{aligned}
\theta_\Sigma(z\,|B) &= \sum_{n\in\mathbb{Z}^g} e^{\frac{1}{2}\left\langle n+\left[X^{-1}\cdot x\right]+\left[\left[X^{-1}\cdot x\right]\right],X\cdot\left(n+\left[X^{-1}\cdot x\right]+\left[\left[X^{-1}\cdot x\right]\right]\right)\right\rangle} \\
&\qquad\qquad \times\, e^{\frac{i}{2}\langle n,Y\cdot n\rangle+i\langle y,n\rangle} \\
&= \sum_{n\in\mathbb{Z}^g} e^{\frac{1}{2}\left\langle n+\left[\left[X^{-1}\cdot x\right]\right],X\cdot\left(n+\left[\left[X^{-1}\cdot x\right]\right]\right)\right\rangle} \\
&\qquad \times\, e^{\frac{i}{2}\left\langle n-\left[X^{-1}\cdot x\right],Y\cdot\left(n-\left[X^{-1}\cdot x\right]\right)\right\rangle+i\left\langle y,n-\left[X^{-1}\cdot x\right]\right\rangle}.
\end{aligned}$$

The dominating terms lie inside ellipsoids of the form

$$S_R(z,B) = \left\{ n\in\mathbb{Z}^g \,\middle|\, \frac{1}{2}\left\langle n+\left[\left[X^{-1}\cdot x\right]\right],(-X)\cdot\left(n+\left[\left[X^{-1}\cdot x\right]\right]\right)\right\rangle < R^2 \right\}.$$

Let $-X = T^T\cdot T$ be the Cholesky decomposition and let

$$\Lambda(z,B) = \left\{ \frac{1}{\sqrt{2}} T\cdot\left(n+\left[\left[X^{-1}\cdot x\right]\right]\right) | n\in\mathbb{Z}^g \right\},$$

then we can identify $S_R(z,B)$ as union of all lattice points of $\Lambda(z,B)$ inside the g-dimensional sphere:

$$S_R(z,B) = \{ v\in\Lambda(z,B)\,|\,||v|| < R\}\ .$$

This gives rise to an approximation of the Riemann theta function. Let us denote the pointwise approximation of $\theta_\Sigma(z\,|B)$ as

$$\theta_\Sigma^{S_R(z,B)}(z\,|B) = \sum_{n\in S_R} e^{-\frac{1}{2}||T\cdot(n+[[X^{-1}\cdot x]])||^2} \times e^{-\frac{i}{2}\langle n-[X^{-1}\cdot x],Y\cdot(n-[X^{-1}\cdot x])\rangle+i\langle y,n-[X^{-1}\cdot x]\rangle}.$$

Then we have

$$\theta(z\,|B) = e^{\frac{1}{2}\langle x,X^{-1}\cdot x\rangle}\lim_{R\to\infty}\theta_\Sigma^{S_R(z,B)}(z\,|B)\,,$$

which shows again that the Riemann theta functions are well-defined. Moreover, this representation provides error estimates.

THEOREM 3.2.1. [**DHB+04**] *(Pointwise approximation) Let ρ be the shortest distance between any two points of the lattice $\Lambda(z,B)$ and let R be the maximum of $\sqrt{g/2}+\rho/2$ and the real positive solution of*

$$\varepsilon = g\,2^{g-1}\Gamma\left(g/2,(R-\rho/2)^2\right)/\rho^g\,. \tag{3.2.1}$$

Then

$$\left|\theta_\Sigma^{S_R(z,B)}(z\,|B) - \theta_\Sigma(z\,|B)\right| < \varepsilon\,.$$

To apply this method we first compute the shortest lattice vector ρ of $\Lambda(z,B)$, which is independent of z. Then we solve for the prescribed error bound ε Equation 3.2.1 to get the radius R, and finally determine all elements of $S_R(z,B)$, using an algorithm first presented in [**Hei95**][2]. In the next section we give a variation of this method for solving a simular problem, that arises in the case of uniform approximation.

Finding the shortest lattice vector is a challenge. It is related to lattice reduction, one of the main tasks in the study of lattices. Lattice reduction searches for a standard form for the matrix of generating vectors t_i, $i=1,\ldots,g$, of a lattice, that is in a certain sense minimal and enables an easier treatment of problems intrinsic to the lattice, such as finding the shortest lattice vector. The problem of finding a standard form for the generating vectors, the "reduced basis", is known to be NP hard [**Val90**]. However, there is a polynomial hard approximation algorithm due to Lenstra, Lenstra, and Lovasz (the LLL algorithm). For $g\le 3$ the LLL algorithm delivers the exact solution and provides additionally upper and lower bounds for the shortest lattice vector ρ [**LLL82**][3] .

3.3. Uniform Approximation

The results of the previous section can be extended to obtain uniform approximations. The shape of the ellipsoid that determines the summation indices

[2] See *riemann.theta.LatticePointsInEllipsoid* for an implementation.

[3] $\hat\rho\,2^{(g-1)/2} \le \rho \le \hat\rho$, where $\hat\rho$ is the result of the LLL algorithm.

in the approximation of θ_Σ only depends on the period matrix B. The argument z only influences the center of the ellipsoid $c = [[-X^{-1} \cdot x]]$, which lies in a g-dimensional cube of volume 1, centered at the origin. Letting

$$U_R(B) = \left\{ n \in \mathbb{Z}^g \,\middle|\, \frac{1}{2} \langle n-c, (-X) \cdot (n-c) \rangle < R^2,\, c \in C^g \right\},$$

with $C^g = \left\{ c \in \mathbb{R}^g \,\middle|\, |c_i| < \frac{1}{2},\, i = 1, \ldots, g \right\}$, we see that

$$U_R(B) = \bigcup_{c \in C} S_R(c, B) .$$

Moreover, analogous to the pointwise approximation with $S_R(z,B)$ and $\theta_\Sigma^{S_R(z,B)}$, we denote by

$$\theta_\Sigma^{U_R(B)}(z\,|B) = \sum_{n \in S_R} \begin{array}{l} e^{-\frac{1}{2}\left|\left|T \cdot \left(n + [[X^{-1} \cdot x]]\right)\right|\right|^2} \\ \qquad \times\, e^{-\frac{i}{2}\left\langle n - \left[X^{-1} \cdot x\right], Y \cdot \left(n - \left[X^{-1} \cdot x\right]\right)\right\rangle + i\left\langle y, n - \left[X^{-1} \cdot x\right]\right\rangle}. \end{array}$$

the uniform approximation of $\theta_\Sigma(z\,|B)$. This leads to the following theorem.

THEOREM 3.3.1. [**DHB+04**] *(Uniform approximation) Let ρ be the shortest distance between any two points of the lattice $\Lambda(0,B)$ and let R be the maximum of $\sqrt{g/2} + \rho/2$ and the real positive solution of $\varepsilon = g\, 2^{g-1} \Gamma\left(g/2, (R-\rho/2)^2\right)/\rho^g$. Then*

$$\left| \theta_\Sigma^{U_R(B)}(z\,|B) - \theta_\Sigma(z\,|B) \right| < \varepsilon .$$

Clearly, the uniform approximation needs more terms than the pointwise one, and this penalty grows exponentially with the genus g. On the other hand, the determination of $S_R(z,B)$ is not cheap, and in many applications one needs to evaluate the Riemann theta function with constant period matrix many times[4]. Ultimately, the application determines whether uniform approximation is preferable to pointwise approximation (3.3.3).

For the uniform approximation we need to determine the set $U_R(B)$. This is a difficult task, so that we have to be satisfied with an approximation. The simplest one is the bounding box of $U_R(B)$, but we would incorporate an unacceptable penalty in higher dimensions. Bounding $U_R(B)$ by a larger ellipsoid is more practical, but this is still hard even if one does not want to determine the smallest.

3.3.1. Approximating $U_R(B)$. We use a variation of the algorithm presented in [**Hei95, DHB+04**], which is guaranteed to deliver an approximation enclosing all of $U_R(B)$. The algorithms solves a more general problem: Let $M = T^T \cdot T$ be the Cholesky decomposition of the positive definite $g \times g$-matrix M and let c the center of the cube

$$C^g_{c,\delta} \left\{ x \in \mathbb{R}^g \,||c_i - x| < \delta_i,\, i = 1, \ldots, g \right\} .$$

[4]Computing helicoids with handles can consume several million evaluations of the theta function .

The goal is to find all lattice points $n \in \mathbb{Z}^g$ with

$$||T\,(n-x)|| < R\,. \tag{3.3.1}$$

with $x \in C^g_{c,\delta}$. Since T is upper triangular, this implies $|T_{gg}\,(n_g - x_g)| < R$, or

$$c_g - \delta_g - \frac{R}{T_{gg}} < n_g < c_g + \delta_g + \frac{R}{T_{gg}}\,.$$

This gives a range of admssible values for n_g. For each we write

$$n = \begin{pmatrix} \hat{n} \\ n_g \end{pmatrix},\ c = \begin{pmatrix} \hat{c} \\ c_g \end{pmatrix},\ \delta = \begin{pmatrix} \hat{\delta} \\ \delta_g \end{pmatrix},\ x = \begin{pmatrix} \hat{x} \\ x_g \end{pmatrix},\ T = \begin{pmatrix} \hat{T} & \hat{t} \\ 0 & Tgg \end{pmatrix},$$

and

$$\begin{aligned} \tilde{R}^2 &= R^2 - \begin{cases} (n_g - c_g + \delta_g)^2 & n_g - c_g < -\delta_g \\ (n_g - c_g - \delta_g)^2 & n_g - c_g > +\delta_g \\ 0 & \text{otherwise} \end{cases} \\ \tilde{c} &= \hat{c} - \hat{T}^{-1}\hat{t}\,(n_g - c_g) \\ \tilde{d} &= \hat{d} + \left|\hat{T}^{-1}\hat{t}\right|\,\delta_g\,. \end{aligned}$$

Now $\left|\left|\hat{T}\,(\hat{n} - \tilde{x})\right|\right| < \tilde{R}$ with $\tilde{x} \in C^{g-1}_{\tilde{c},\tilde{\delta}}$ implies (3.3.1) because

$$\begin{aligned} & & ||T\,(n-x)||^2 &< R^2 \\ &\iff & T^2_{gg}\,(n_g - x_g)^2 + \left|\left|\hat{T}\,(\hat{n} - \hat{x}) + \hat{t}\,(n_g - x_g)\right|\right|^2 &< R^2 \\ &\Rightarrow & \left|\left|\hat{T}\left(\hat{n} - \left(\hat{x} - \hat{T}^{-1}\hat{t}\,(n_g - x_g)\right)\right)\right|\right|^2 &< R^2 - \min_{|c_g - x_g| < \delta_i} (n_g - x_g)^2\,. \end{aligned}$$

At this point we have reduced the problem to a similar problem in one dimension lower. We iterate this procedure[5]. For $\delta = 0$ the algorithm equals the original and delivers exactly the integer vectors inside the ellipsoid $||T\,(n-c)|| < R$.

Figure 3.3.1 shows the results of the above algorithm. The new algorithm picked 21 indices as an approximation for $U_R\,(B)$, which has 17 elements.

3.3.2. Evaluating uniform approximation. The evaluation method for uniform approximations takes fixed indices sets, because they enable precalculations. These precalculations naturally start with the determination of the approximation of the index set itself. The next optimization uses the semiperiodicity of theta functions:

$$\theta\,(z\,|B\,) = e^{-\frac{1}{2}\langle m, B\cdot m + z\rangle}\theta\,(z + 2\pi i\,n + B\cdot m\,|B\,)\ , \tag{3.3.2}$$

with $n, m \in \mathbb{Z}^g$. Using this transformation on $n = 0$ and $m = -\,[X^{-1}\cdot x]$ we get

$$\theta\,(z\,|B\,) = e^{\varphi(z|B\,)}\theta\,(\tau\,(z\,|B\,)\,|B\,)$$

[5]See *riemann.theta.LatticePointsForUniformApproximation* for an implementation.

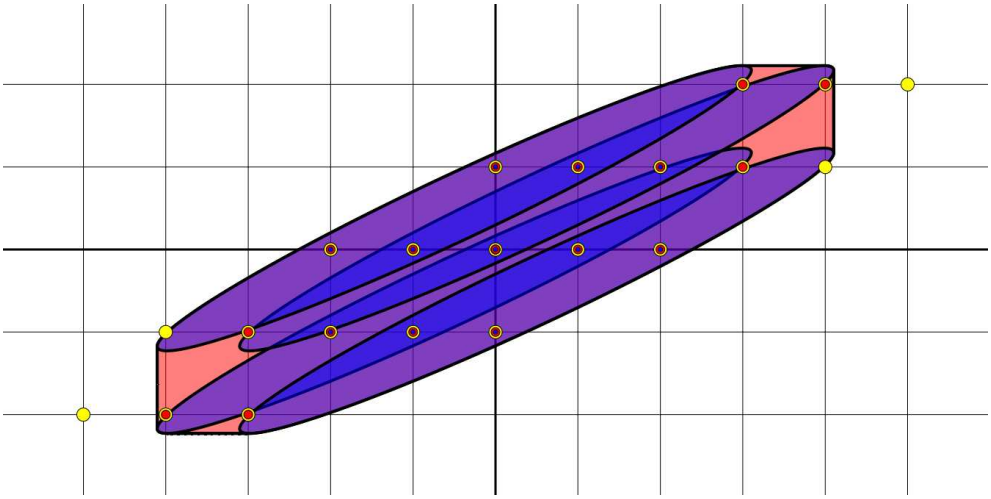


FIGURE 3.3.1. Results of algorithms to approximate $U_R(B)$, which elements are colored in red. Indices determined by our new algorithm are colored in yellow, indices enclosed by $\cup_{z=(\pm 1/2,\pm 1/2)} S_R(z|B)$ are colored in blue.

with $\varphi(z|B) = -\frac{1}{2}\langle [X^{-1}\cdot x],\, B\cdot[X^{-1}\cdot x]+z\rangle$, and $\tau(z|B) = z - B\cdot[X^{-1}\cdot x]$[6]. Since $[X^{-1}\cdot \Re\tau(z|B)] = 0$ the series for $\theta_\Sigma(\tau(z|B)|B)$ can be drastically simplified:

$$\theta_\Sigma(\tau(z|B)|B) = \sum_{n\in\mathbb{Z}^g} e^{-\frac{1}{2}\left|\left|T\cdot\left(n+X^{-1}\cdot\Re\tau(z|B)\right)\right|\right|^2 - \frac{i}{2}\langle n, Y\cdot n\rangle + i\langle \Im\tau(z|B), n\rangle}.$$

The values $\frac{i}{2}\langle n, Y\cdot n\rangle$ can be tabled in a precalculation step. Therefore evaluating a single term of the sum costs only $g^2/2+g/2+1$ multiplications and one evaluation of a complex exponential function. If the intermediate result satisfies

$$\frac{1}{2}\left|\left|T\cdot\left(n+X^{-1}\cdot\Re\tau(z|B)\right)\right|\right|^2 \geq R^2$$

we can discard the whole term. Thus awe practically compute the pointwise approximation $\theta_\Sigma^{S_R(\tau(z|B),B)}(\tau(z|B)|B)$.

3.3.3. Pointwise vs. uniform approximation. For helicoids of higher genus $g \geq 4$, the associated ellipsoids are rather eccentric[7]. Several eigenvalues λ_i, $i = 1,\ldots,g_0$ are smaller than R^2. Thus one can find 2^{g_0} ellipsoids $S_R(z_i, B)$, $z_i \in C^g$ that only have a small overlap. In these cases $\#U_R(B)$ is about 2^g times bigger than $\#S_R(z,B)$. If the eigenvalues become small compared to R^2, the situation gets worse. In all these cases the pointwise approximation is superior

[6]With *riemann.theta.TransformationPropertySupport* we supply a class performing this transformations.

[7]Even after applying the Siegel reduction algorithm, see Section 3.4.

g	PA	UA	PA/UA	PA	UA	PA/UA
1	4.5	7	0.6456	6.8	9	0.7608
2	14.0	25	0.5580	29.4	47	0.6247
3	37.0	111	0.3337	107.6	235	0.4577
4	88.5	435	0.2034	335.5	1013	0.3311
5	200.6	1649	0.1216	929.1	5013	0.1853
6	413.1	6655	0.0620	2428.5	22615	0.1073
7	974.5	33233	0.0293	6369.9	113493	0.0561
8	1187.7	85733	0.0138	10200.0	342379	0.0297
	$(\varepsilon = 10^{-5})$			$(\varepsilon = 10^{-15})$		

TABLE 1. Number of terms needed by the pointwise approximation (PA) and the uniform approximation (UA) for the theta function associated to the symmetric helicoids $\mathcal{H}e_g$ of genus $g = 1, \ldots, 8$ for error bounds $\varepsilon = 10^{-5}, 10^{-15}$.

g	PA	UA	PA/UA	PA	UA	PA/UA
1	0.03	0.02	1.5454	0.04	0.02	2.1695
2	0.05	0.03	1.8679	0.08	0.04	1.7913
3	0.06	0.15	0.3571	0.16	0.21	0.7740
4	0.48	0.40	1.2177	0.83	0.84	0.9916
5	0.33	1.92	0.1718	3.46	4.34	0.7972
6	2.09	6.05	0.3454	8.29	20.78	0.3989
7	6.07	32.04	0.1894	24.49	108.20	0.2263
8	4.70	79.72	0.0590	42.81	321.88	0.1329
	$(\varepsilon = 10^{-5})$			$(\varepsilon = 10^{-15})$		

TABLE 2. Average evaluation duration (milli seconds) of the pointwise approximation (PA) and the uniform approximation (UA) for the theta function associated to a squared lattice ($B = -2\pi I$) of genus $g = 1, \ldots, 8$ for error bounds $\varepsilon = 10^{-5}, 10^{-15}$.[9]

to the uniform and requires new ways of optimizing the algorithm computing the elements of $S_R(z, B)$. We translated the recursive algorighm from above into an iterative one, which allows to iterate conveniently and effectively through all indices[8]. For the period matrices of the symmetric helicoids $\mathcal{H}e_g$ of genus $g = 1, \ldots, 8$ (see 4.4.3) Tables 1 and 2 list the Riemann theta function for 1000

[8]See *riemann.theta.LatticePointsInEllipsoidIterator* for an implementation.

randomized arguments z_i and different error bounds ε. Table 1 compares the average size of $S_R(z_i, B)$ with the of $U_R(B)$ for $\varepsilon = 10^{-5}, 10^{-15}$. For the period matrix of the $\mathcal{H}e_8$ the uniform approximation (UA) needs 342379 terms for the error $\varepsilon = 10^{-15}$ compared to the average 10200.0 terms of the pointwise approximation (PA). Even the UA needs more then 30 times more terms then the PA, it is only less then 10 times slower, which means that determining the indices takes in this case more then 3 times longer then computing the term itself. Table 2 lists the average duration of an evaluation of the theta functions for the same setup as Table 1. It shows that for genus $g \geq 5$ the pointwise approximation was always faster, gaining a factor 2 in each dimension. For squared lattices, given by the period matrices $B_g = -2\pi I$, the ratio $\#S_R(z, B)/\#U_R(B)$ can be considered maximal. Once again, the pointwise approximation outperformed the uniform one for genus $g \geq 5$ because the main advantage of the uniform approximation, i.e., the tabling of precalculations, becomes a burden if the data exceeds the size of the caches of the processor.

3.4. The Modular Transformation Property

Unlike the semiperiodicity property (Section 3.4) the modular transformation property is a deep result in the theory of Riemann theta functions [**Igu72, Mum84**]. The Riemann matrix B usually originates as a period matrix of a Riemann surface, which incorporates a choice of a canonical intersection basis of its homology group. If $\mathcal{B} = \{a_1, b_1, \ldots, a_g, b_g\}$ is the canonical basis generating the period matrix B, then any canonical homology basis can be written as $\sigma(\mathcal{B}) = \mathcal{B}'$ with

$$\left.\begin{array}{rcl} a_i' & = & d_i^j a_j + b_i^j b_j \\ b_i' & = & a_i^j b_j + b_i^j a_j \end{array}\right\} \quad \text{where} \quad \sigma = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \in SP(2g, \mathbb{Z}) \ .$$

The modular group $SP(2g, \mathbb{Z})$ is a subgroup of $GL(2g, \mathbb{Z})$ and consists of all those elements that leaves the standard symplectic structure invariant, i.e.,

$$\sigma^T \begin{pmatrix} 0 & I \\ -I & 0 \end{pmatrix} \sigma = \begin{pmatrix} 0 & I \\ -I & 0 \end{pmatrix} .$$

Then the transformed basis $\sigma(\mathcal{B})$ generates the transformed period matrix

$$\sigma(B) = 2\pi i (aB + 2\pi i b)(cB + 2\pi i d)^{-1} \ .$$

This can be read as an equivalence relation on the set of Riemann matrices, which yields equivalence classes $[B] = \{\sigma B \,|\sigma \in SP(2g, \mathbb{Z})\}$.

Since Riemann theta functions are used to construct meromorphic functions, which do not depend on the choice of a homology basis, it is not surprising that theta functions possess a transformation property that relates equivalent period matrices:

$$\theta\left(z\,|B\right) = k\, e^{f_\sigma(z|B)}\, \theta\left(\sigma\left(z\,|B\right)|\sigma\left(B\right)\right) \tag{3.4.1}$$

with

$$\sigma\left(B\right) = 2\pi i\left(aB + 2\pi ib\right)\left(cB + 2\pi id\right)^{-1}\ ,$$

$$\begin{aligned} f_\sigma\left(z\,|B\right) &= R_\sigma\left(B\right)^T z - z^T A_\sigma\left(B\right) z + \delta_\sigma\left(B\right) \\ \sigma\left(z\,|B\right) &= H_\sigma\left(B\right) z + S_\sigma\left(B\right) \end{aligned}$$

and

$$\begin{aligned} R_\sigma\left(B\right) &= \pi i\left(cB + 2\pi id\right)^{-1} \operatorname{diag}\left(cd^T\right) \\ A_\sigma\left(B\right) &= \frac{1}{2}\left(cB + 2\pi id\right)^{-1} c \\ \delta_\sigma\left(B\right) &= \frac{1}{8}\operatorname{diag}\left(cd^T\right)^T \sigma\left(B\right) \operatorname{diag}\left(cd^T\right) \\ &\quad -\frac{g}{2}\log\left(2\pi i\right) - \log\sqrt{\det\left(cB + 2\pi id\right)} \\ H_\sigma\left(B\right) &= 2\pi i\left(\left(cB + 2\pi id\right)^{-1}\right)^T \\ S_\sigma\left(B\right) &= \pi i \operatorname{diag}\left(ab^T\right) + \frac{1}{2}\sigma\left(B\right) \operatorname{diag}\left(cd^T\right)\ . \end{aligned}$$

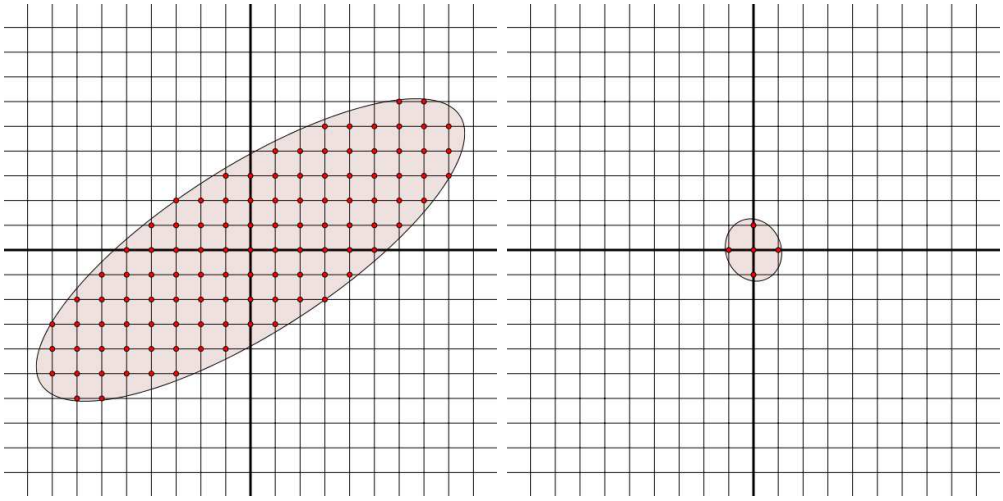
Finally k is an 8th-root of unity that can usually be ignored because one only deals with ratios of theta functions. The formulas above are implemented in *riemann.theta.ModularPropertySupport*, which provides a convenient frontend to this property of the theta functions (Section 5.6.2.2).

The importance of the modular transformation property lies in the fact that modular equivalent period matrices can behave completely different in the approximation process, e.g., some need many terms to approximate the oscillatory part of the Riemann theta function $\theta_\Sigma\left(z\,|B\right)$ and some only a few. An example: consider the period matrix

$$B = -\begin{pmatrix} \cos 1 & \sin 1 \\ -\sin 1 & \cos 1 \end{pmatrix}\begin{pmatrix} 3 & 0 \\ 0 & \frac{1}{3} \end{pmatrix}\begin{pmatrix} \cos 1 & \sin 1 \\ -\sin 1 & \cos 1 \end{pmatrix}\ ,$$

with a moderate eccentricity of 8/9. To reach an error of $\varepsilon = 10^{-12}$ for $z = 0$ one needs 107 terms associated to indices enclosed by the ellipse $S_R\left(0\,|B\right)$, with $R \approx 5.763$ (left hand side of Figure 3.4.1). The right hand side of the same figure shows the ellipse and the enclosed indices for the same error and the same position, but for the modular transformed period matrix

$$\sigma\left(B\right) \approx -\begin{pmatrix} \cos\alpha & \sin\alpha \\ -\sin\alpha & \cos\alpha \end{pmatrix}\begin{pmatrix} 34.235 & 0 \\ 0 & 45.525 \end{pmatrix}\begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix}\ ,$$



(a) before Siegel reduction (b) after Siegel reduction

FIGURE 3.4.1. Ellipses associated to the approximation of theta functions with modular equivalent period matrices.

with $\alpha = -1.181$ and

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

The ellipse $S_R(0\,|\sigma B)$, with $R \approx 7.189$, of the modular transformed period matrix only encloses 5 indices. The benefits of modular property transformation grow exponentially with the genus. The Siegel reduction algorithm computes the desired modular transformation. It finds a sequence of modular transformation matrices successively improving the period matrix [**Sie89, Hei95, DHB+04**]. The algorithm does not find an optimal solution, but it produces satisfactory results, as the above example shows. It also guarantees bounds for the reduced period matrix and the associated lattice.

THEOREM 3.4.1. *(Siegel reduction). For every Riemann matrix $B = X + iY$ Siegel's reduction algorithm delivers a modular transformation σ, such that $\sigma(B) = \hat{B} = \hat{X} + i\hat{Y}$, $\hat{X} = -T^T T$, with $\hat{X}(\hat{Y})$ the real (imaginary) part of $\hat{B}$ and T upper triangular, and*

1. $\left|\hat{Y}_{ij}\right| \leq \pi$, $i, j = 1, \ldots, g$,
2. *the length of the shortest lattice vector ρ of the lattice generated by the columns of T is bounded below by $\sqrt{\sqrt{3}\pi}$, and*

3. $\max\{|N_j| \,|\, |T \cdot N \leq R, R > 0,$ *fixed* $N \in \mathbb{Z}^g\}$ *has an upper bound that only depends on* g *and* R*.*

The proof is constructive, following the steps of Siegel's reduction algorithm[10] [**Sie89**]. In the elliptic case Siegel's algorithm always succeeds and finds the optimal result, which lies in the fundamental domain of the modulus.

3.5. Theta Functions with Characteristics

The multi-dimensional Fourier series

$$\theta\left[(\alpha,\beta)\right](z\,|B) = \sum_{n\in\mathbb{Z}^g} e^{\frac{1}{2}\left\langle n+\frac{1}{2}\alpha, B\cdot\left(n+\frac{1}{2}\alpha\right)\right\rangle + \langle z+\pi i\beta, n\rangle},$$

where $\alpha, \beta \in \mathbb{R}^g$, is called Riemann theta function with characteristics $[(\alpha,\beta)]$. Every point $e \in \mathbb{C}^g$ is written uniquely as

$$P = (\alpha,\beta)\begin{pmatrix} 2\pi i \\ B \end{pmatrix}.$$

Thus a theta function with characteristic $[(\alpha,\beta)]$ just uses a different reference point P of the Abelian torus C^g/Λ, $\Lambda = 2\pi i\, n + B \cdot m$, $n, m \in \mathbb{Z}^g$, in its Fourier series, instead of 0 like the ordinary one. But also theta functions with non-zero characteristics can be expressed in terms of the ordinary theta function:

$$\theta\left[(\alpha,\beta)\right](z\,|B) = e^{\frac{1}{8}\langle\alpha, B\alpha\rangle + \frac{1}{2}\langle z+\pi\, i\,\beta, \alpha\rangle}\theta\left(z + \pi\, i\,\beta + \frac{1}{2}\alpha B\,|B\right). \tag{3.5.1}$$

Usually one is only concerned with integer characteristics ($\alpha, \beta \in \mathbb{Z}^g$) and especially with those having only 0 and 1 as entries. The latter are called half-period characteristics. For integer characteristics (3.5.1) implies that

$$\begin{aligned} \theta\left[(\alpha+2\alpha', \beta+2\beta')\right](z\,|B) &= (-1)^{\langle\alpha',\beta'\rangle}\,\theta\left[(\alpha,\beta)\right](z\,|B), \\ \theta\left[(\alpha,\beta)\right](-z\,|B) &= (-1)^{\langle\alpha',\beta'\rangle}\,\theta\left[(\alpha,\beta)\right](z\,|B). \end{aligned}$$

This yields 2^{2g} different theta functions with half-period characteristics. If $\langle\alpha,\beta\rangle$ is even (odd) the characteristic is also called even (odd). $2^{g-1}(2^g+1)$ of the 2^{2g} different characteristics are even, $2^{g-1}(2^g-1)$ odd[11].

The modular transformation property can be expressed more natural by using theta function with characteristics, because the modular transformation of the period matrix induces a change of the Abelian torus, which can be further translated into a transformation of the characteristics:

$$\sigma(\alpha,\beta) = (\alpha,\beta)\,\sigma^{-1} + \left(\mathrm{diag}\left(cd^T\right), \mathrm{diag}\left(ab^T\right)\right).$$

[10] See *riemann.theta.SiegelReduction* for an implementation.

[11] *riemann.theta.ThetaCharIterator* is a support class that allows toggling through even and odd half-period characteristics (Section 5.6.3.1).

Then the modular transformation becomes

$$\theta\left[(\alpha,\beta)\right](z\,|B\,)=\frac{C\,e^{f_\sigma(z|B)}}{\sqrt{\det\left(cB+2\pi id\right)}}\,\theta\left[\sigma\left(\alpha,\beta\right)\right]\left(\sigma\left(z\,|B\,\right)|\sigma\left(B\right)\right)$$

with

$$\begin{aligned} f_\sigma\left(z\,|B\,\right) &= -\frac{1}{2}\sum_{i\leq j} z_i\,z_j\,\partial/\partial B_{ij}\log\det\left(cB+2\pi id\right) \\ \sigma\left(z\,|B\,\right) &= 2\pi i\left(\left(cB+2\pi id\right)^{-1}\right)^T z \end{aligned}$$

where C is a constant independent of z and B.

Unfortunately, the code does not benefit from this representation. The class *riemann.theta.ThetaWithChar* uses Equation 3.5.1 to reduce the computation of theta function with characteristics to the implementation of ordinary theta functions. In Section 5.6.3.1, we give a more complex example, showing the evaluation of products of spinors describing all helicoids with handles.

CHAPTER 4

Computing Helicoids With Handles

Bobenko defined helicoids of higher genus and gave explicit formulas in terms of theta functions.

Computing the Weierstrass spinors involves the evaluation of Riemann theta functions with characteristics and the calculation of differentials and integrals of different kinds for higher genus Riemann surfaces, which are not necessarily hyperelliptic.

To realize the minimal surface the spinors have to be integrated. Generic examples introduce translation periods, which have to be closed by a root-finder. The numerical computations may take weeks and cannot guaranteed to succeed it needs luck, because closed examples are rare.

We start with a brief introduction to the theory of helicoids with handles as it is presented in [**Bob98**]. In the second section we consider the problem of describing the data, which determines the helicoids using Schottky uniformization. The next section details the numerical construction for the actual problem and compares different approaches. In the final section we present various case studies as well as the examples we have found.

4.1. Mathematical Foundations of Helicoids With Handles

4.1.1. Helicoid and helicoidal end singularity. The starting point is the helicoid, the minimal immersion of the plane,

$$F(x+iy) = 2 \begin{pmatrix} \sinh x \, \sin y \\ -\sinh x \, \cos y \\ -y \end{pmatrix}$$

which is represented by the holomorphic Weierstrass data

$$\begin{aligned} g &= \exp z, \\ \zeta &= i \, dz \end{aligned}$$

To emphasize helicoidal asymptotics we change coordinates and choose $w = 1/z$ as local parameter which vanishes at the singularity. Then the helicoid is given as the minimal immersion of the punctured sphere $\overline{\mathbb{C}} \setminus \{0\}$ with the Weierstrass

data

$$\begin{aligned} g &= \exp(1/w), \\ \zeta &= \frac{-i}{w^2}\, dw. \end{aligned}$$

This classical data translates to the holomorphic spinors

$$\begin{aligned} a &= e^{-\frac{\pi i}{4}} e^{\frac{1}{2w}} \frac{\sqrt{dw}}{w}, \\ b &= e^{-\frac{\pi i}{4}} e^{\frac{-1}{2w}} \frac{\sqrt{dw}}{w}, \end{aligned} \tag{4.1.1}$$

on $\overline{\mathbb{C}} \setminus \{0\}$.

DEFINITION 4.1.1. We call a minimal immersion $F : \mathcal{R} = \mathrm{C} \setminus \{P_0\} \to \mathbb{R}^3$ a *helicoid of genus g* if C is of genus g and the Weierstrass spinors a, b are holomorphic on $\mathcal{R}$ and have the asymptotics

$$\begin{aligned} a &= e^{-\frac{\pi i}{4}} (1 + o(1)) e^{\frac{1}{2w}} \frac{\sqrt{dw}}{w}, \\ b &= e^{-\frac{\pi i}{4}} (1 + o(1)) e^{\frac{-1}{2w}} \frac{\sqrt{dw}}{w}, \qquad w \to 0 \end{aligned} \tag{4.1.2}$$

at the puncture P_0(where w is a holomorphic coordinate with $w(P_0) = 0$).

4.1.2. The Weierstrass data for helicoids with handles.

THEOREM 4.1.2. [**Bob98**] *Let $\mathcal{B} = \{a_1, b_1, \ldots, a_g, b_g\}$ be a canonical homology basis of the compact Riemann surface C of genus g, P_0 a point on C and w a chart that vanishes at P_0. Let $v_1, \ldots, v_g$ be the normalized ($\int_{a_i} v_i = 2\pi i$) differentials of first kind and $\int_{P_0}^{P} \Omega$ be the normalized ($\int_{a_i} \Omega = 0$) Abelian integral of second kind with the asymptotics*

$$\int_{P_0}^{P} \Omega = \frac{1}{w(P)} + o(1), \qquad P \to P_0$$

at P_0. Let V be the derivative of the Abel map at P_0 ($V_k dw = v_k(P_0)$). Then there generically (i.e., $\theta[\varepsilon](V/2) \neq 0$) exist for each theta characteristics ε unique spinors a, b, on $C \setminus \{P_0\}$, with Asymptotics (4.1.2) at the puncture P_0. These

spinors are given by the formulas

$$\begin{aligned} a(P) &= c_\delta e^{-\frac{\pi i}{4}} \frac{\theta[\varepsilon]\left(\int_{P_0}^{P} v - \frac{1}{2}V\right)}{\theta[\delta]\left(\int_{P_0}^{P} v\right)\theta[\varepsilon]\left(\frac{1}{2}V\right)} e^{\frac{1}{2}\int_{P_0}^{P}\Omega} h_\delta(P)\,, \\ b(P) &= c_\delta e^{-\frac{\pi i}{4}} \frac{\theta[\varepsilon]\left(\int_{P_0}^{P} v + \frac{1}{2}V\right)}{\theta[\delta]\left(\int_{P_0}^{P} v\right)\theta[\varepsilon]\left(-\frac{1}{2}V\right)} e^{-\frac{1}{2}\int_{P_0}^{P}\Omega} h_\delta(P)\,, \end{aligned} \tag{4.1.3}$$

where δ is a non-singular ($D\theta[\delta]_{|0} \neq 0$) odd theta characteristics and all integration paths coincide. The holomorphic spinor h_δ and normalization constant c_δ are given by

$$h_\delta = \sqrt{D\theta[\delta]_{|0} \cdot v} \quad \textit{and} \quad c_\delta = \sqrt{D\theta[\delta]_{|0} \cdot V}.$$

The pairing $[\mathcal{B}, \varepsilon]$ represents a quadratic form

$$s : H_1(C) \to \mathbb{Z}_2\,,$$

that is

$$s(\gamma_1 + \gamma_2) = s(\gamma_1) + \gamma_1 \circ \gamma_2 + s(\gamma_2)$$

and taking the values of the characteristics on the basis cycles, i.e.,

$$s(a_i) = \alpha_i \text{ and } s(b_i) = \beta_i\,,$$

where $\varepsilon = [\alpha, \beta]$. Let σ be an element of the modular group $\mathrm{Sp}(2g, \mathbb{Z})$ and $\mathcal{B}' = \sigma(\mathcal{B})$ (Section 3.4). Then the pairing $[\mathcal{B}', \varepsilon']$ represents the same quadratic form as $[\mathcal{B}, \varepsilon]$ iff $\varepsilon' = \sigma(\varepsilon)$ (Section 3.5). Applying these transformations to the Weierstrass data in Theorem 4.1.2 reveals that it only depends on the quadratic form s.

We call two pairings $[\mathcal{B}, \varepsilon]$ and $[\mathcal{B}', \varepsilon']$ equivalent iff they define the same quadratic form. The quadratic form s has a simple but important geometric meaning. Let $M(\gamma) \in \mathbb{Z}$ be the number of twists of the normal field along the contour γ. Then

$$\begin{aligned} M(a_i) &= s(a_i) \\ M(b_i) &= s(b_i) \end{aligned} \quad \mod 2$$

We call two local parameters w and $\tilde{w}$ at P_0 equivalent if they vanish in P_0, i.e., $w(P_0) = \tilde{w}(P_0) = 0$, and $d\tilde{w}/dw = 1$. The corresponding equivalence class $[w]$ can be identified as an element of the tangent space $T_{P_0}C$. Therefore equivalent local parameters generate the same Weierstrass data.

Thus the data $\{C, P_0, [w], s\}$ uniquely determine Weierstrass data with Asymptotics (4.1.2) at the puncture P_0, but it only defines a helicoid if this data leads

to a minimal immersion $F : \mathcal{R} = C \setminus \{P_0\} \to \mathbb{R}^3$. This is the case only if all the translation periods vanish, i.e.,

$$\text{Re} \int_\gamma \left(-a^2 + b^2, i\left(a^2 + b^2\right), 2ab\right) = 0 \quad \forall \gamma \in \mathcal{B} \tag{4.1.4}$$

4.1.3. Periodicity conditions. The last subsection outlines our numerical analysis. First we must be able to evaluate for given data $\{C, P_0, [w], s\}$ the related helicoidal Weierstrass data and integrate it along a homology basis. Generically this will deliver translation periods that we will try to annihilate by varying the data. We need to parameterize the data $\{C, P_0, [w], s\}$, but we can only hope to succeed if the number of free parameters exceeds or equals the number of constraints given by Equation 4.1.4. The moduli space of compact Riemann surfaces of genus $g \geq 2$ has complex dimension $3g - 3$. P_0 and $[w]$ add another complex dimension each, so that we have

$$dim_{\mathbb{R}} \{C, P_0, [w], s\} = 6g - 2,$$

which is not enough. But we know at least one example: the Karcher-Hoffmann-Wei genus 1 helicoid [**HWK93**]. It possesses a normal symmetry, i.e., 180° rotation about a line that intersects it orthogonally. This can be translated into a symmetry condition on the data that reduces the number of free parameters to the number of periodicity conditions.

Let $C \to C_0$ be a ramified double covering of genus $g = 2g_0 + N$ of a compact Riemann surface C_0 of genus g_0 with $2N + 2$ branch points and let $\pi : C \to C$ be the conformal involution that exchanges the two sheets of that covering.

Definition 4.1.3. The data $\{C, P_0, [w], s\}$ is called *admissible* if
1. $C \to C_0$ is a two-sheeted ramified covering as described above,
2. P_0 is a branch point of the covering; $\pi(P_0) = P_0$,
3. $\pi^* [w] = -[w]$, and
4. $\pi^* s = s$.

Theorem 4.1.4. [**Bob98**] *Minimal immersions with admissible data* $\{C, P_0, [w], s\}$ *possess normal symmetry, i.e., a* 180° *rotation about the first coordinate axis*

$$(F_1, F_2, F_3) \to (F_1, -F_2, -F_3).$$

A Riemann surface from the definition above admits a canonical basis

$$\mathcal{B} = \left\{a_1, b_1, \ldots, a_{g_0}, b_{g_0}, \tilde{a}_1, \tilde{b}_1, \ldots, \tilde{a}_{g_0}, \tilde{b}_{g_0}, \hat{a}_1, \hat{b}_1, \ldots, \hat{a}_N, \hat{b}_N\right\}$$

of $H_1(C, \mathbb{Z})$ such that $a_1, b_1, \ldots, a_{g_0}, b_{g_0}$ is a canonical basis of $H_1(C_0, \mathbb{Z})$ and

$$\begin{array}{rclcrcl} \pi(a_i) & = & -\tilde{a}_i & , & \pi(b_i) & = & -\tilde{b}_i \\ \pi(\hat{a}_j) & = & -\hat{a}_j & , & \pi\left(\hat{b}_j\right) & = & -\hat{b}_j. \end{array} \tag{4.1.5}$$

For such a basis $\mathcal{B}$ the constraint on the spin in Definition 4.1.3 translates to a condition on the theta characteristics:

$$\varepsilon = \left[(\alpha, \tilde{\alpha}, \hat{\alpha}) , \left(\beta, \tilde{\beta}, \hat{\beta} \right) \right]$$

with $\alpha = \tilde{\alpha}$ and $\beta = \tilde{\beta}$.

THEOREM 4.1.5. [**Bob98**] *The space of admissible data* $\{C, P_0, [w], s\}$ *has real dimension*

$$6g_0 + 4N .$$

It coincides with number of independent period conditions. For a canonical basis $\mathcal{B}$ *as described above*

$$\begin{aligned} Re \int_\gamma (-a^2 + b^2, i(a^2 + b^2), 2ab) &= 0 \quad with \quad \gamma = a_1, b_1, \ldots, a_{g_0}, b_{g_0} , \\ Re \int_\gamma (i(a^2 + b^2), 2ab) &= 0 \quad with \quad \gamma = \hat{a}_1, \hat{b}_1, \ldots, \hat{a}_N, \hat{b}_N \end{aligned}$$

define a set of independent period conditions.

The helicoid of Karcher-Hoffmann-Wei has another symmetry, a 180° rotation about a line that lies on it. This symmetry corresponds to an antiholomorphic involution of the Riemann surface.

Let $\tau : C \to C$ be an anti-holomorphic involution of the Riemann surface with exactly one real oval with the puncture P_0 on it.

DEFINITION 4.1.6. The data $\{C, P_0, [w], s\}$ is called *symmetric* if
1. $\tau : C \to C$ is an anti-holomorphic involution as described above,
2. P_0 lies on the real oval of τ; $\tau(P_0) = P_0$,
3. $\tau^* [w] = [\bar{w}]$, and
4. $\tau^* s = s$.

THEOREM 4.1.7. *Minimal immersions with symmetric data* $\{C, P_0, [w], s\}$ *possess symmetry, i.e. a* 180° *rotation about the vertical axis*

$$(F_1, F_2, F_3) \to (-F_1, -F_2, F_3) .$$

We choose again a special canonical basis $a_1, b_1, \ldots, a_g, b_g$ of $H_1(C, \mathbb{Z})$ such that

$$\begin{aligned} \tau(a_i) &= \pm a_i , \\ \tau(b_i) &= \mp b_i + H_i^j a_j , \end{aligned} \tag{4.1.6}$$

with $H_i^j \in \mathbb{Z}_2$. We will call such a basis *positive symmetric* if $\tau(a_i) = +a_i$ and *negative symmetric* otherwise.

THEOREM 4.1.8. *The space of symmetric data* $\{C, P_0, [w], s\}$ *has real dimension* $3g - 1$. *The number of independent periodicity conditions is* $3g$. *For a*

positive symmetric basis $\mathcal{B}_+$, the integrals

$$\begin{aligned} Re\int_{a_i}\left(-a^2+b^2, i\left(a^2+b^2\right)\right) &= 0\,, \\ Re\int_{b_i}(2ab) &= 0\,. \end{aligned}$$

(with $i=1,\ldots,g$) define a set of $3g$ independent period conditions. Similarly, for a negative symmetric basis $\mathcal{B}_-$ the set of $3g$ independent period conditions is

$$\begin{aligned} Re\int_{a_i}(2ab) &= 0\,. \\ Re\int_{b_i}\left(-a^2+b^2, i\left(a^2+b^2\right)\right) &= 0\,. \end{aligned}$$

This shows that an immersion with symmetry τ must have additional symmetries, motivating the following conjecture.

CONJECTURE 4.1.9. [**Bob98**]*Any immersed minimal surface of finite topology with one helicoidal end is invariant under a* 180° *rotation about a line orthogonal to the surface.*

For admissible and symmetric data $\{C, P_0, [w], s\}$, $\tau\circ\pi$ generates an other symmetry, a 180° rotation about the second coordinate axis of the minimal immersion:

$$(F_1, F_2, F_3) \to (-F_1, F_2, -F_3)\,.$$

The dimension of the space of admissible symmetric data is $3g_0+2N$ and therefore half the dimension of the general space of admissible data.

The spectrum of admissible data has two extremes related to the covering $C \to C_0$. One end is formed by the hyperelliptic examples ($g_0=0$, $N=g$), which includes the genus 1 helicoid of Karcher-Hoffmann-Wei. Here one expects the handles to be located along the normal line of symmetry. The other contains coverings with only two branch points ($g_0=g$, $N=0$). In this case, the normal symmetry line intersects the surface only at the branch point $P\neq P_0$ and one should expect vertical displaced handles. Our numerical analysis only found hyperelliptic examples. In all the other cases we examined, the numerics strongly suggest that these surfaces do not exist.

CONJECTURE 4.1.10. *Any immersed minimal surface of finite topology with one helicoidal end is a hyperelliptic Riemann surface.*

4.2. Schottky Uniformization of Helicoids with Handles

In the previous section we saw that for admissible data $\{C, P_0, [w], s\}$ with $s=[\mathcal{B}, \varepsilon]$ we may hope to find examples with closed periods. Cutting C along

simple disjoint loops

$$v_1, \ldots, v_{g_0}, \widetilde{v}_1, \ldots, \widetilde{v}_{g_0}, \widehat{v}_1, \ldots, \widehat{v}_N \tag{4.2.1}$$

homologically equivalent to the a-cycles of the admissible homology basis $\mathcal{B}$ yields a plane region, which, according to a classical theorem [**For29**], can be conformally mapped to the fundamental domain F of a corresponding Schottky group G. The boundary of F consists of the images of the Loops (4.2.1) forming pairs of curves

$$(C_1, C_1'), \ldots, (C_{g_0}, C_{g_0}'), (\widetilde{C}_1, \widetilde{C}_1'), \ldots, (\widetilde{C}_{g_0}, \widetilde{C}_{g_0}'), (\widehat{C}_1, \widehat{C}_1'), \ldots, (\widehat{C}_N, \widehat{C}_N'), \tag{4.2.2}$$

which are mapped onto each other by the generators

$$\sigma_1, \ldots, \sigma_{g_0}, \widetilde{\sigma}_1, \ldots, \widetilde{\sigma}_{g_0}, \widehat{\sigma}_1, \ldots, \widehat{\sigma}_N$$

of the Schottky group G. If the isometric circles of the purely loxodromic generators and their inverses are external to one another the loops defined in Equation 4.2.1 can be altered such that the pairs of curves given in Equation 4.2.2 become the isometric circles.

The numerical methods we described in Chapter 2 are restricted to iso-classical Schottky uniformization. From now on let G be a iso-classical Schottky group and let the curves in Equation 4.2.2 be its isometric circles. The holomorphic involution π on C defines then a holomorphic involution π_F on F that maps a plane region bounded by a finite number of complete circles onto itself; π_F is linear fractional [**For29**, p. 282]. By construction, we have

$$\begin{aligned} \sigma_i \, \pi_F &= \pi_F \, \widetilde{\sigma}_i, \\ \widehat{\sigma}_j \, \pi_F &= \pi_F \, \widehat{\sigma}_j, \end{aligned} \tag{4.2.3}$$

Since π_F is an involution on the whole of $\mathbb{C}P^1$ it can be transformed into the 180° rotation about the origin by conjugation with a linear transformation ρ, i.e.,

$$\rho \, \pi_F \, \rho^{-1} = -id \, .$$

The linear transformation ρ is only defined up to scaling and an inversion at the unit circle, which we will use later. Transforming the classical Schottky group G by conjugation with ρ,

$$G \, \rightarrow \, \rho G \rho^{-1}, \quad F \, \rightarrow \, \rho \, (F)$$

we achieve a different Schottky uniformization of the Riemann surface C, which may not be classical any more. Then Equations 4.2.3 translate to

$$\begin{aligned} \sigma_i \, (z) &= -\widetilde{\sigma}_i \, (-z) \, , \\ \widehat{\sigma}_j \, (z) &= -\widehat{\sigma}_j \, (-z) \, . \end{aligned}$$

The admissible data not only consists of the Riemann surface C admitting a holomorphic involution π, but also of an equivalence class $[w]$ of local parameters at a specific point P_0, which is one of the $2N + 2$ fixed points of the involution π. Each of the $\widehat{v}_1, \ldots, \widehat{v}_N$ loops touches exactly two fixed points; without loss of

generality we could have chosen these loops such that none of them touches the point P_0. Thus P_0 is conformally mapped to one of the two fixed points of π_F, which is, after conjugation, either 0 or ∞. The other $2N$ fixed points are given by $\widehat{\sigma}_j(z) = -z$,

$$\left(z - \widehat{A}_j\right)^2 = \widehat{\mu}_j \left(z + \widehat{B}_j\right)^2, \;\; j = 1, \ldots, N$$

and lie in pairs on the isometric circles $\widehat{C}_j$ and therefore also on $\widehat{C}'_j$. The transformation ρ is uniquely defined if we demand that P_0 be sent to ∞ and that $[w] = [1/z]$. The uniformization data

$$\begin{aligned} S = \{ & A_1, B_1, \mu_1, \quad \ldots \quad , A_{g_0}, B_{g_0}, \mu_{g_0}, \\ & \widetilde{A}_1, \widetilde{B}_1, \widetilde{\mu}_1, \quad \ldots \quad , \widetilde{A}_{g_0}, \widetilde{B}_{g_0}, \widetilde{\mu}_{g_0}, \\ & \widehat{A}_1, \widehat{B}_1, \widehat{\mu}_1, \quad \ldots \quad , \widehat{A}_N, \widehat{B}_N, \widehat{\mu}_N \; \} \end{aligned} \tag{4.2.4}$$

with

$$A_i = -\widetilde{A}_i, \quad B_i = -\widetilde{B}_i, \quad \mu_i = \widetilde{\mu}_i, \quad \widehat{A}_j = -\widehat{B}_j \tag{4.2.5}$$

defines canonically admissible data apart from the spin structure s . By construction, the Schottky basis has the symmetry required in Equation 4.1.5. The number of free real parameters of S is $6g_0 + 4N$ and therefore coincides with the dimension of the space of admissible data. Thus we can parameterize all of the surfaces we are able to compute with means presented in Section 2.

A similar argument holds for symmetric data. The anti-holomorphic involution can be chosen as $\tau(z) = \bar{z}$. This yields the uniformization data

$$S = \{A_1, B_1, \mu_1, \ldots, A_g, B_g, \mu_g\}$$

with

$$A_i = \overline{B_j}, B_i = \overline{A_j}, \mu_i = \overline{\mu_j} \iff \tau \circ \sigma_i = \sigma_j^{-1} \circ \tau, \tag{4.2.6}$$

or alternatively

$$A_i = -\overline{A_j}, B_i = -\overline{B_j}, \mu_i = \overline{\mu_j} \iff \tau \circ \sigma_i = \sigma_j \circ \tau \tag{4.2.7}$$

with $i, j = 1, \ldots, g$. The number of free parameters of S coincide again with the dimension of the space of symmetric data. The Schottky data with Constraints (4.2.6) have an associated positive symmetric homology basis, and while Constraints (4.2.7) generate a negative symmetric basis.

To achieve positive (negative) symmetric admissible data the Constraints (4.2.5) and (4.2.6) (resp. (4.2.7)) have only to be combined in the obvious way. All the a-cycles of any Schottky basis associated to symmetric admissible Schottky data fulfill the Constraints (4.1.5) for an admissible basis as well as the Constraints (4.1.6) for a symmetric basis. The b-cycles of such a Schottky basis can be chosen so that they fulfil both requirements when we allow them to intersect in the origin.

LEMMA 4.2.1. *For positive symmetric admissible Schottky data and associated basis $\mathcal{B}_+$ as described above,*

$$\begin{aligned} Re\int_\gamma (-a^2+b^2, i(a^2+b^2)) &= 0 \quad with \quad \gamma = a_1, \ldots, a_{g_0}\,, \\ Re\int_\gamma (i(a^2+b^2)) &= 0 \quad with \quad \gamma = \hat{a}_1, \ldots, \hat{a}_N\,, \\ Re\int_\gamma (2ab) &= 0 \quad with \quad \gamma = b_1, \ldots, b_{g_0}, \hat{b}_1, \ldots, \hat{b}_N\,. \end{aligned}$$

define a set of $3g_0 + 2N$ independent period conditions. For negative symmetric admissible Schottky data and associated basis B_- the set of $3g_0+2N$ independent period conditions is

$$\begin{aligned} Re\int_\gamma (-a^2+b^2, i(a^2+b^2)) &= 0 \quad with \quad \gamma = b_1, \ldots, b_{g_0}, \\ Re\int_\gamma (i(a^2+b^2)) &= 0 \quad with \quad \gamma = \hat{b}_1, \ldots, \hat{b}_N \\ Re\int_\gamma (2ab) &= 0 \quad with \quad \gamma = a_1, \ldots, a_{g_0}, \hat{a}_1, \ldots, \hat{a}_N\,. \end{aligned}$$

4.3. Numerical Construction

4.3.1. Numerical evaluation of spinors. The Spinors (4.1.3) require the evaluation of Riemann theta functions. The methods described in [**DHB+04**] are implemented in the *jtem*[1] project *riemann*[2], enabling us to evaluate Riemann theta functions effectively within prescribed error bounds.

In Chapter 2 we dealt with the evaluation of Abelian differentials and integrals on Schottky uniformized Riemann surfaces. This, too, is part of the project *riemann*.

The only entity that is not covered by the above is the normalized Abelian integral of second kind $\int_{P_0}^{P}\Omega$ with asymptotics

$$\int_{P_0}^{P}\Omega = \tfrac{1}{w(P)} + o\,(1)\,, \qquad P \to P_0\ .$$

P_0 had be chosen to be ∞. Because of the holomorphic involution $\pi : z \mapsto -z$ we have

$$\int_\infty^z \Omega = \int_0^z \Omega + \int_\infty^0 \Omega = \int_0^z \Omega + k\pi i.$$

The generators of an admissible Schottky group can always be chosen such that $k = g$ (or $k = g \mod 2$). The differential of second kind only appears as the argument of the exponential function, so that $k\pi i$ at most generates a sign, which will evantually cancel when we compute the Weierstrass forms $-a^2+b^2, i(a^2+b^2)\,, 2ab$. By defining

$$\int_0^z \Omega = \sum_{\sigma\in G} \sigma\,(z) - \sigma\,(0)$$

[1] http://www.jtem.de

[2] http://www.jtem.de/riemann

one can verify that $\int_0^z \Omega$ and $\int_\infty^z \Omega$ have the desired asymptotics:

$$\begin{aligned}\int_0^z \Omega &= z + \sum_{\sigma\in G,\sigma\neq id} \sigma(\infty) - \sigma(0) + o(1) \\ &= z + \sum_{\sigma\in G,\sigma\neq id} \frac{\alpha_\sigma}{\gamma_\sigma} - \frac{\beta_\sigma}{\delta_\sigma} + o(1) \text{ , where } \sigma = \begin{pmatrix} \alpha_\sigma & \beta_\sigma \\ \gamma_\sigma & \delta_\sigma \end{pmatrix} \\ &= z + \sum_{\sigma\in G,\sigma\neq id} \frac{1}{\gamma_\sigma\delta_\sigma} + o(1) \text{ , } z\to\infty,\end{aligned}$$

because $\alpha_\sigma\delta_\sigma - \beta_\sigma\gamma_\sigma = 1$. The Schottky group G admits the holomorphic involution, thus we have with $\sigma \in G$ also $z \mapsto -\sigma(-z) \in G$, i.e.

$$\begin{pmatrix} -\alpha & \beta \\ \gamma & -\delta \end{pmatrix} \in G \iff \begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix} \in G\,.$$

Then $\sum_{\sigma\in G,\sigma\neq id} \frac{1}{\gamma_\sigma\delta_\sigma} = 0$ and the integral has the desired asymptotics.

The evaluation methods for the integral of first kind presented in Chapter 2 can be adapted to work for this integral of second kind. Now that we have the last ingredient we can construct the spinor, but we have to be aware of some pitfalls.

To circumvent the burden of choosing the right branch of the roots involved in the holomorphic section

$$h_\delta = \sqrt{D\theta\left[\delta\right]_{|0}\cdot v}$$

and normalization constant

$$c_\delta = \sqrt{D\theta\left[\delta\right]_{|0}\cdot V}$$

we actually compute

$$\begin{aligned} a^2(P) &= C(P)\,\theta\left[\varepsilon\right]^2\left(\int_{P_o}^P v - \frac{1}{2}V\right) \\ b^2(P) &= C(P)\,\theta\left[\varepsilon\right]^2\left(\int_{P_o}^P v + \frac{1}{2}V\right) \qquad (4.3.1) \\ ab(P) &= (-1)^{<\varepsilon>}\,C(P)\,\theta\left[\varepsilon\right]\left(\int_{P_o}^P v - \frac{1}{2}V\right)\theta\left[\varepsilon\right]\left(\int_{P_0}^P v + \frac{1}{2}V\right)\end{aligned}$$

with the common factor

$$C(P) = -i\frac{D\theta\left[\delta\right]_{|0}\cdot V}{\theta\left[\varepsilon\right]^2\left(\frac{1}{2}V\right)}\frac{D\theta\left[\delta\right]_{|0}\cdot v}{\theta\left[\delta\right]^2\left(\int_{P_0}^P v\right)}\exp\left(\int_{P_0}^P \Omega\right)\,.$$

Here we used $\theta[\varepsilon](-V/2) = (-1)^{<\varepsilon>}\theta[\varepsilon](V/2)$, where $<\alpha,\beta>=\alpha\cdot\beta$ is called the *parity* of the spin structure.

The normalized Abelian differentials v_i are produced by Poincare theta series. The derivative V of the Abel map at $P_0 = \infty$ can be computed using the Riemann bilinear relation for normalized Abelian differentials of the first and the second kind:

$$V_n = -\int_{b_n} \Omega \,.$$

Let z_n be a point on the circle C_n. Then b_n joins z_n with $\sigma_n(z_n)$, and we have:

$$\begin{aligned} V_n &= -\int_{z_n}^{\sigma_n(z_n)} \Omega \\ &= \sum_{\sigma\in G} \sigma(z_n) - \sigma(\sigma_n(z_n)) \qquad (4.3.2) \\ &= \sum_{\sigma\in G/G_n} \sum_{k=-\infty}^{\infty} \sigma\circ\sigma_n^k(z_n) - \sigma\circ\sigma_n^{k+1}(z_n) \\ &= \sum_{\sigma\in G/G_n} \sigma(A_n) - \sigma(B_n) \,. \qquad (4.3.3) \end{aligned}$$

Formula (4.3.3) is much more efficient than (4.3.2) and needs only a slightly different algorithmic treatment[3]. The fact that the b-cycles of the homology basis associated to a Schottky group are only defined up to a-cycles causes no problems here because Ω is normalized to integrate to zero around a-cycles.

The situation is different for the period matrix. The Schottky series 5.6.3 only determines the period matrix, as well as the Abel map, up to periods of the logarithm function. This ambiguity can be encoded in a class of modular transformations for the Schottky basis:

$$\begin{aligned} a_i' &= D_i^j a_j + C_i^j b_j \,, \\ b_i' &= A_i^j b_j + B_i^j a_j \end{aligned}$$

with $D_i^j = A_i^j = \delta_i^j$, $C_i^j = 0$. This would result in a change of the characteristics

$$\begin{aligned} \alpha_i' &= D_j^i \alpha_j \,, \\ \beta_i' &= \beta_j - B_j^i \alpha_j + B_i^i \end{aligned}$$

which would in general change the values of the spinors. Thus we have to hold the b-cycles fixed, and determine the correct branch of the logarithm. We do this by integrating the holomorphic differentials along the b-cycles. The integration does not need to be accurate, because we only use it to determine the right period. Because of the quasi-periodicity of the θ-function the ambiguity of the Abel map has no impact on the spinors.

[3]Skipping this optimization is excusable because V only needs to be calculated once per surface, so that it only a minor impact on the total computational cost, anyhow we are using it.

A last issue is related to the roots of $\theta[\delta]\left(\int_{P_0}^{P} v\right)$ that cancel with the roots of the holomorphic section h_δ. From the point of view of numerics this is rather delicate. One might hope that the Jacobian variety is large enough and that we practically never hit one of the $g-1$ roots of $\theta[\delta]$, but even getting close would spoil the accuracy. As a matter of fact, in the hyperelliptic case the $g-1$ roots are a subset of the $2N+2$ fixed points of the involution π. These fixed points will play an important role when we generate cycles taking advantage of the full symmetry of the surface. But fortunately there is an easy solution: the spinors are independent of the choice of the odd characteristics δ. Whenever the absolute value of the θ-function drops below a certain threshold, which means we "may approach" a root, we only need to go through the $2^{g-1}(2^g-1)$ odd characteristics until we find one that generates no root.

At this point we are able to evaluate the spinors in a stable manner. In Section 5.6.3.1 we present an unoptimized implementation of the above using the *jtem* project *riemann*.

4.3.2. Choosing a concrete homology basis. The evaluation of Weierstrass forms is rather expensive, so that it pays to take advantage of the surface symmetries. The estimates in Section 2.4 and Figures 2.4.5, 2.4.6, and 2.4.7 show that the evaluation of the Poincare theta series gets more expensive as we get close to the isometric circles. We will try to choose all cycles such that they stay as far away from the boundary of the fundamental domain F as possible. However, b-cycle must cross the associated isometric circles, so the best that we can do here is to make it cross orthogonally.

The following strategy works. For each isometric circle C define a concentric circle C^* with radius equal to half the minimum distance of C to all other isometric circles. Discretize[4] these circles and use half of them as a-cycles. Because of the holomorphic involution none of these circles has the origin as an inner point. To construct the b-cycles proceed as follows. For each generator σ_i choose a point z_i on the associated isometric circle C around the repelling fixed point. If C contains fixed points, choose z_i to be one of these. Connect z_i to the origin with a path γ_i that proceeds first radially until it reaches C_i^* and then proceeds to the origin, bypassing the interior of all circles C^* or at least their discretization. Proceed similarly to get a path γ_i^* connecting $\sigma_i(z_i)$ and the origin. If z_i is a fixed point of the holomorphic involution choose $\gamma_i^* = -\pi^*\gamma$. Finally we let $b_i = \gamma_i - \gamma_i^*$. Such a homology basis satisfies the symmetry requirements in Equation 4.1.5 for Theorem 4.1.5, which now yields a set of independent period conditions. In the case of symmetric admissible Schottky data, choose $a_j = \tau a_i = \pm\overline{a_j}$ and $\gamma_j = \tau\gamma_i = \overline{\gamma_i}$ for those generators with $\sigma_j(z) = \overline{\sigma_j^{-1}(\overline{z})}$ (resp. $\sigma_j(z) = \overline{\sigma_j(\overline{z})}$) and find a Schottky basis stisfying the hypotheses of Lemma 4.2.1 (Figure 4.3.1).

[4]We are using squares having one diagonal alined with the origin.

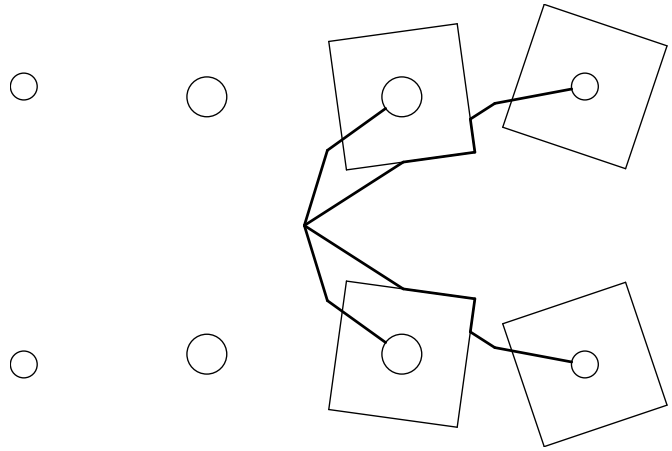


FIGURE 4.3.1. a-cycles and paths γ_i for a symmetric hyperelliptic example of genus 4

4.3.3. Integrating Weierstrass forms. Compared to the cost of evaluating spinors the cost of the actual integration method is negligible. Evaluating one coordinate of the Weierstrass representation is as expensive as evaluating them all, thus the integration should process all three coordinates simultaneously. Furthermore the integrator must be adaptive because the conformal factor can vary by several orders of magnitude and takes its highest values close to the boundary of the fundamental domain F, which cannot be avoided in the integration process.

Integrators based on adaptive ODE-solvers using an extrapolating scheme worked best. These methods allow a dynamic change of the integration order, which is especially suitable for analytic functions. In theory, this makes them superior to single-step methods like Runge-Kutta-Fehlberg (RKF), which surprisingly performed better in our case than the classical extrapolating method of Burlisch-Stoer (BS) [**PTVF92, SB90**]. The single-step method of RKF sometimes had problems reaching the prescribed precision. Both RKF and BS have the problem that the number of evaluations they need to reach a precision grows almost quadratically with it. The "Extrap" [**HNW93**] ODE-solver by E. Hairer and G. Wanner was the most successful: the number of evaluations grows just linearly with the precision and it always reached the prescribed error bound (Table 1,2).

We can read off the accuracy of integration methods from the first coordinate of $\hat{a}$-cycle periods, which have to vanish according to Theorem 4.1.5 (Table 2).

We discuss an implementation in Section 5.3.2.6.

4.3.4. Solving the period condition. Theorem 4.1.5 and Lemma 4.2.1 guarantees that we have exactly as many free parameters as we have independent period conditions. Thus we have a classical root finding problem. Even though

precision	Extrap	BS	RKF
10^{-1}	148	247	256
10^{-2}	169	582	409
10^{-3}	231	2660	1441
10^{-4}	364	38456	20346
10^{-5}	448	not reached	81712
10^{-6}	542		not reached
10^{-7}	698		
10^{-8}	858		
10^{-9}	980		

TABLE 1. Number of spinor evaluations for the helicoid $\mathcal{H}e_3$ (Section 4.4.3) using the adaptive ODE-solver Extrap, Bulirsch-Stoer (BS), and Runge-Kutta-Fehlberg(RKF).

this sounds like a standard task in numerics, there are just no good, general methods for solving systems of more then one nonlinear equations. Furthermore, we are not able to compute the derivative of our problem analytically.

For minimization problems there exist methods that do not need any derivatives. We translated the root finding problem into a minimization problem using the target energy

$$E_s\left(x_1,\ldots,x_n\right)=\frac{1}{2}\sum_{\gamma=a_1,b_1,\ldots,a_g,b_g}\left(\operatorname{Re}\int_\gamma\left(-a^2+b^2,i\left(a^2+b^2\right),2ab\right)\right)^2,$$

where the subscript s indicates the spin structure. Finding a root of the energy is equivalent to solving the period condition. The parameters $x_1,\ldots,x_n$ parameterize the Schottky data. For the minimization process it is important that the parameters be all of the same magnitude. This is given for the fixed points of involution, but not for the loxodromic factors μ_i, whose magnitudes ranges for example for the He_8 from 10^{-2} to 10^{-10} . It turned out to be best to represent the loxodromic factors in polar coordinates and scale the the parameter for the absolute value to be of order 1.

For the actual minimization process we mainly used variations of Powell's directions set method [**Bre02**] and the downhill simplex method of Nelder and Mead [**NM65**]. These methods require only function evaluations, not derivatives. In some versions Powell's method converges quadratically. Thus for well-conditioned problems Powells's method is almost surely faster than the downhill simplex method, which, on the other hand, is known to be extremely robust. Our experiments confirm this. For genus 1 and 2 Powell's methods was clearly faster, but already for genus 3 the simplex downhill method produced better results (Section 5.3.2.3).

precision	Extrap	BS	RKF
10^{-1}	$0.11{\cdot}10^{-1}$	$0.07{\cdot}10^{-1}$	$0.07{\cdot}10^{-1}$
10^{-2}	$0.24{\cdot}10^{-2}$	$0.05{\cdot}10^{-2}$	$0.18{\cdot}10^{-2}$
10^{-3}	$0.05{\cdot}10^{-3}$	$0.07{\cdot}10^{-3}$	$2.3{\cdot}10^{-3}$
10^{-4}	$0.20{\cdot}10^{-4}$	$0.55{\cdot}10^{-4}$	$2.0{\cdot}10^{-4}$
10^{-5}	$0.17{\cdot}10^{-5}$	not reached	$15{\cdot}10^{-5}$
10^{-6}	$0.26{\cdot}10^{-6}$		not reached
10^{-7}	$0.12{\cdot}10^{-7}$		
10^{-8}	$0.22{\cdot}10^{-8}$		
10^{-9}	$0.49{\cdot}10^{-9}$		

TABLE 2. Numererical error of numerical integration of the first coordinate of $\hat{a}$-cycle periods of the helicoid $\mathcal{H}e_3$(Section 4.4.3). The integration was performed using the adaptive ODE-solver Extrap, Bulirsch-Stoer (BS), and Runge-Kutta-Fehlberg(RKF).

Whether a minimum of the target function is a root or not cannot be decided by just looking at the target function. An error tolerance of 10^{-15} for the integrator is the best what we can expect from double precision[5]. Thus pushing E_s below

$$\frac{1}{2}\sum_{\gamma=a_1,b_1,\ldots,a_g,b_g}\left(\left(10^{-13},10^{-13},10^{-13}\right)\right)^2 = 3g10^{-30}$$

is the most what we can hope to achieve.

To distinguish a solution of the period problem from a local minimum of the target function, we go back to the original root finding problem. In order to check whether $x_r \in \mathbb{R}^n$ is a root of a map $F : \mathbb{R}^n \to \mathbb{R}^n$, we check the convergence of the quasi-Newton method. In the absence of a an analytic derivative, approximate the derivative numerically at x_r[6]. Perform some iterations of the quasi-Newton method:

$$\begin{aligned} x_0 &= x_r \\ x_{n+1} &= x_n - DF^{-1}_{|x_r} \cdot F\left(x_n\right) . \end{aligned}$$

If it does not diverge it will jump around in a neighborhood of the numerical solution x_r. Let us denote the center of mass of the set $\{x_n\}$ by c and

$$d\left(x_r\right) = \max_n \left|x_n^i - c^i\right| e_i$$

[5]Double precision representation allows 16 decimals, so that in this representation is $1 + 10^{-16} = 1$.

[6]We are using an algorithm given by Ridders [**Rid, PTVF92**] based on the general idea of "Richardson's deferred approach to the limit". Ridders' method also provides an estimate of the error in the derivative (Section 5.3.2.2).

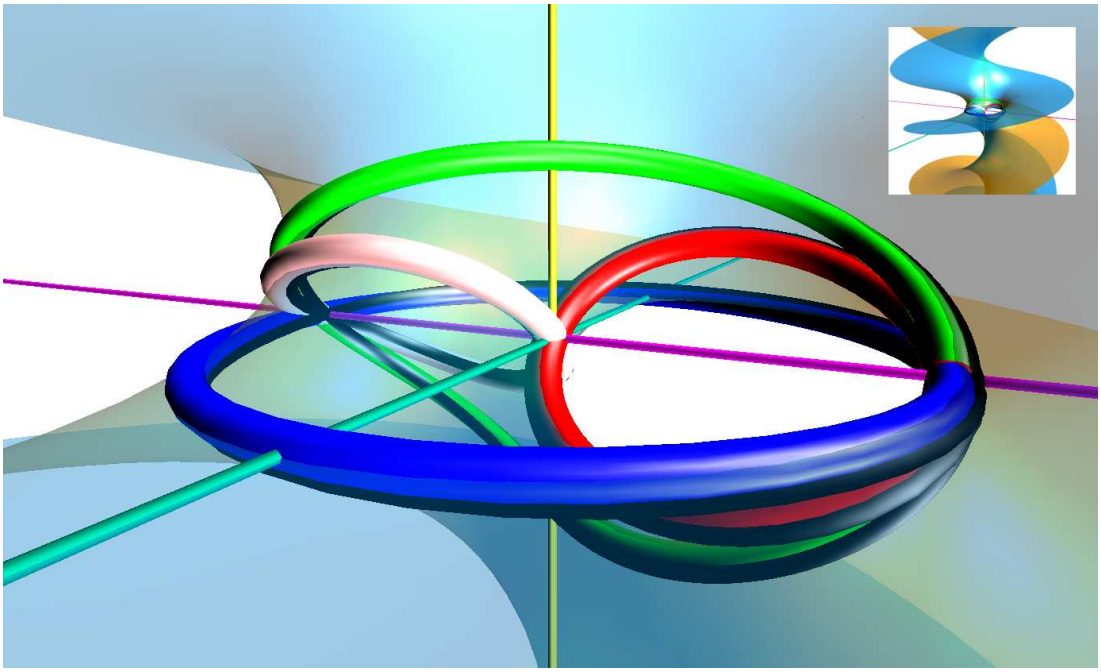


FIGURE 4.4.1. Cycles a_+(green), a_-(blue), b_+(pink), and b_-(red) of the Schottky basis $\mathcal{B}_+ = \{a_+, b_+\}$, $\mathcal{B}_- = \{a_-, b_-\}$, and $\mathcal{B}_0 = \{b_+, b_-\}$ associated to the Schottky data S_+, S_-, and S_0 respectively for the $\mathcal{H}e_1$.

the quasi-Newton deviation vector of the root x_r, which has proven to be a good measure for the precision of x_r (Section 4.4.1).

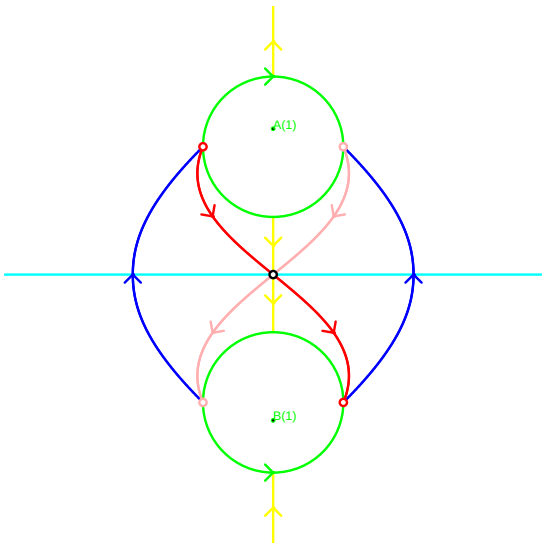
4.4. Numerical Analysis of the Examples

4.4.1. The Symmetric Genus One Case. We start our analysis with the symmetric genus one case for which we know the existence of at least one example: the famous helicoid of Karcher-Hoffmann-Wei, which we will refer to as $\mathcal{H}e_1$.

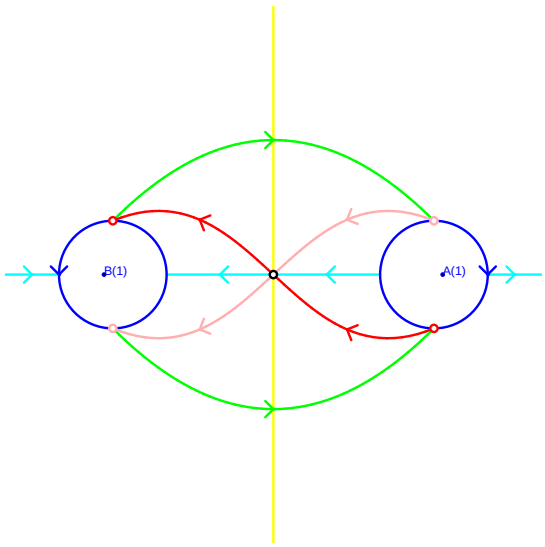
According to Section 4.2, we have two principle possibilities to generate symmetric admissible Schottky data:

$$S_+ = \{ix, -ix, y\} \quad \text{or} \quad S_- = \{x, -x, y\}$$

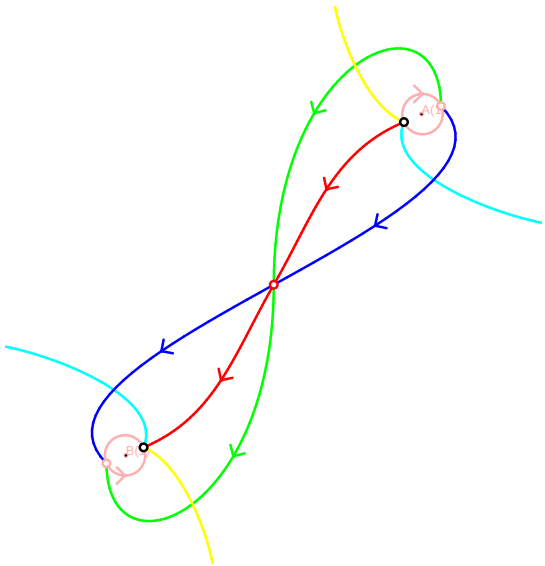
with $x, y \in \mathbb{R}$ (Figure 4.4.2). Figure 4.4.1 shows a close-up of these cycles on the $\mathcal{H}e_1$ and the three symmetry axes. The x-axis (purple) intersects the surface orthogonally in the three finite fixed points (black, red, and pink) of the holomorphic involution π. Any 180° rotation about one of the three coordinate axes will map the a-cycles a_+ and a_- onto themselves. The b-cycles are only invariant under the 180° rotation about the x-axis. A 180° rotation about the other coordinate axis (yellow, cyan) will map b_+ onto b_- and vice-versa.



(a) Cycles and symmetries for S_+



(b) Cycles and symmetries for S_-



(c) Cycles and symmetries for S_0

FIGURE 4.4.2. Cycles a_+(green), a_-(blue), b_+(pink), and b_-(red) of the Schottky basis $\mathcal{B}_+ = \{a_+, b_+\}$, $\mathcal{B}_- = \{a_-, b_-\}$, and $\mathcal{B}_0 = \{b_+, b_-\}$ associated to the Schottky data S_+, S_-, and S_0 respectively for the $\mathcal{H}e_1$. The vertical symmetry axis is colored in yellow, the horizontal one in cyan. The three fixed points of the holomorphic involution π are marked by small circles (black, red, and pink).

In the positive symmetric case with Schottky data S_+ the vertical symmetry line is broken by the isometric circles. The same holds for S_- and the horizontal symmetry line. A short computation shows that in the case of negative μ the intersections of the axis with circles are identified in such a way that the associated anti-holomorphic involutions comprise exactly one real oval as desired in Definition 4.1.6 for symmetric data[7]. A computation in the symmetric case shows that the isometric circles intersect for $\mu < \sqrt{8} - 3 \approx -0.172$, so that $\mu \in \left]\sqrt{8} - 3, 0\right[$.

Tracking the cycles in Figure 4.4.1, we can see that the a-cycles a_+ and a_- are twisted because the spin structure respects the anti-holomorphic involution τ, i.e. $\tau^* s = s$. For both symmetric bases we have

$$\tau b_\pm = \overline{b_\pm} = -b_\pm - a_\pm \,.$$

This yields a condition on the characteristics $\epsilon = (\alpha, \beta)$. We have

$$\beta = s\left(b_\pm\right) = \tau^* s\left(b_\pm\right) = s\left(-b_\pm - a_\pm\right) = s\left(b_\pm\right) + s\left(a_\pm\right) + a_\pm \circ b_\pm = \beta + \alpha + 1\,,$$

so that $\alpha = 1$ and ϵ is either $(1,0)$ or $(1,1)$. This means that for symmetric Schottky data the a-cycle is twisted and that one has a choice of an untwisted or twisted $b-$cycle. For $\mathcal{H}e_1$, b_+ and $b_-$are untwisted, so that $\epsilon = (1,0)$.

We could also have chosen b_+ and b_- as a homology basis, which is not symmetric but still admissible. Then the Schottky data is given by $S_0 = \{A, -A, \mu\}$ with $A, \mu \in \mathbb{C}$ (Equation 4.2.4). The resulting uniformization picture is also shown in Figure 4.4.2. For this asymmetric uniformization the characteristics must equal $(0,0)$ to generate the properly embedded $\mathcal{H}e_1$. As a matter of fact, when describing embeddings one can always choose a basis such that $\epsilon = (0,0)$.

For the genus one case it is possible to compute how admissible Schottky data transforms under a transformation of the homology basis. Let $S' = \{A', -A', \mu'\}$ be generic admissible Schottky data and $\{a', b'\}$ the associated basis. Let

$$\begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix} \in SP\left(2, \mathbb{Z}\right)$$

be a modular or Siegel transformation. This yields a new homology basis $\{a'', b''\}$ with

$$\begin{aligned} a'' &= \delta a' + \gamma b'\,, \\ b'' &= \alpha b' + \beta a'. \end{aligned}$$

According to a classic result [**BBE+94, Mum83**] the period matrix $B' = \log \mu'$ transforms to

$$B'' = 2\pi i \frac{\alpha B' + 2\pi i \beta}{\gamma B' + 2\pi i \delta}\,,$$

which results in:

$$\mu'' = \exp\left(2\pi i \frac{\alpha \log \mu' + 2\pi i \beta}{\gamma \log \mu' + 2\pi i \delta}\right)\,.$$

[7]A positive μ leads to two real ovals.

	to $\mathcal{B}_+$	to $\mathcal{B}_-$	to $\mathcal{B}_0$
from $\mathcal{B}_+$	$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$	$\begin{pmatrix} -1 & 1 \\ -2 & 1 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 \\ -1 & 1 \end{pmatrix}$
from $\mathcal{B}_-$	$\begin{pmatrix} 1 & -1 \\ 2 & -1 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$	$\begin{pmatrix} -1 & 1 \\ -1 & 0 \end{pmatrix}$
from $\mathcal{B}_0$	$\begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}$	$\begin{pmatrix} 0 & -1 \\ 1 & -1 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$

TABLE 4. Modular transformations that transforms the Schottky basis $\mathcal{B}_+ = \{a_+, b_+\}$, $\mathcal{B}_- = \{a_-, b_-\}$, and $\mathcal{B}_0 = \{b_+, b_-\}$ associated to the Schottky data S_+, S_-, and S_0 into each other.

To compute A'' we use V, the derivative of the Abel map at ∞, and Equation 4.3.3. We obtain

$$V = A - B = 2A\,.$$

Let us denote by ω' and ω'' the normalized holomorphic differentials integrating to $2\pi i$ on $a'$and a''respectively. Then

$$\omega'' = \frac{2\pi i}{\int_{a''}\omega'}\omega' = \frac{2\pi i}{\int_{\delta a'+\gamma b'}\omega'}\omega' = \frac{2\pi i}{\delta 2\pi i + \gamma \log \mu'}\omega'$$

and finally we get:

$$A'' = \frac{2\pi i}{\delta 2\pi i + \gamma \log \mu'}A'\,.$$

For the symmetric Schottky data S_+ and S_- and negative μ the period matrix is of the form $B_\pm = x + \pi i$ with $x \in \mathbb{R}$. Transforming the associated basis $\mathcal{B}_+$ and $\mathcal{B}_-$ to $\mathcal{B}_0$ using the modular transformations listed in Table 4, we obtain

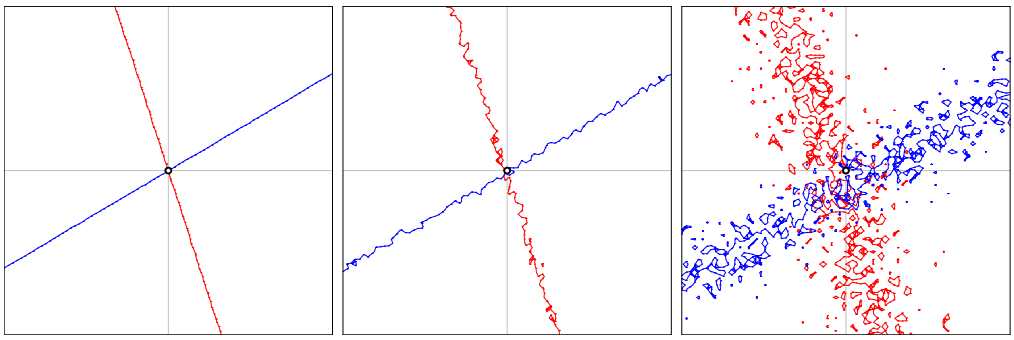
$$B_0 = \pm 2\pi i \frac{x \mp \pi i}{x \pm \pi i}.$$

Then $|B_0| = 2\pi$ so that the underlying torus is rhombic, which had to be the case by construction.

According to Lemma 4.2.1 the independent period conditions for the positive symmetric admissible Schottky data S_+ are

$$f\,(x,y) \;=\; \mathrm{Re}\int_a i\big(a^2+b^2\big) \tag{4.4.1}$$

$$\text{and } g\,(x,y) \;=\; \mathrm{Re}\int_b 2ab\,. \tag{4.4.2}$$



(a) $\delta = 10^{-11}$ (b) $\delta = 10^{-12}$ (c) $\delta = 10^{-13}$

FIGURE 4.4.3. Series of magnifications of the vicinity of the root (x_r, y_r) associated to the Schottky data for the $\mathcal{H}e_1$. The plots show f and g in the domains $D_\delta(x_r, y_r)$ with $\delta = 10^{-11}, 10^{-12}$, and 10^{-13}.

Integrating the periods takes less then a millisecond on a modern PC[8]. This enables us to scan the domain in a wide range with a high resolution. For Figures 4.4.4 and 4.4.5, we integrated the periods of 2 million surfaces[9] . The figures suggest that there are infinite many untwisted and twisted closed symmetric helicoids of genus one. The only embedded example seems to be $\mathcal{H}e_1$, which is marked in Figure 4.4.4 by a circle and shown in Figure 4.4.6 . Figure 4.4.7 shows the most simple twisted example which is also marked by circle in Figure 4.4.5.

The Schottky data for the $\mathcal{H}e_1$ is $S_+ = \{ix_r, -ix_r, y_r\}$with

$$x_r = 1.276472774241 \quad \text{and} \quad y_r = 6.578689918442 \cdot 10^{-2}\,.$$

The periods $f(x_r, y_r)$ and $g(x_r, y_r)$ are about 10^{-13} and realize the chosen accuracy for the integrator[10]. In this case the target energy $E_s(x_r, y_r) = 2 \cdot 10^{-26}$ reaches the predicted optimal value. However, noe of this tells us any thing about the precision of the Schottky data.

The two-dimensional root problem admits a nice graphical analysis of the precision of root (x_r, y_r). We magnify the vicinity of the root until we approach the ultimate resolution. The failure of the numerics will blur the picture. Figure

[8]Intel Pentium 5, 2.5GHz

[9]The plots have been generated using a standard "marching squares" algorithm (*numericalMethods.geometry.hyperSurfaces.MarchingSqaures)*.

We chopped off the lower right part of the plots, which is heavily effected by numerical noise. The noise is due to the fact that $2a(z)b(z)$ and $i\exp(z)$ have the same asymptotics and thus g grows exponentially in x (with constant y).

[10]We choose the accuracy for the θ-function and Schottky series always two digits more accurate than the of the integration.

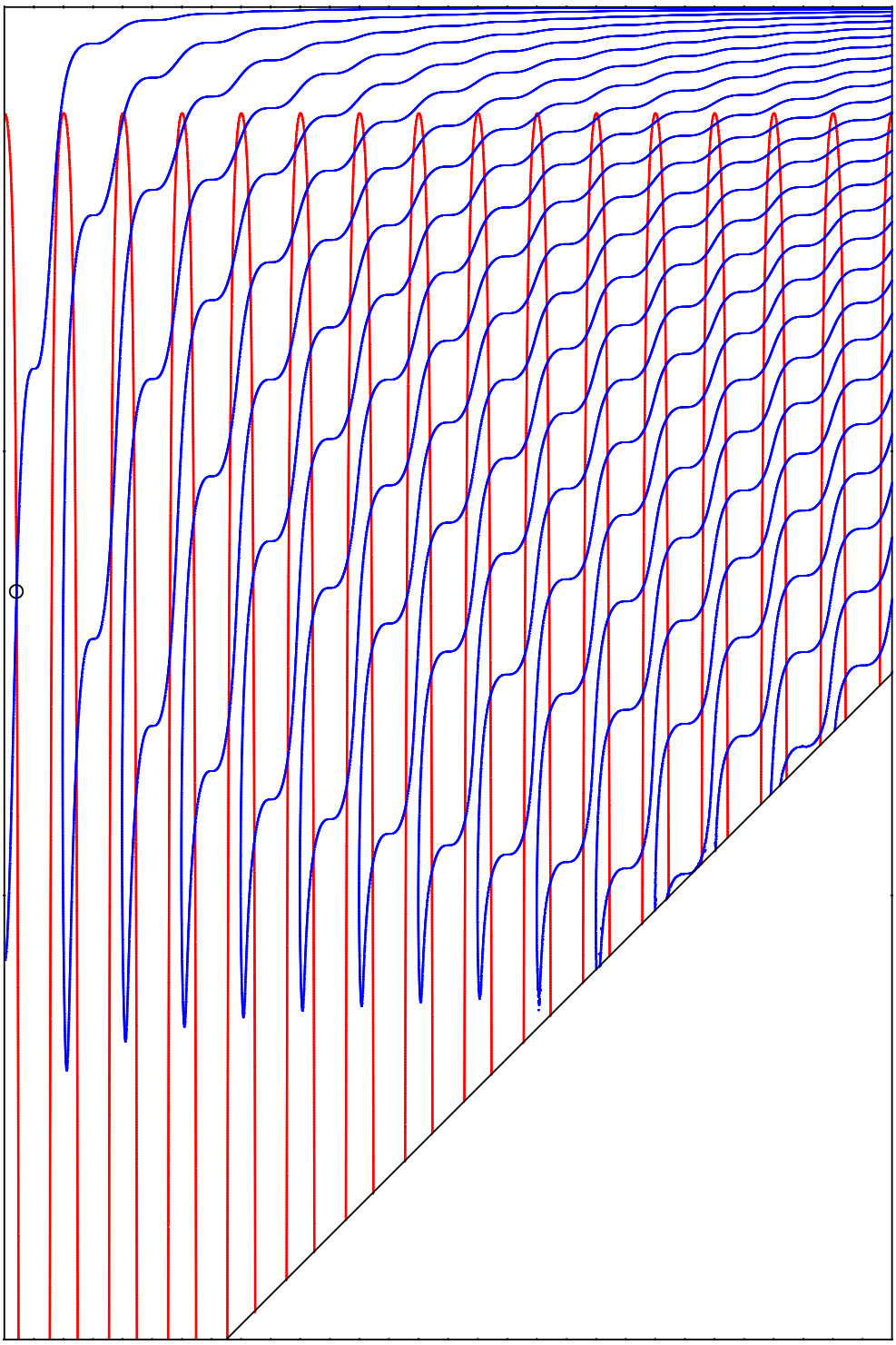


FIGURE 4.4.4. Zero level lines of the two independent period conditions $f(x,y) = \mathrm{Re}\int_a i(a^2+b^2)$ (red) and $g(x,y) = \mathrm{Re}\int_b 2ab$ (blue) in the domain $]0, 30\pi[\times]-0.15, 0[$ for symmetric genus one case with characteristics $\epsilon = (1,0)$. The example of Karcher-Hoffmann-Wei is marked by a circle (Figure 4.4.6).

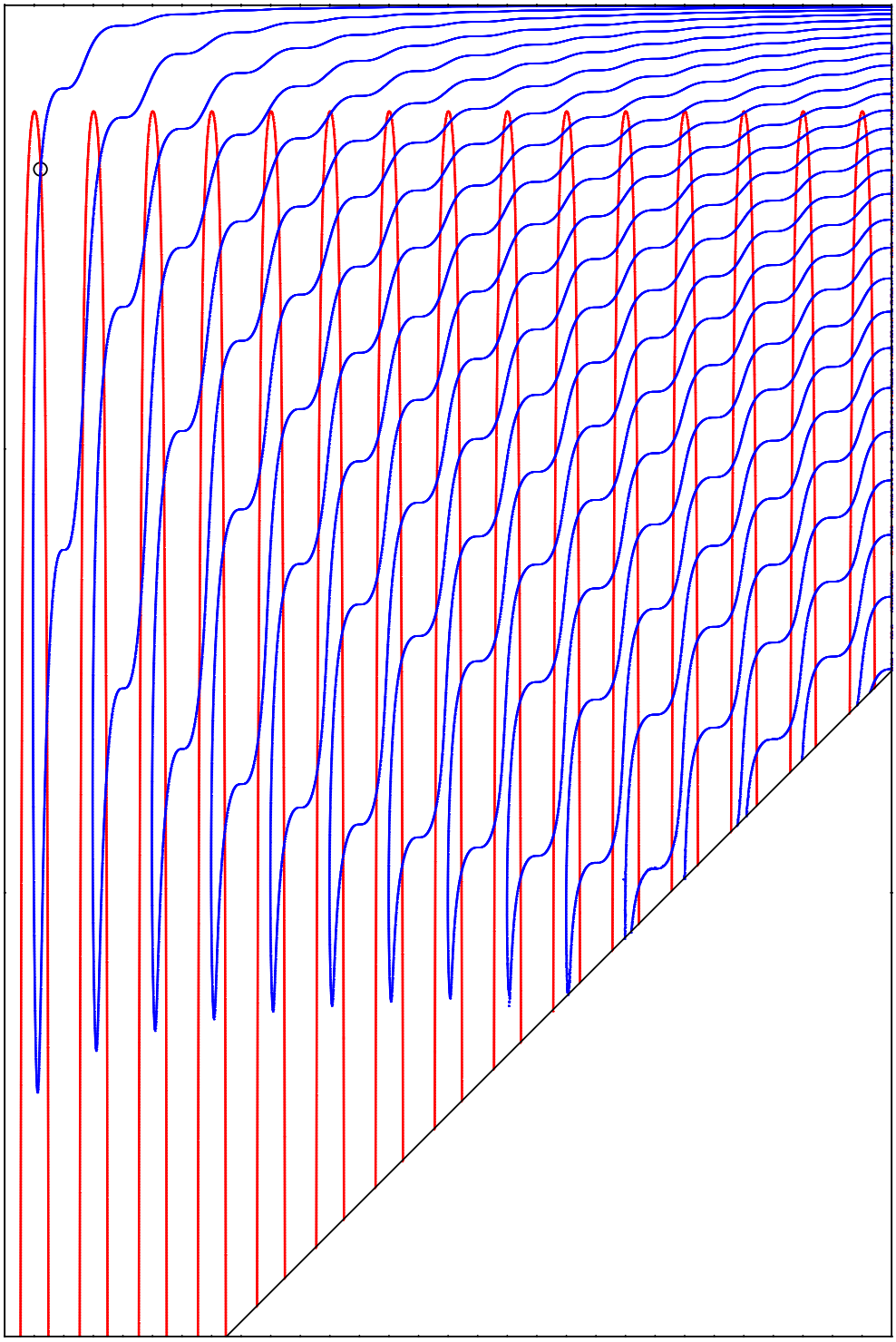


FIGURE 4.4.5. Zero level lines of the two independent period conditions $f(x,y) = \mathrm{Re}\int_a i(a^2+b^2)$ (red) and $g(x,y) = \mathrm{Re}\int_b 2ab$ (blue) in the domain $]0, 30\pi[\times]-0.15, 0[$ for symmetric genus one case with characteristics $\epsilon = (1,1)$. The most simple example of a twisted genus-one helicoid is marked by a circle (Figure 4.4.7).

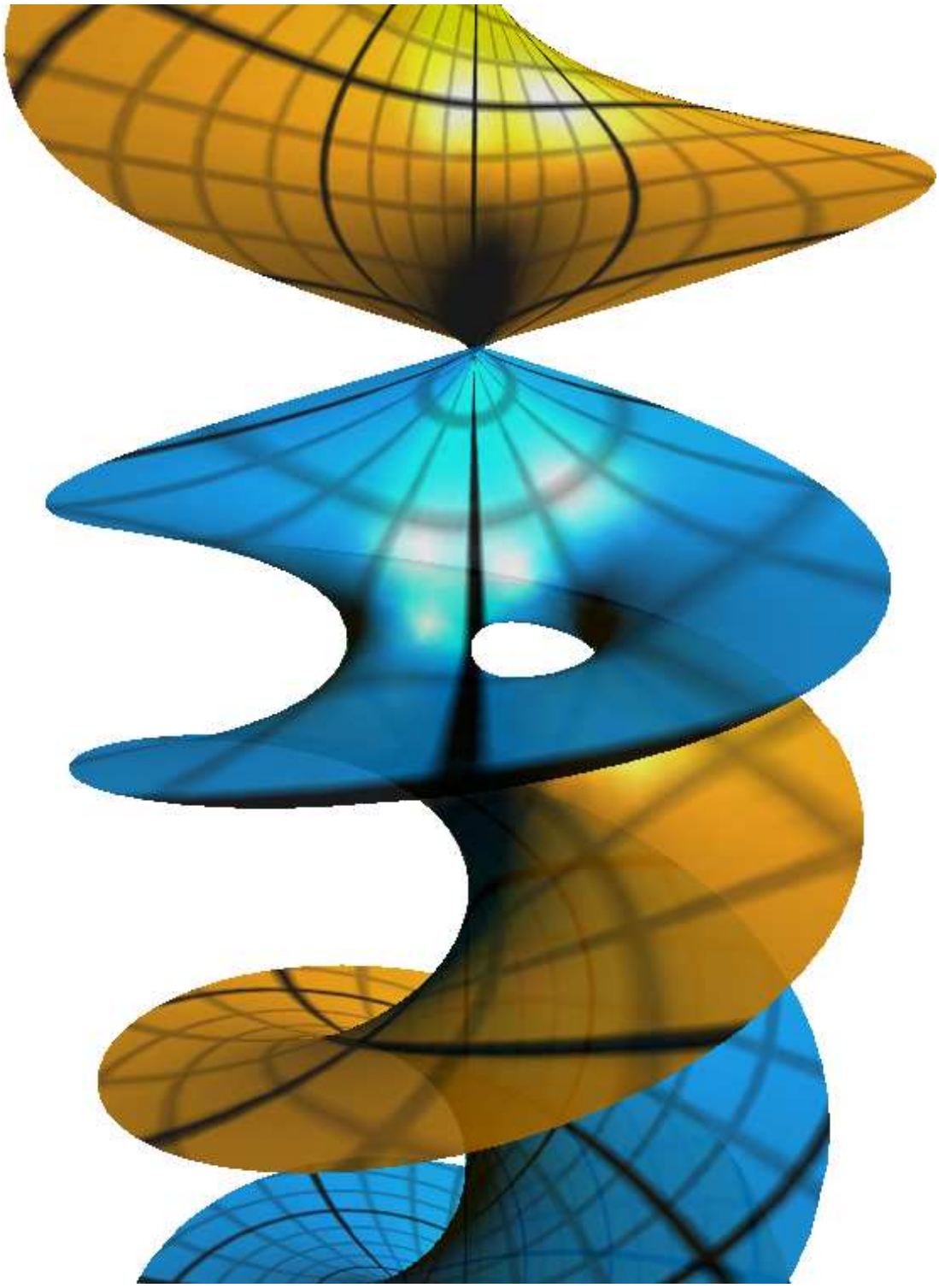


FIGURE 4.4.6. $\mathcal{H}e_1$, the example of Karcher-Hoffmann-Wei.

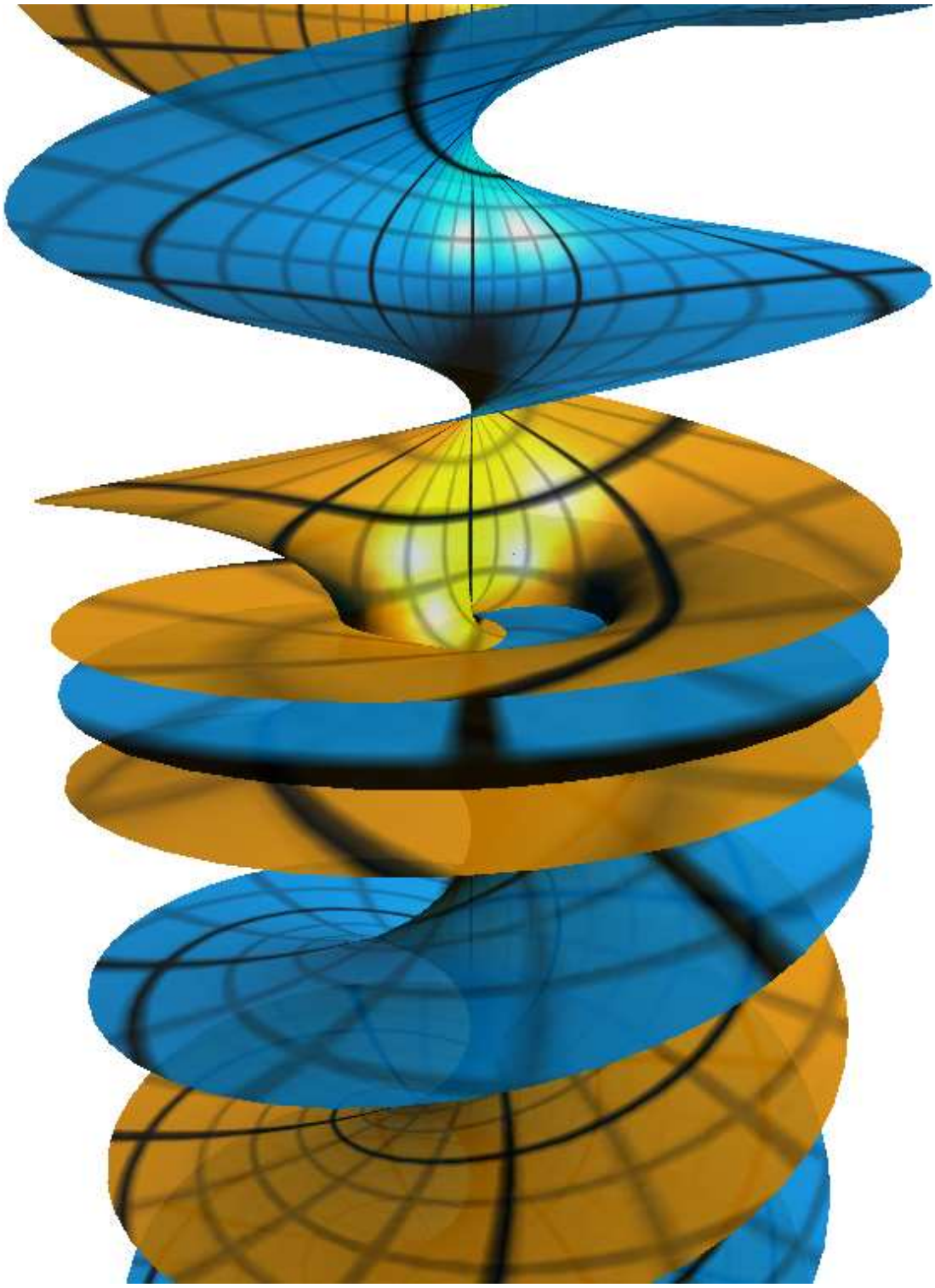
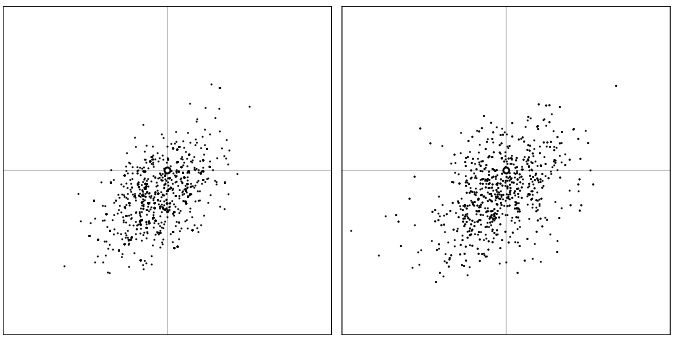


FIGURE 4.4.7. The most simplest twisted symmetric genus-one helicoid, which was found in [**Bob98**].



(a) $\delta = 10^{-13}$ (b) $\delta = 10^{-13}$

FIGURE 4.4.8. Quasi-Newton iteration for he root (x_r, y_r) associated to the Schottky data for the $\mathcal{H}e_1$, using the highly accurate numerical Derivative (4.4.4) on the left and the coarse Approximation (4.4.5) on the right.

4.4.3 shows a series of zooms: the plots show f and g in the domains

$$D_\delta(x_r, y_r) = [(1-\delta)\,x_r, (1+\delta)\,x_r] \times [(1-\delta)\,y_r, (1+\delta)\,y_r] \tag{4.4.3}$$

with $\delta = 10^{-11}, 10^{-12},$ and 10^{-13}. The series shows that we can take 12 digits for granted

We now compare the above with the quasi-Newton deviation method we described at the end of Section 4.3.4. This method has, compared to any graphical method, the advantage that it works for any number of parameters. The drawback of this method is that it requires the derivative, which we have to approximate numerically. Fortunately the method is very robust concerning the accuracy of the derivative. Ridders' method predicts an accuracy of 10^{-10} for the derivative

(4.4.4)

$$DF_{|(x_r,y_r)} = \begin{pmatrix} \partial_x f & \partial_y f \\ \partial_x g & \partial_y g \end{pmatrix}_{|(x_r,y_r)} \approx \begin{pmatrix} 2.5723986145 & -65.957791646 \\ -5.8519250933 & -28.707915932 \end{pmatrix}.$$

Figure 4.4.8 shows on the left hand side the point cloud generated by the quasi-Newton method using the approximation above. The maximal deviation is less then $0.5 \cdot 10^{-14}$. On the right hand side Figure 4.4.8 we show the result of the quasi-Newton iteration using the matrix

$$\begin{pmatrix} 2 & -60 \\ -5 & -20 \end{pmatrix} \tag{4.4.5}$$

as approximation of the derivative. Although this matrix is, with a relative error of more then 20%, just a coarse approximation of the derivative, the methods still predicts with $0.9 \cdot 10^{-14}$ a suitable precision.

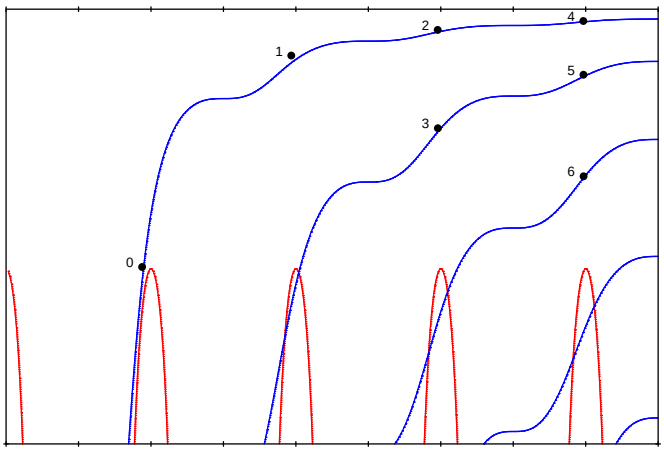


FIGURE 4.4.9. Magnification of upper left part of Figure 4.4.4 showing the zero levels of f and g in the domain $]0, 9\pi[\times]-0.02, 0[$. The indexed black points mark the projections of the Schottky data S_0 (Table 5) of closed asymmetric helicoids to the symmetric data S_+.

4.4.2. The Asymmetric Genus One Case. In this case the Riemann surface has no anti-holomorphic involution. In the previous section we saw that a rhombic torus can be uniformized by the unsymmetric Schottky data S_0. To deterimine whether a torus is rhombic we have to transform the modulus $B = \log \mu$ using the right modular transformation into the fundamental domain of the modular group. This task is easy for a torus ([**Hei95**, p. 25-26]) and handled by Siegel's reduction algorithm [**Sie89, Hei95, DHB$^+$04**], which also does the job also for higher genus (Section 3.4). Once we have the reduced modulus, we only have to check whether its absolute value equals 2π.

#	A	μ	$\frac{1}{2\pi}\lvert B_r\rvert$
0	0.34087472626 + 5.89807485189 i	-0.0118611871544 - 0.0033906555950 i	1.0616787876
1	1.255994424 + 12.366740055 i	-0.00213350902 - 0.0010667089 i	1.05131502
2	1.498577 + 18.714425 i	-0.000953910 - 0.00038601 i	1.179504
3	0.872954 + 18.723598 i	-0.00547658 - 0.0011283 i	1.035305
4	1.6347 + 25.03044 i	-0.0005440 - 0.000183 i	1.269
5	1.14437 + 25.0353 i	-0.003015 - 0.000623 i	1.032
6	0.6621 + 25.040 i	-0.007688 - 0.00088 i	1.021

TABLE 5. Schottky data and the absolute value of their reduced modulus normalized by 2π of asymmetric helicoids with characteristics $\epsilon = (1, 0)$.

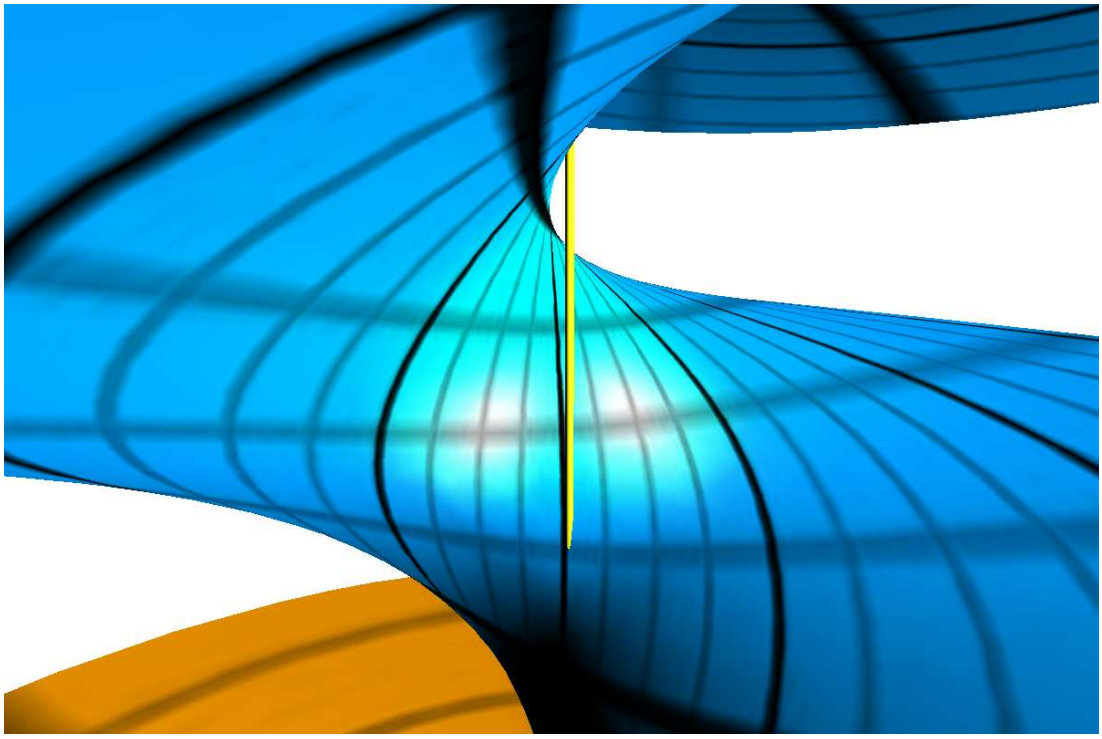


FIGURE 4.4.10. Magnification of a traversal intersection of the vertical axis (yellow) with the asymmetric Example 2 (Table 5).

The Schottky data for the asymmetric case is given by $S_0 = \{A, -A, \mu\}$ with no additional constraints. In principle we have the choice of all four different characteristics, but we stick to those two from the symmetric case. Then the symmetric cases with the Schottky data S_+ and S_- becomes a subset of S_0.

The plots shown in Figure 4.4.4 and 4.4.5 were a crucial step toward finding asymmetric examples. I noticed that the zero level lines of f (Equation 4.4.1) and g (Equation 4.4.2) in some regions almost touch: there I started to search again and succeeded. Figure 4.4.9 is a magnification of upper left part of Figure 4.4.4 showing the zero levels in the domain $]0, 9\pi[\times]-0.02, 0[$. The black points mark the projections of the Schottky data S_0 of the closed asymmetric examples we have found to the symmetric data S_+. The points all lie close to the intersection of a "blue line" and a parameter line $x = 2n\pi$ [11]. Table 5 lists their Schottky data as well as the absolute value of their reduced modulus normalized by 2π. The displayed digits can be taken as precise according to the quasi-Newton deviation test. The figures for the fourth series with Examples 4, 5, and 6 laying close to the parameter line $x = 8\pi$ already have a pure precision, and we do not give more examples.

[11]For the twisted examples the situation is quite simular, just that the examples lie close to the parameter lines $x = (2n + 1)\,\pi$.

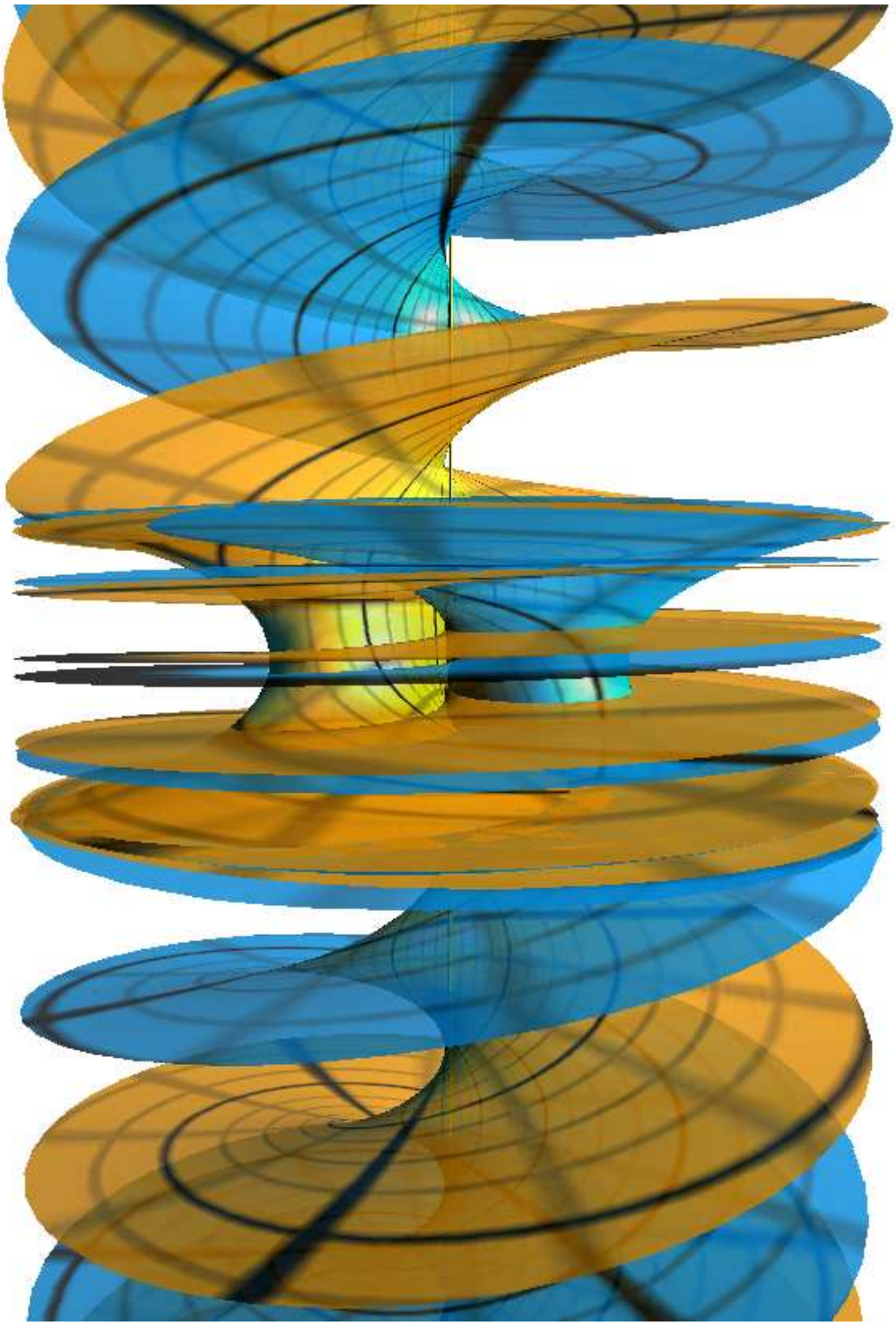


FIGURE 4.4.11. Asymmetric Example 2 (Table 5).

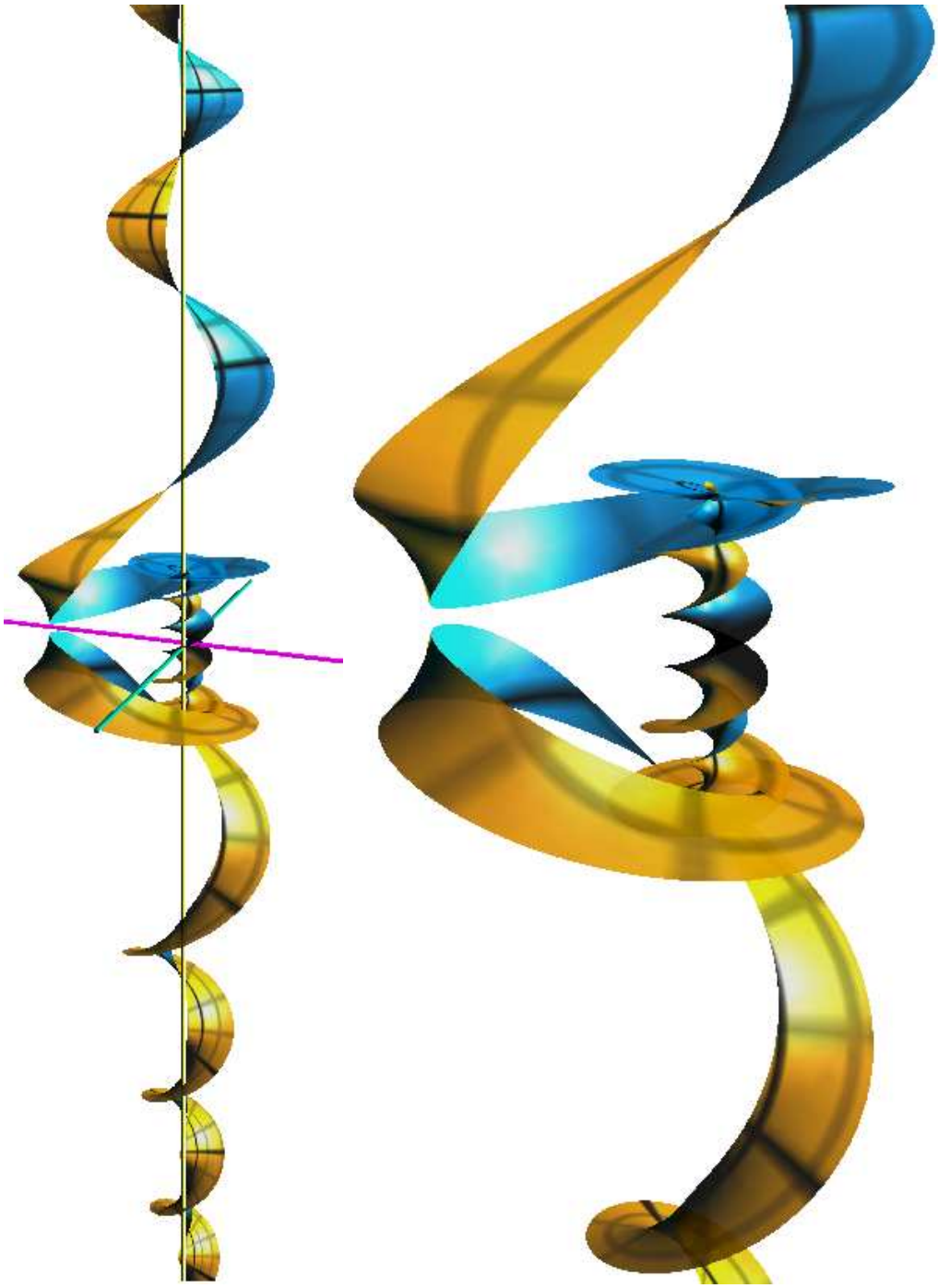


FIGURE 4.4.12. Strip of constant width whose center line is the image of the imaginary axis of the asymmetric Example 2 (Table 5).

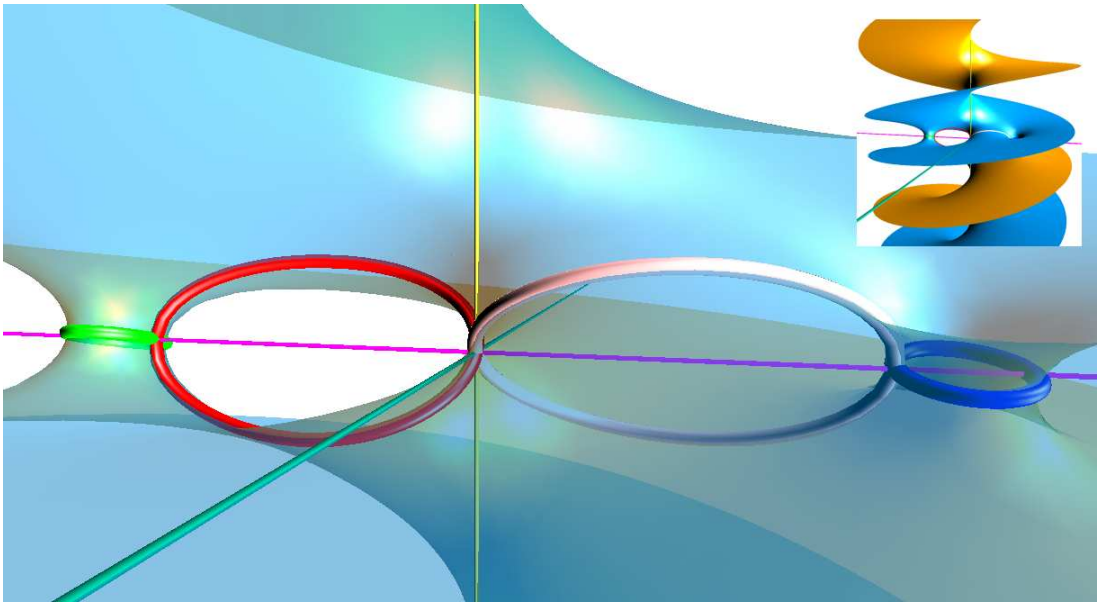


FIGURE 4.4.13. Loops on $\mathcal{H}e_2$

Unfortunately, none of these asymmetric examples are embedded and they intersect in a complicated manner (Figure 4.4.11) . In contrast to the symmetric case the vertical axis does not lie on asymmetric examples. Figure 4.4.10 is a magnification of a traversal intersection of the vertical axis with Example 2 (Table 5). To give a better visual idea of the surface we plotted a strip of constant width that converges asymptotically to the vertical axis (Figure 4.4.12).

4.4.3. Symmetric Hyperelliptic Higher Genus Examples. For the higher genus examples we restrict ourselves to the symmetric case. According to the definition of admissible data the Riemann surface $C \to C_0$ is, a ramified double covering of genus $g = 2g_0 + N$ of a compact Riemann surface C_0 of genus g_0 with $2N + 2$ branch points, the fixed points of the involution π. For $g_0 = 0$ the Riemann surface C is hyperelliptic. All examples we have found are of this type. The numerical analysis of some nonhyperelliptic examples in Section 4.4.4 strongly suggest that these surface do not exist.

From now on we always will use the negative symmetric Schottky uniformization data (4.2.6) because we experienced faster convergence of the Schottky series than for positive Data (4.2.7). For genus $g = 2$ we have:

$$S_{-,2} = \left\{A, -A, \mu, -\overline{A}, \overline{A}, \mu\right\}$$

with $A, \mu \in \mathbb{C}$. Figure 4.4.14 shows the associated fundamental domain and Schottky basis. The five fixed points of the hyperelliptic involution π (marked again as small circles colored black,red,pink,green, and blue) will be the intersection of the helicoid with the x-axis. Thus we can expect the cycles to be displaced

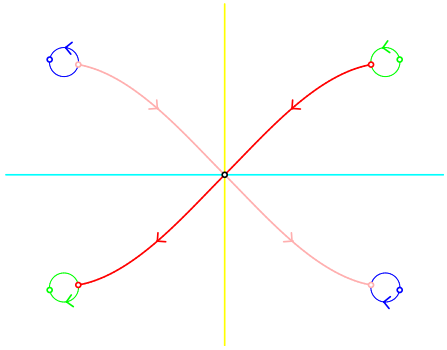


FIGURE 4.4.14. Schottky uniformization of $\mathcal{H}e_2$.

horizontally along the x-axis, the b-cycles in the center, and the a-cycles to the outside. Figure 4.4.13 shows a close-up of these cycles on the $\mathcal{H}e_2$, the embedded symmetric hyperelliptic of genus 2 (Figure 4.4.15) . A 180° rotation about the x-axis maps all cycles onto themselves. A 180° rotation about the other two coordinate axes exchange the a-cycles and $b-$cycles in each case. All the cycles are untwisted, which refers to the characteristics $\epsilon = ((0,0),(0,0))$. This "reverse engineering" is not the preferred course of action when one wants to model a surface. Usually one has a "qualitatively correct" picture in mind, like this one of four untwisted horizontal displaced cycles with the prescribed symmetry.

In general the negative Schottky data for the hyperelliptic examples are

$$S_{-,2n+1} = \left\{x, -x, y, A_1, -A_1, \mu_1, -\overline{A}_1, \overline{A_1}, \overline{\mu_1}, \ldots, A_n, -A_n, \mu_n, -\overline{A}_n, \overline{A_n}, \overline{\mu_n}\right\}$$

for odd genus $g = 2n+1$ and

$$S_{-,2n} = \left\{A_1, -A_1, \mu_1, -\overline{A}_1, \overline{A_1}, \overline{\mu_1}, \ldots, A_n, -A_n, \mu_n, -\overline{A}_n, \overline{A_n}, \overline{\mu_n}\right\}$$

for even genus $g = 2n$. The embedded examples must have characteristics

$$\epsilon_{2n+1} = ((1,0,\ldots,0),(0,\ldots,0)) \quad \text{and} \quad \epsilon_{2n} = ((0,\ldots,0),(0,\ldots,0)).$$

We have been able to determine the data for the embedded symmetric examples up to genus 6 (Tables 6 and 7). The displayed digits are all confirmed by the quasi-Newton deviation test and its convergence suggests that these surface all exist. The numerics also handles even higher genus than six, but unfortunately we have not yet been able to close the periods. For genus seven and eight the target energy is small enough for plotting convincing pictures, but the quasi-Newton deviation test does not converge. The minimization process for such high genus is laborious: the evaluation of the periods in the genus 6(8) case takes on average about 70 (450) seconds and the number of free parameters is 12 (16). It took weeks to determine the data for $\mathcal{H}e_6$, and it was only possible because we were able to generate good start positions for the minimization process.

The Schottky uniformization method enables us to extrapolate the Schottky data for higher genus examples from the known data of lower genus cases. To

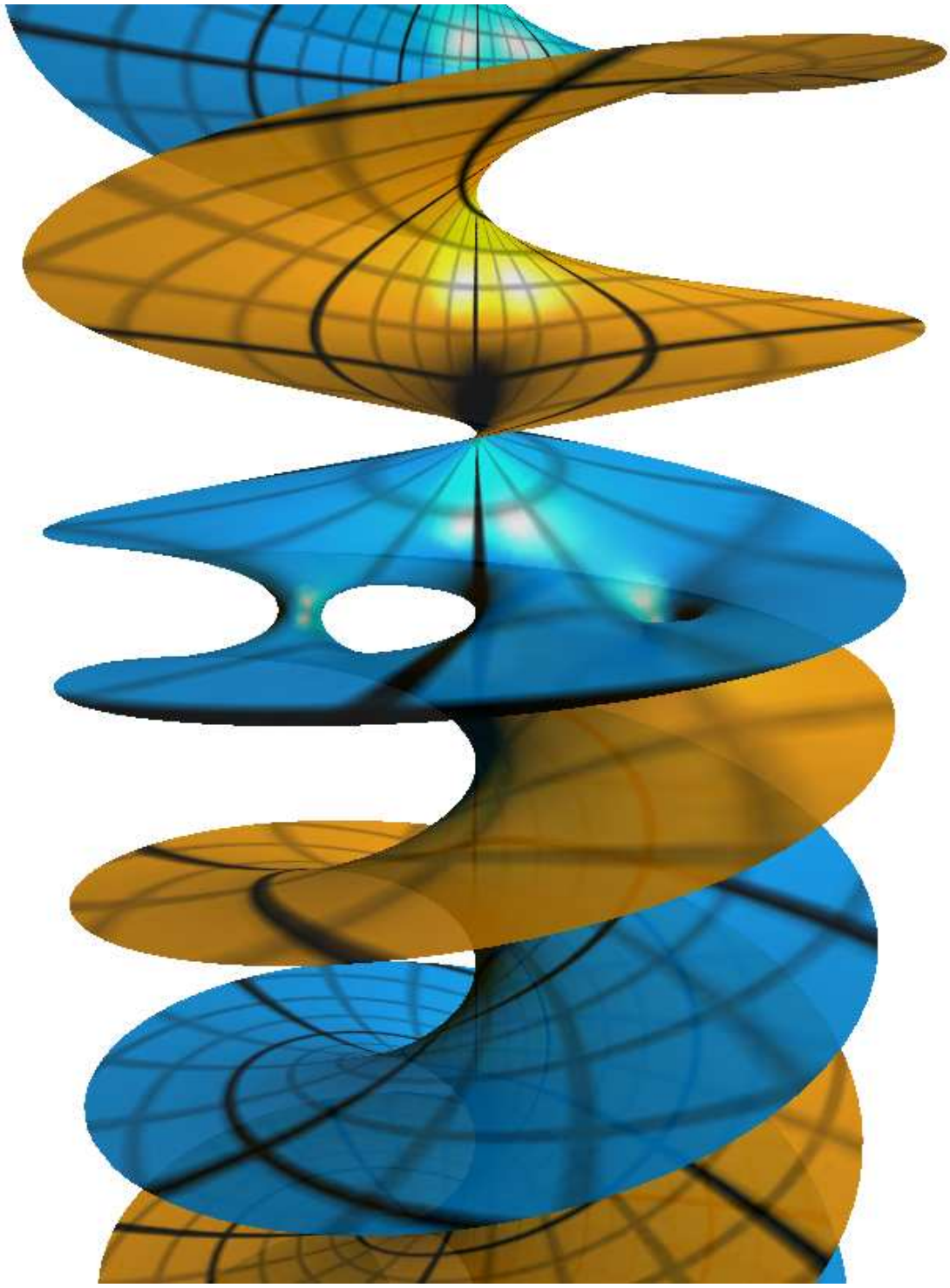
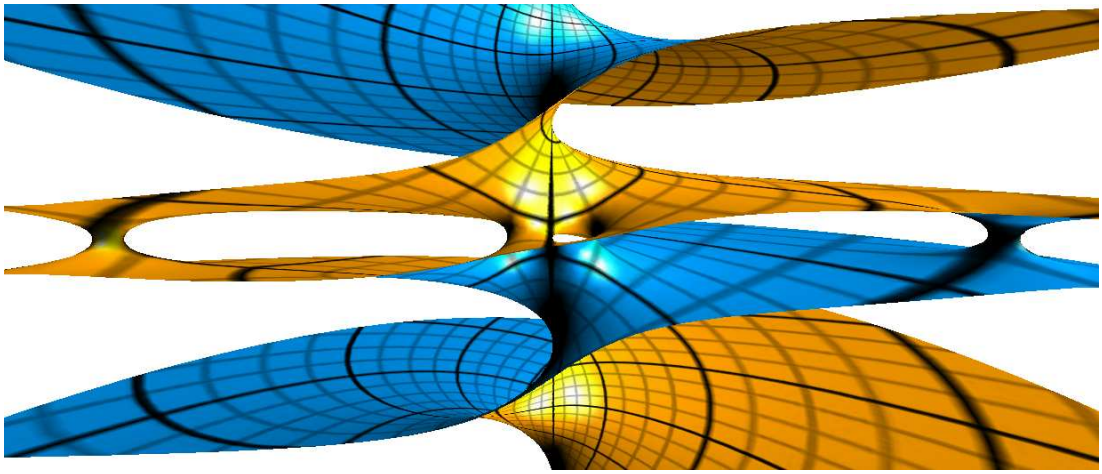
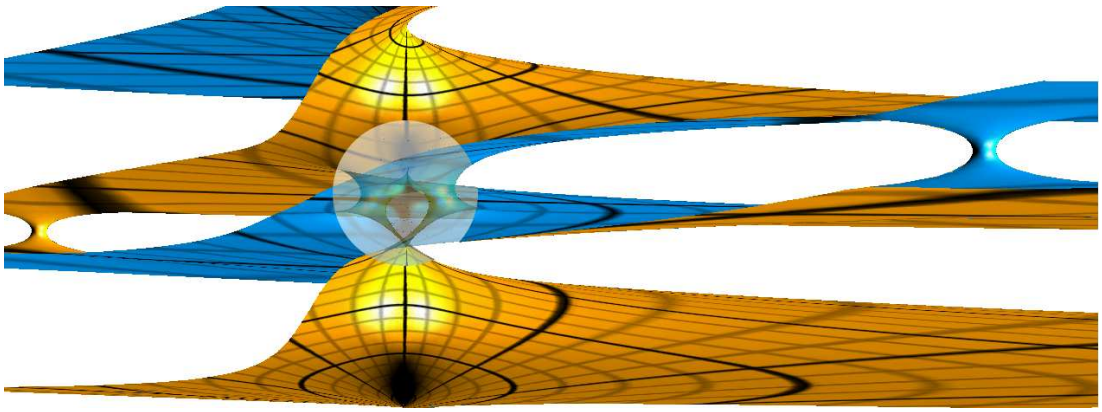


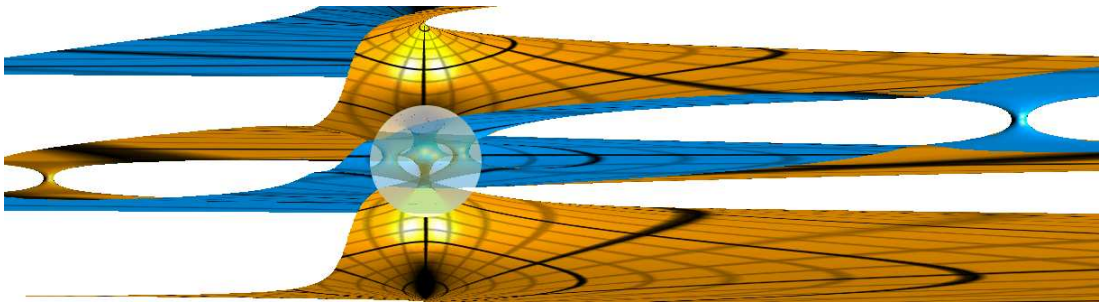
FIGURE 4.4.15. $\mathcal{H}e_2$, the embedded symmetric hyperelliptic helicoid of genus 2



(a) $\mathcal{H}e_3$



(b) $\mathcal{H}e_4$



(c) $\mathcal{H}e_5$

FIGURE 4.4.16. The embedded symmetric hyperelliptic helicoids of genus 3, 4, and 5.

g	x	A_1	A_2
1	1.473599588787236		
3	1.368669935	3.25927558 + 1.402595892 i	
5	1.3167	2.87 + 1.3521 i	5.31 + 1.5106 i
g	y	$10^6 \cdot \mu_1$	$10^9 \cdot \mu_2$
1	-0.026602460340538755		
3	-0.02798164608	2.0792494 - 1.4477405 i	
5	-0.028697	5.53 - 3.73 i	6.5 - 3.6 i

TABLE 6. Negative symmetric Schottky data for $\mathcal{H}e_1$, $\mathcal{H}e_3$, and $\mathcal{H}e_5$.

g	A_1	A_2	A_3
2	1.8997448266 + 1.3383277988 i		
4	1.73010822 + 1.27451238 i	4.1722873 + 1.5234329 i	
6	1.638957 + 1.2357420 i	3.646450 + 1.501396 i	6.16028 + 1.54885 i
g	$10^{-4} \cdot \mu_1$	$10^{-7} \cdot \mu_2$	$10^{-10} \cdot \mu_1$
2	0.877611203 - 1.056423685 i		
4	1.3708127 - 1.4972566 i	1.0581083 - 0.8072951 i	
6	1.736253 - 1.776168 i	3.72284 - 3.16433 i	6.1778 - 3.1884 i

TABLE 7. Negative symmetric Schottky data for $\mathcal{H}e_2$, $\mathcal{H}e_4$, and $\mathcal{H}e_6$.

extrapolate the data for genus g we take the data from the genus $g-2$ case and add suitable parameters $A_{[g/2]}$ and $\mu_{[g/2]}$. If we choose the parameter $\mu_{[g/2]}$ to be small[12], the associated generators have no effect on the Weierstrass spinors (4.1.3) and we just plugged four tiny holes into the helicoid at the images of $A_{[g/2]}$,$-A_{[g/2]}$,$\overline{A_{[g/2]}}$, and $-\overline{A_{[g/2]}}$. Out of these holes we will grow a new pair of handles, but we still have to decide where we want to "plant" them. The real part of $A_{[g/2]}$ encodes the distance of the new pair of handles from the z-axis. It is the only parameter that models the initial position. The imaginary part of $A_{[g/2]}$ can be determined by a variation process such that the holes have only vertical offsets. In a second pre-minimization step we vary the three real parameters $\Im A_{[g/2]}$, $\Re \mu_{[g/2]}$, and $\Im \mu_{[g/2]}$, which will let grow the holes toward eachother. In the third minimization step we vary all parameters and reach for the final solution. If $\Re A_{[g/2]}$ was chosen large enough the target energy is already small after this second step and we always get a visually convincing picture. For finding the right data, which masters the quasi-Newton deviation test, it takes

[12]Choosing the parameter $\mu_{[g/2]}$ to be zero has the effect that we identify the fix points $A_{[g/2]}$ with $-A_{[g/2]}$ and $\overline{A_{[g/2]}}$ with $-\overline{A_{[g/2]}}$, which means we add two nodes to the Riemann surface. Thus giving $\mu_{[g/2]}$ a very small value can be understood as opening up the nodes or as a numerical desingularization.

g	r_1	$r_{2,3}$	$r_{4,5}$
1	1.290		
3	0.701	0.485	
5		0.470	0.341

g	d_1	$d_{2,3}$	$d_{4,5}$
1	0		
3	0	20.208	
5	0	13.323	173.11

TABLE 8. Necksizes r_i and distances of the handle d_i from the center of $\mathcal{H}e_1$, $\mathcal{H}e_3$, and $\mathcal{H}e_5$.

g	$r_{1,2}$	$r_{3,4}$	$r_{5,6}$
2	0.689		
4	0.632	0.398	
6	0.595	0.393	0.304

g	$d_{1,2}$	$d_{3,4}$	$d_{5,6}$
2	5.726		
4	4.547	59.985	
6	3.990	34.558	447.86

TABLE 9. Necksizes r_i and distances of the handle d_i from the center of $\mathcal{H}e_2$, $\mathcal{H}e_4$, and $\mathcal{H}e_6$.

many approaches. The additional outer handles push the inner more towards the z-axis and let their necksizes shrink (Tables 8 and 9). If we have chosen the initial value for $\Re A_{[g/2]}$ too big, then new pair will not push the inner handles together, but if it is too small, we will disturb the lower genus picture and get no result at all. A successful approach has to balance these phenomena.

The distance of the the handles from the z-axis grows exponentially with the genus. In the geuns 6 case the most outer pair of handles is about 100 times further from the center than the inner most pair (Table 9).

Algebraic Data of the Hyperelliptic Examples. In the case of symmetric hyperelliptic examples, we can explicitly represent the Riemann surface as an algebraic curve. A hyperellitpic curve of genus g is given by the equation

$$\mu^2 = \prod_{j=1}^{2g+2} (\lambda - \lambda_j) \ . \tag{4.4.6}$$

In the hyperelliptic case with holomorphic involution $\pi(z) = -z$ the coordinate function λ can be written as

$$\lambda(z) = q \sum_{\sigma \in G} \sigma(z)^2 - \sigma(0)^2 \tag{4.4.7}$$

[**Bob91, BBE$^+$94**, p. 179]. The branch points λ_i are the fixed points of the hyperelliptic involution. In terms of the Schottky parametrization they are the images $\lambda_i = \lambda(\Lambda_i)$ of 0, ∞, or one of the $2g$ solutions of the equations

$$\hat{\sigma}_j(z) = \pi(z) = -z \,, \quad j = 1, \ldots, g \,,$$

	$g=2$	$g=4$	$g=6$
$b_1=\overline{b_2}$	1.482155 + 4.728121 i	1.048956 + 4.017513 i	0.853747 + 3.642826 i
$b_3=\overline{b_4}$	2.196811 + 5.445948 i	1.741124 + 4.811201 i	1.521140 + 4.470075 i
$b_5=\overline{b_6}$		15.059780 + 12.693695 i	10.991637 + 10.917349 i
$b_7=\overline{b_8}$		15.166960 + 12.735775 i	11.149793 + 10.989604 i
$b_9=\overline{b_{10}}$			35.570792 + 19.081307 i
$b_{11}=\overline{b_{12}}$			35.587271 + 19.085591 i

TABLE 10. Branch points of hyperelliptic curve for $\mathcal{H}e_2$, $\mathcal{H}e_4$, and $\mathcal{H}e_6$. Uncertain digits are printed in italics.

	$g=1$	$g=3$	$g=5$
$b_1=\overline{b_2}$	0.818063 + 2.543532 i	0.646490 + 2.237452 i	0.569868 + 2.090695 i
$b_3=\overline{b_4}$		7.634140 + 9.147499 i	5.448675 + 7.767172 i
$b_5=\overline{b_6}$		7.914972 + 9.300899 i	5.799157 + 7.991905 i
$b_7=\overline{b_8}$			25.146739 + 16.072406 i
$b_9=\overline{b_{10}}$			25.187059 + 16.084873 i

TABLE 11. Branch points of hyperelliptic curve for $\mathcal{H}e_1$, $\mathcal{H}e_3$, and $\mathcal{H}e_5$. Uncertain digits are printed in italics.

which is equivalent to

$$\left(z-\widehat{A}_j\right)^2=\widehat{\mu}_j\left(z+\widehat{B}_j\right)^2,\ \ j=1,\ldots,g\,,$$

with $S_{-,g}=\left\{\widehat{A}_1,\widehat{B}_1,\widehat{\mu}_1,\ldots,\widehat{A}_g,\widehat{B}_g,\widehat{\mu}_g\right\}$.

The scaling factor q in (4.4.7) can be chosen to be 1 because scaling the branch points does not change the Riemann surface represented by the curve 4.4.6. The solutions lie in pairs on the isometric cirlces $\hat{C}_j$ and $\hat{C}'_j$ which are identified by $\hat{\sigma}_j$ (Section 4.2). For small ismoetric circles the pairs of solutions are close to oneanother which carries over to the associated pair of branch points. In the case of negative symmetric Schottky data $S_{-,g}$ we have $\tau\circ\sigma_i=\sigma_j\circ\tau$,where $\tau(z)=\overline{z}$. The additional antiholomorphic involution yields conjugate pairs of solutions $\Lambda_i=\overline{\Lambda_j}$ as well as associated conjugate pairs $\lambda_i=\overline{\lambda_j}$ of branch points (Figure 4.4.17). Tables 11 and 10 lists the conjugate pairs of branch points for $\mathcal{H}e_g, g=1,\ldots,6$ derived from its Schottky data as described above.

4.4.4. The Nonhyperelliptic Case. We already mentioned that we did not find any closed nonhyperelliptic examples. We conjecture that any immersed minimal surface of finite topology with one helicoidal end can be parameterized by a hyperelliptic Riemann surface. We will present the analysis that motivated this conjecture. An other purpose of this section is to show that the Schottky

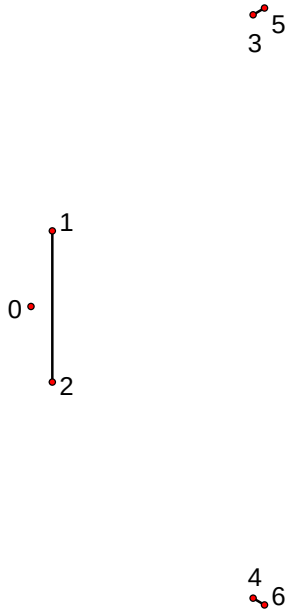


FIGURE 4.4.17. Branch point diagram of $\mathcal{H}e_3$. The branch cuts relate to the a-cycles of the Schottky basis.

uniformization method allows an uncomplicated treatment of nonhyperelliptic situations. In conlusion we add with Figures 4.4.20 and 4.4.21 two more episodes to the chronicle of the death of visual proof.

For genus $g = 2$ the only nonhyperelliptic Riemann surface is a twofold covered torus ($g_0 = 1, N = 0$) that is modeled by the positive symmetric Schottky data

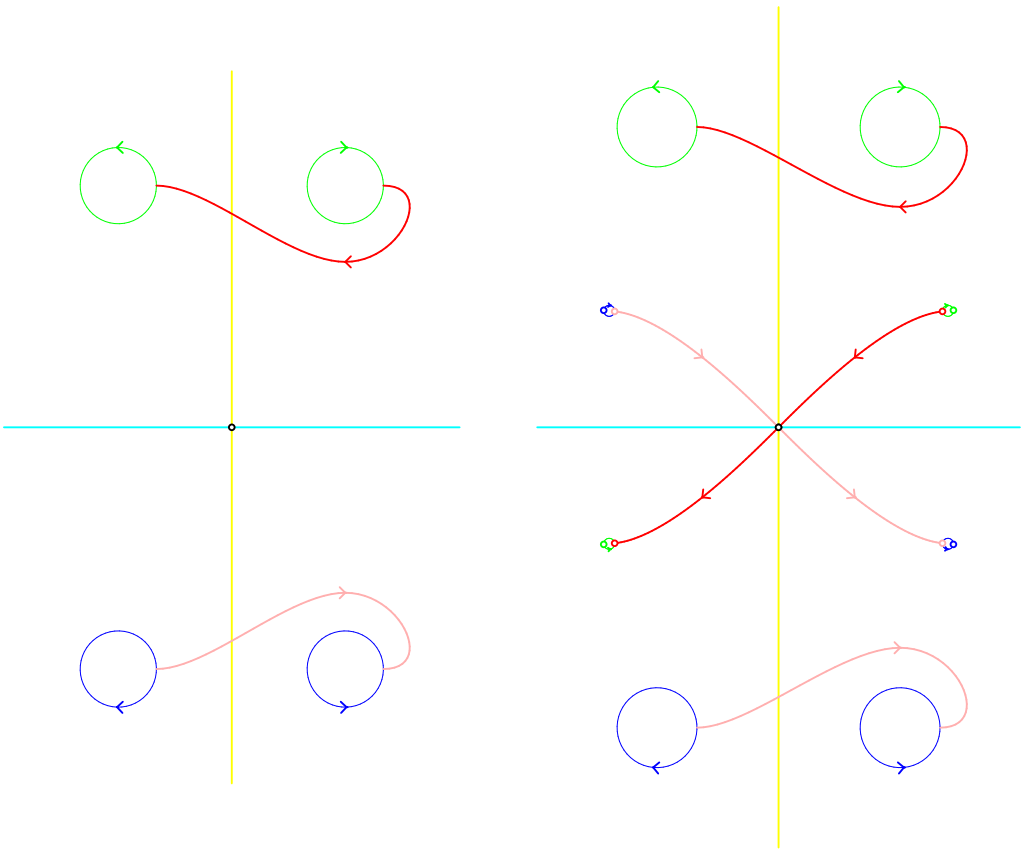
$$S_{+,1,1} = \left\{A, -\overline{A}, y, -A, \overline{A}, y\right\}$$

with $A \in \mathbb{C}, y \in \mathbb{R}$. On the left of Figure 4.4.18 we show the Schottky uniformization picture for the data $A = 1.50448 + 3i$, $y = -0.026722$. Figure 4.4.20 shows the associated helicoid with its two vertically displaced handles. Its period does not quite close. The error imposes translation periods of a magnitude of about a percent of the necksize of the handles, which is not noticeable any more in the picture. When we first tried to realize this example the minimizer almost immediately came up with a reasonable value for the target energy E. As we tried to improve this we noticed that the data did not converge, but slowly increased the imaginary part of A. To quantize this behavior we minimized the target energy E for fixed values of $\Im A$ and only varying $\Re A$ and y. The resulting graph $E\,(\Im A)$, shown in Figure 4.4.19 , confirmed our conjecture that the handles repel each other. $\Re A(\Im A)$ and $y\,(\Im A)$ converge to the corresponding values of the negative symmetric Schottky data S_- of the $\mathcal{H}e_1$as $\Im A$ goes to infinity, which is indeed not a surprise.

All nonhyperelliptic examples incorporate a vertical displacement of handles and in all our experiments[13], we had the same kind of repelling phenomena.

To illustrate how to model surface properties using Schottky uniformization we give another symmetric nonhyperelliptic example. On the right of Figure 4.4.18 we show the uniformization picture of a genus 4 example. We just "added" the Schottky data for the $\mathcal{H}e_2$ with the previous example. The resulting helicoid is shown in Figure 4.4.21. Although it is not noticeable in the picture, its periods do not close. A further minimization of the target function would send the outer pair of handles to infinity and leave us with the $\mathcal{H}e_2$.

[13]We also tested asymmetric examples.



(a) Cycles and symmetries for $S_{-,1,1}$ (b) Cycles and symmetries for $S_{-,1,2,1}$

FIGURE 4.4.18. Uniformization pictures of nonhyperelliptic examples

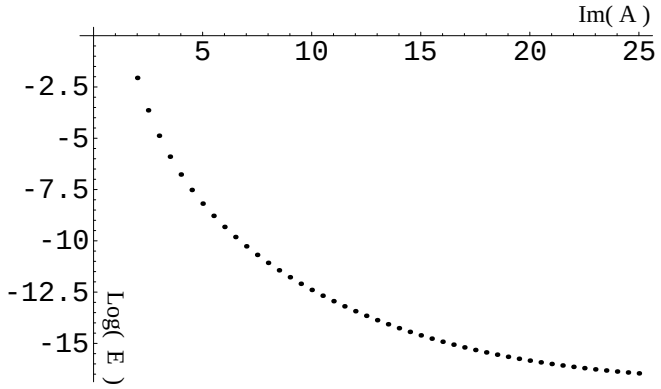


FIGURE 4.4.19. The target energy E as a function of $\Im A$.

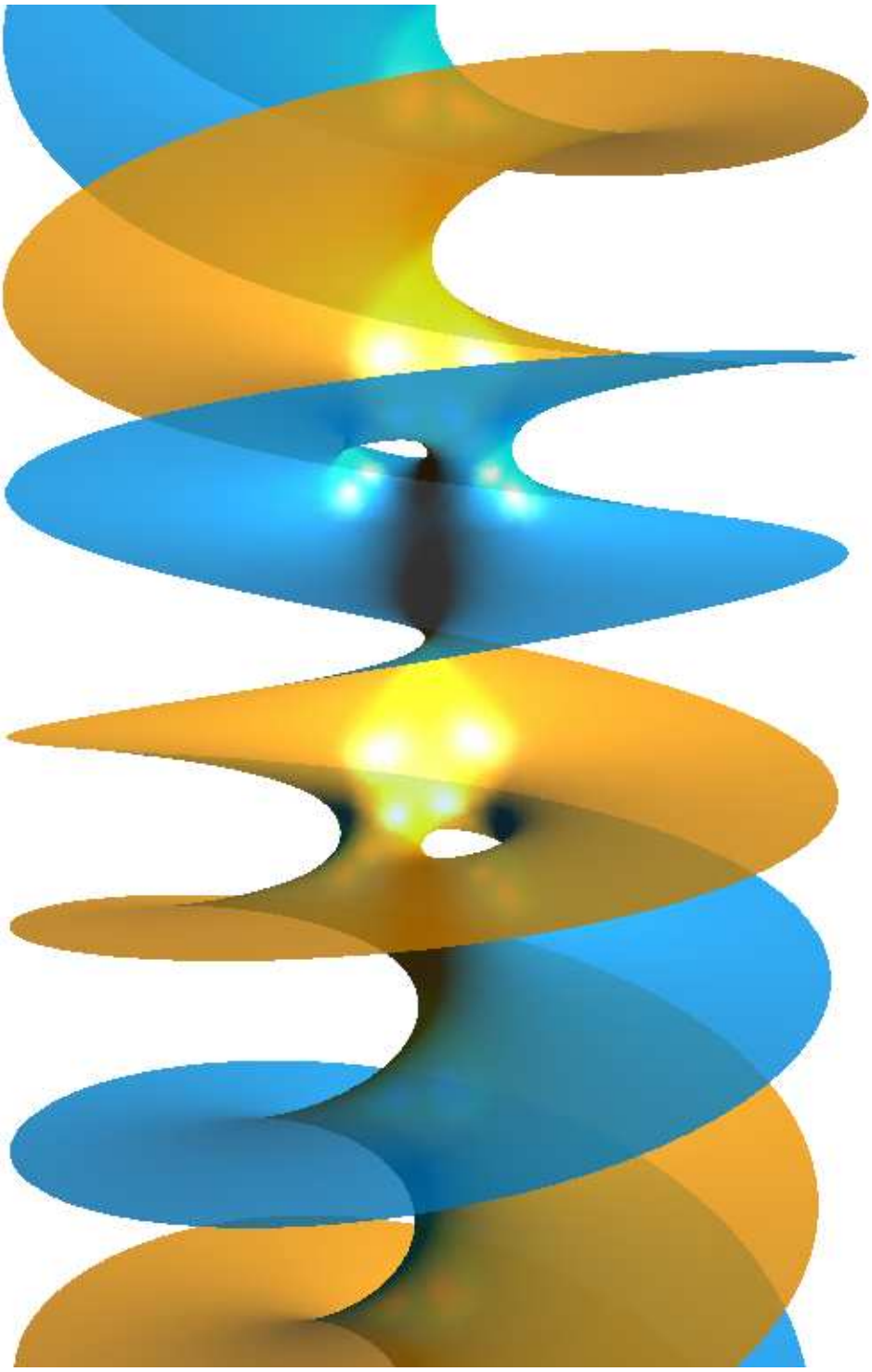


FIGURE 4.4.20. Nonexisting nonhyperelliptic symmetric genus 2 example.

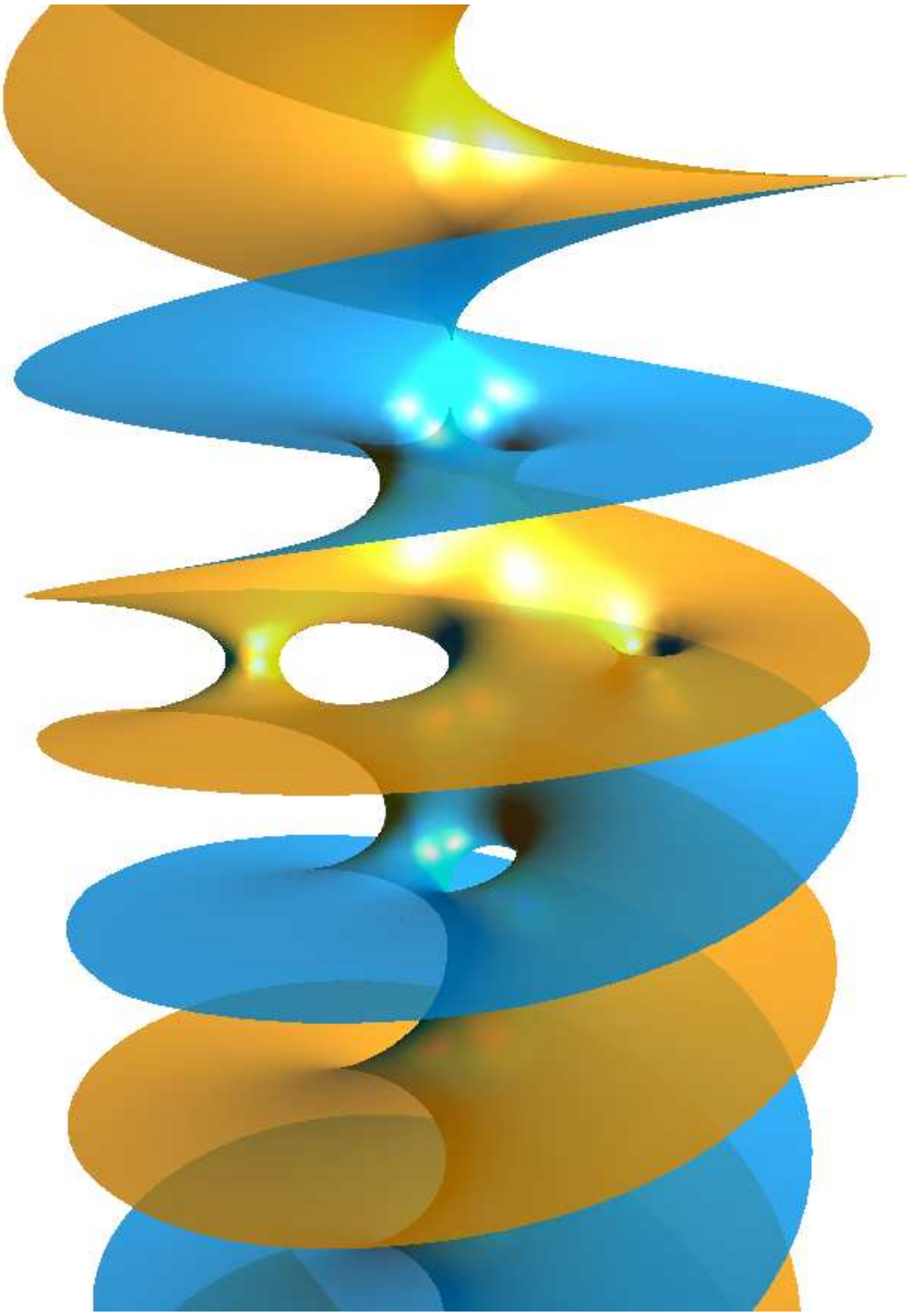


FIGURE 4.4.21. Nonexisting nonhyperelliptic symmetric genus 4 example.

CHAPTER 5

jtem - Numerical Libraries

5.1. Introduction

The core of the *jtem*[1] project consists of a collection of scientific, mostly mathematical, libraries implemented in pure standard Java[2]. The project is develeoped according to the following major guidelines:

(1) using only pure Java as programming language, and
(2) separating applications and the application generation process from the mathematical content.

These decisions were driven by the experiences we gained from previous work. From 1994 to 1997 we developed a prototyping environment for experimental mathematics: *Oorange* (Object ORiented Analysis Numerics and Graphics Environment) [**GOP$^+$97**]. The *Oorange* project had a hybrid language scheme: numerics and graphics were implemented in *Objective-C*; the graphical programming environment used *Tcl* for scripting purposes. To accomplish our demanding goals more computer languages and non main stream software had to be incorporated. In combination with the approach to develop a generic tool designed for several different tasks the project came to the edge of maintainability.

Java offered main stream solutions for all major technical problems we had to deal with in the original *Oorange* project. This enabled us to focus on contents, which we broke up into three seperated projects: *Java Oorange*[3], *jreality*[4], and *jtem*.

With *Java Oorange* we refined the concepts for graphical programming and rapid prototyping developed for the original Oorange.

jReality, a Java 3D viewer for mathematics, took over the part of the 3d graphics system.

jtem provides a collection of numerical libraries for the scientific and mathematical content. The project currently offers almost a dozen different subprojects, but its core and origin are three numerical libraries*: numericalMethods*, *mfc*, and *riemann*, which are the subject of this chapter.

[1] http://www.jtem.de
[2] Java is an object oriented programming language and trademark of Sun Microsystems.
[3] http://www.oorange.de
[4] http://www.jreality.de

5.2. Design of the JTEM - Numerical Libraries

The basic design consists of two major layers, which divides the code into two, for mathematicians natural, categories:

(1) algorithms, and
(2) mathematical objects

These categories reflect also two basic approaches in software design: procedural and object oriented. The design of the *jtem* libraries tries to bring these two concepts together in order to achieve maximal flexibility.

The bottom layer is formed by a library of algorithms and numerical procedures bundled in the project *numericalMethods*. The implementation strives for a pure procedural realization by static methods using only primitive types, arrays of primitive types, or Java foundation classes. Thes functions have a high reusabiltiy value if they avoid internal dependencies. Firstly, they can be referenced by others without imposing any changes to their code. Secondly, the code of such methods can be easily incorporated in different projects. This allows people to avoid the dependency to our project, or to enable them to adapt the code to their needs.

It is clear that such a pure approach can not be kept up in an object oriented language like Java without counteracting its design. Very often it is not sensible or even possible to avoid classes and interface. But the definition of classes and interfaces is kept as local as possible avoiding any kind of fancy hierarchies in order to maximize the reusability value as described above.

On the other hand, the ascetic design helps to incorporate code from others as the project *numericalMethods* imposes almost no restrictions to the code. For many algorithms implementations exist in C or FORTRAN. In most situations the only way to incorporate the sources into a pure Java project[5] is to translate them manually. This can be a tedious task and it helps a lot to stick to the blueprint, which has normally a procedural design, as close as possible.

The second layer consist of a hierarchy of libraries providing implementations of mathematical terms in a classical object oriented manner. The spectrum of classes reach from basic terms like complex numbers, vectors, and matrices to highly sophisticated notions like Riemann surfaces. Many of these classes just wrap the functionality of the *numericalMethods* package and provid a convenient front-end. This seems to generate an unnecessary overhead, but it keeps the front-end simple. Numerical routines tend to have many tuning parameters, which need specialized knowledge about the algorithm. This should be not visible in the API of the easy-to-use classes. If an application needs special treatment, i.e., the default setup does not suffice, the developer can directly use the hard core routines in the *numericalMethods* package.

Currently there are four numerical libraries available to the public:

[5]This excludes native libraries or special virtual machines.

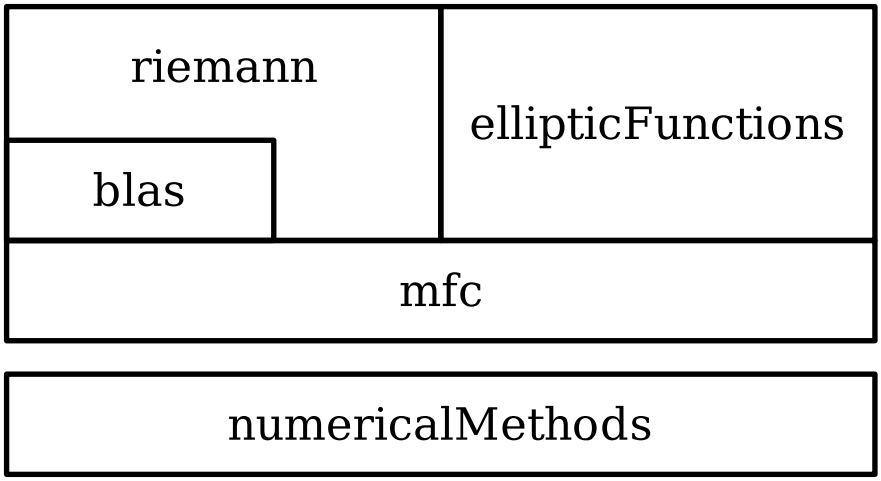


FIGURE 5.2.1. A layer diagram showing the dependencies of the jtem numerical libraries.

(1) mfc - mathematical foundation classes,
(2) blas - basic linear algebra system,
(3) riemann - tools for the study of riemann surfaces, and
(4) ellipticFunctions - a collection of elliptic Functions.

Figure 5.2.1 shows their dependencies. In the following sections we give description of these projects.

5.3. The numericalMethods project

The numericalMethods package is the foundation of the numerical jtem-projects. It is a library of numerical routines solving algebraic, analytic and geometric problems which is reflected in its three main sub-packages *numericalMethods.algebra*, *numericalMethods.calculus,* and *numericalMethods.geometry*. The design of the code follows a procedural approach typical for numerical routines (Section 5.2). The main packages have very few inter-dependencies. All of them are caused by references to algebra routines, which is a necessity.

5.3.1. numericalMethods.algebra. The algebra package currently provides numerical routines in three topics: groups, polynomials, and linear algebra. Figure 5.3.1 shows an UML class an package diagram[6]. Unlike FORTRAN, Java has no complex primitive type. For complex arithmetic one either has to provide a class or simply work with existing primitve types. We decided for the last and provide no class implementation of a complex number in the numericalMethods project[7] for several reasons:

The existence of a complex type would unstoppably cause a dependency of huge part of the project on this type and heavily reduce its reusability. Further, the complex type would certainly appear in the API and cause also the user of the library to use it. Even if complex numbers are widely used, not ever implementation is useful for every application. Even the realization of such a trivial notion as a complex number involves several basic design decisions. A necessity for one application can be a restriction for an other. As an example we raise the question whether a complex number should be mutable[8], so we can reuse the instances and relieve the garbage collector[9], or should it be better immutable so we can share instances safely.

Another reason is that numerical routines should neither force nor support the creation of objects, because this can reduce the performance drastically. Representing each entry of a large complex matrix by an object would be an example for a particularly bad design. Thus in the linear-algebra packages a complex matrix is always represented by two real matrices, its real and imaginary part.

The linear-algebra routines are the working horses for many applications, thus their implementation should be especially efficient. Vectors and matrices are

[6]UML (unified modeling language) diagram types [**FS97**].

[7]An implementation of complex number is provided in the *mfc* project, *mfc.field.Complex* and *mfc.field.ComplexConstant.*

[8]immutable objects can not be changed, mutable can.

[9]Java has a build-in memory management: allocated memory can not be freed manually; it is done automatically by the "garbage collector". The garbage collector analysis all currently existing objects in the virtual machine and determines those objects which can not be referred anymore by the program and frees them.

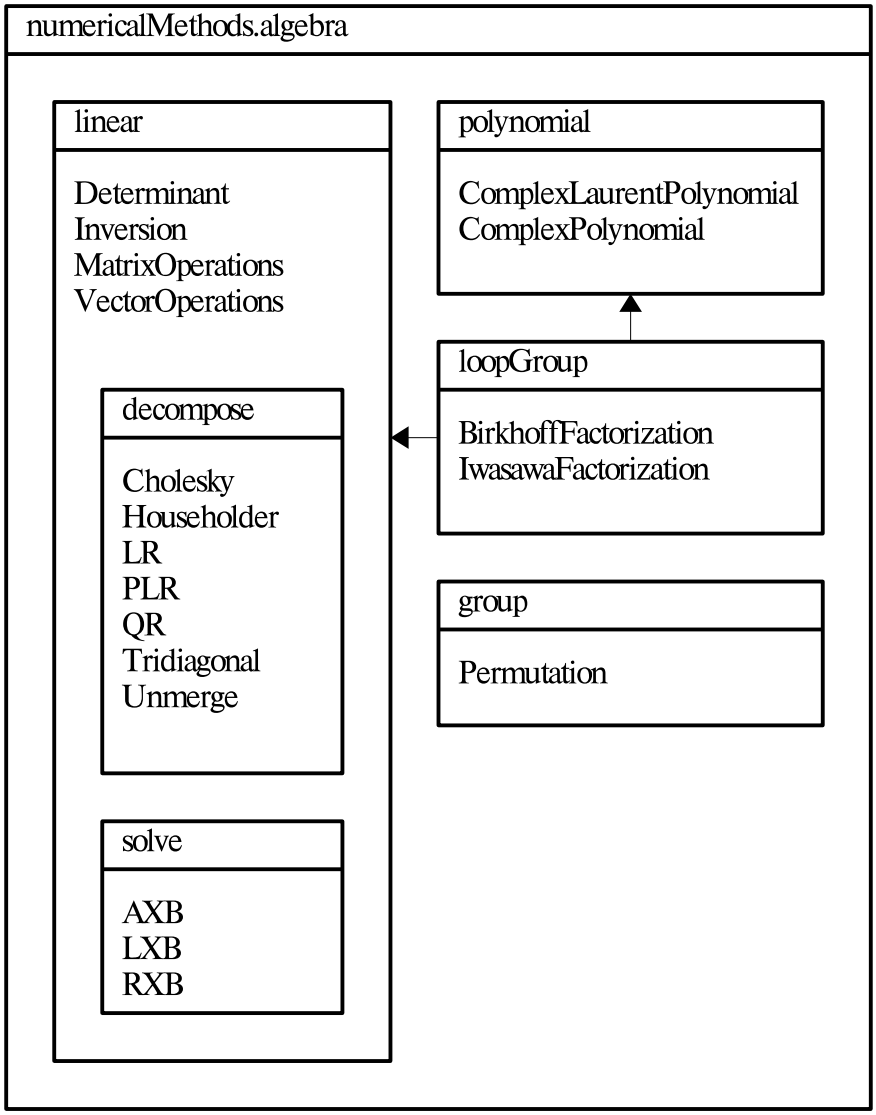


FIGURE 5.3.1. UML diagram of classes and packages in *numerical.algebra*.

represented as one-dimensional and rectangular two-dimensional arrays[10]. Due to performance reasons the routines never check whether a two-dimensional array is rectangular. This responsibility is left to the application programmer[11].

Unnecessary object creation can significantly reduce the performance of an application. Many algorithms need to create storage for temporary data. The API offers for those algorithms for which this is significant the possibility to pass

[10]The first index represents the row, the second the column. This is standard, because it is optimal for matrix-vector multiplications.

[11]The utility class *numericalMethods.algebra.linear.MatrixOperations* provides a static methods *boolean isRectangular(double[][])* for this task.

the temporary data to it. This is important in situations where you call this function with similar data for many times. In situations where this is not an issue you might prefer to use the method with an easy-to-use signature, which is always provided.

An example: Inverting a real matrix. This task is handled by two methods in the class numericalMethods.algebra.linear.Inversion:

(1) static boolean compute(double[][] A, double[][] I)
(2) static boolean compute(double[][] A, double[][] t0, double[] t1, double[] t2)

The signature of the first method is almost self-explanatory: the method computes the inverse of the matrix *A* and returns the result in the matrix *I*; if the computation was not successful, i.e., *A* is (numerically) singular, the methods returns *false* otherwise *true*. The implementation assumes that both arguments are consistent. The second signature is the front-end to the core algorithm and more complicate: firstly, there are the arguments *t0,t1,t2* for the temporary storage, they may be *null,* which forces the routine to create the data. Secondly, it lacks the argument for the result. It is quite common that numerical procedures destroy the input or, like here, overrides the input with the output. This implies the necessity to copy the argument beforehand if it still needed.

Below we give an example implementation; a method that inverts an array of matrices in-place which have all the same shape:

```
public void invert( double [][][] listOfSquaredMatrices ) {
     final int n = listOfSquaredMatrices[0].length;
     final double [][] t0 = new double[n][n];
     final double []   t1 = new double[n];
     final double []   t2 = new double[n];
     for( int i=0; i<listOfSquaredMatrices.length; i++ ) {
           final int A = listOfSquaredMatrices[i];
           numericalMethods.algebra.linear.
                Inversion.compute( A, t0, t1, t2 );
     }
}
```

5.3.2. numericalMethods.calculus. The calculus sub-package is the most extensive among the main package of the numericalMethods project. It currently provides routines for integration and differentiation, function minimization, root finding, function interpolation and approximation, and some special functions (Figure 5.3.2). It is the nature of this topic that it deals with functions, thus it is neither surprising nor avoidable that most of the sup-packages depend on the package which provides their definitions:

5.3.2.1. *numericalMethods.calculus.function.* Interfaces impose only relative small restrictions to the user code. They can easily be added to or removed from code and with the use of anonymous classes they can be even defined locally. The

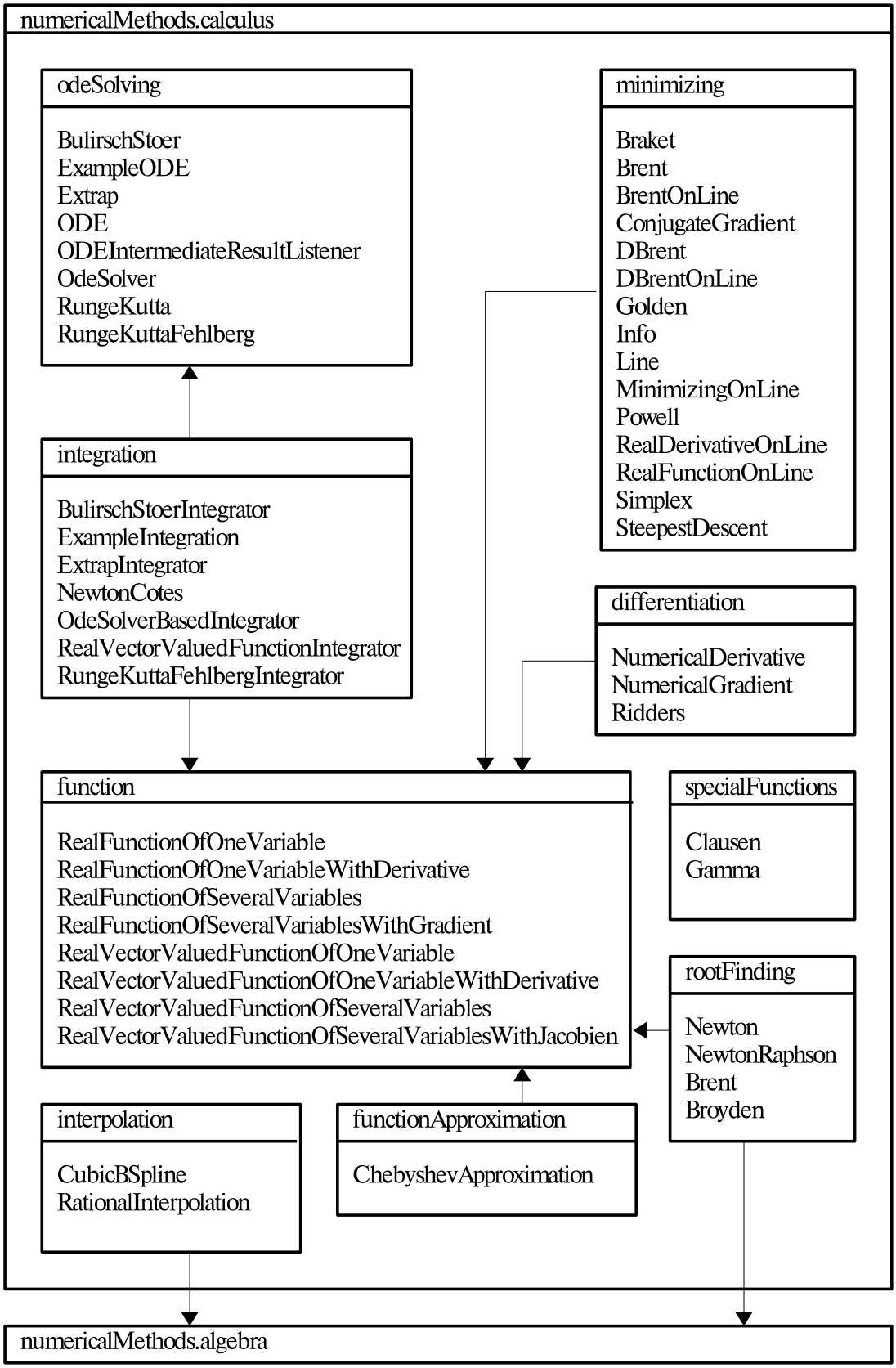


FIGURE 5.3.2. UML diagram of classes and packages in *numerical.calculus*.

function package provides interfaces for real functions for four different types with and without derivative information:

(1) *RealFunctionOfOneVariable* ($f : \mathbb{R} \to \mathbb{R}$),
(2) *RealFunctionOfSeveralVariables* ($f : \mathbb{R}^n \to \mathbb{R}$),
(3) *VectorValuedFunctionOfOneVariable* ($\gamma : \mathbb{R} \to \mathbb{R}^n$), and
(4) *VectorValuedFunctionOfSeveralVariables* ($F : \mathbb{R}^n \to \mathbb{R}^n$).

The interfaces that provides additional derivative information extend those without and are respectively called

(1) *RealFunctionOfOneVariableWithDerivative*,
(2) *RealFunctionOfSeveralVariablesWithGradient*,
(3) *VectorValuedFunctionOfOneVariableWithDerivative*, and
(4) *VectorValuedFunctionOfSeveralVariablesWithJacobian* .

The length of the indeed very descriptive names are no handicap, because for modern IDE[12] name-completion is a standard feature.

To show how these interfaces work we give an example implementation for

$$f(x, y) = \sin(x) \cos(y) , \tag{5.3.1}$$

which is a function of type 2. The interface *RealFunctionOfSeveralVariables* consists of two methods

(1) *public double eval(double[] x)*, and
(2) *public int getNumberOfVariables().*

The interface is self-explanatory; the code below creates an instance of an anonymous class implementing the interface.

```
RealFunctionOfSeveralVariables f = new RealFunctionOfSeveralVariables() {
    public double eval(double[] x) {
        double s = x[0];
        double t = x[1];
        return Math.cos(t)*Math.sin(s);
    }
    public int getNumberOfVariables() {
        return 2;
    }
};
```

The interface *RealFunctionOfSeveralVariablesWithGradient* extends the previous interface by the single method*:*

public double eval(double[] x, double[] gradient).

The method returns the function value at the position ***x*** and computes the gradient of the function which is written into *gradient.* An implementation for

[12]IDE - Integrated Developer Environment.

the function f from above with

$$\nabla f = \begin{pmatrix} -\sin(t)\,\sin(s) \\ \cos(t)\,\cos(s) \end{pmatrix}$$

could look like

```
public double eval(double[] x, double[] gradient ) {
     double s = x[0];
     double t = x[1];
     gradient[0] = -Math.sin(t)*Math.sin(s);
     gradient[1] =  Math.cos(t)*Math.cos(s);
     return Math.cos(t)*Math.sin(s);
}
```

5.3.2.2. *numericalMethods.calculus.differentiation.* In many cases the derivative is very hard to compute analytically and a numerical approximation is the only choice to gain derivative information. This always involves specific knowledge about the function. We are using an algorithm given by Ridders[13] [**Rid, PTVF92**] based on the general idea of "Richardson's deferred approach to the limit". Ridders' method is especially robust and also provides error estimates for the derivative. For an easy generation of numerical derivative information the package provides a factory class: *NumericalDerivative*. Is `f` an instance of any of the four function interfaces, then

`NumericalDerivative.create( f, h, maxTableLength )`

creates an instance of the related interface with derivative information which computes the differential using Ridders method[14].

[13] *numericalMethods.calculus.differentiation.Ridders*

[14] The parameters `h` and *`maxTableLength`* determine the initial step-size and the maximal length of the extrapolation table, consult the class documentation or [**Rid, PTVF92**] for details.

5.3.2.3. *numericalMethods.calculus.minimizing.* This was the first package of all, which already shows the importance for our work. Unfortunately, this can also be seen in the code and thus the hole package will soon be subject to a major refactoring. The package provides various methods for one- and multi-dimensional minimization. Apart form the downhill simplex method of Nelder and Mead [**NM65**], which has an entirely self-contained strategy, all other algorithms reduce multidimensional minimization problems to the one-dimensional case. The algorithms finally differ merely in ther choce of the directions for their next *line search*. The steepest descent method and the conjugate gradient method use derivative information for this. However, Powell's directions set method [**Bre02**] and the downhill simplex method of Nelder and Mead [**NM65**] require only function evaluations.

All one-dimensional minimum search algorithms assume that you are able to bracket the minimum, i.e., you can provide positions a, b, c such that the function values at a and c are bigger the value at b. Finding the initial bracket is therefore always the first step in any minimum search, which is done by a bracketer. The minimizing algorithms separate into three different categories:

(1) bracketer,
(2) one-dimensional, and
(3) multidimensional search algorithms.

A choice of a multidimensional method also involves a choice of a one-dimensional search algorithm and a bracketer. Which combination is best depends on the target-function.

Below we give an example implementation for minimizing the target-function

$$f(s,t) = \cos(t)\,\sin(s)\ ,$$

which we already used in Section 5.3.2.1 to illustrate the function interfaces, with several different minimizers. The code below uses a typical alternative implementation, avoiding code doublication:

```
package helicoid.example;
import numericalMethods.calculus.minimizing.*;
public class FunctionMinimizingWithDerivatives {
     RealFunctionOfSeveralVariablesWithGradient
           f = new RealFunctionOfSeveralVariablesWithGradient() {
           public double eval(double[] x, double[]
           gradient) {
                double s = x[0];
                double t = x[1];
                if( gradient != null ) {
                  gradient[0] =
                  -Math.sin(t)*Math.sin(s);
                  gradient[1] =
                  Math.cos(t)*Math.cos(s);
                }
                return Math.cos(t)*Math.sin(s);
           }
           public double eval(double[] x) {
                return eval( x, null );
           }
           public int getNumberOfVariables() {
                return 2;
           }
     };
     /** minimizes function f(s,t)=cos(t)sin(s) with prescribed error tolerance
     * and start position using Powell's direction set methods. */
     public double minimizeByPowell( double[] startPos, double tol ) {
           return Powell.search(startPos, tol, f );
     }
     /** minimizes function f(s,t)=cos(t)sin(s) with prescribed error tolerance
     * and start position using the downhill simplex method of Nelder & Mead. */
     public double minimizeByNelderMead( double[] startPos, double tol ) {
           return NelderMead.search(startPos, tol, f );
     }
     /** minimizes function f(s,t)=cos(t)sin(s) with prescribed error tolerance
     * and start position using the conjugate gradient method. */
     public double minimizeByConjugateGradient( double[] startPos, double tol ) {
           return ConjugateGradient.search( startPos, tol, f );
     }
     /** minimizes function f(s,t)=cos(t)sin(s) with prescribed error tolerance
     * and start position using the steepest descent method.
     public double minimizeBySteepestDescent( double[] startPos, double tol ) {
           return SteepestDescent.search( startPos, tol, f );
     }
}
```

5.3.2.4. *Root Finding and Nonlinear Equations.* Root finding in several dimensions is always a challenge. There are just no good general solutions for this problem and it is very likely that there will never be any [**PTVF92**, page 379]. However, with an adoption of Newton's method[15] and Broyden's multidimensional secant method[16] we provide two globally convergent methods for nonlinear systems of equations. Both methods need derivative information as they translate the root finding problem into a minimizing problem and perform a steepest descent step as long they are not close to a solution.

5.3.2.5. *numericalMethods.calculus.odeSolving.* This package provides four algorithms for the integration of ordinary differential equations (ODEs): Runge-Kutta[17], Runge-Kutta-Fehlberg[18] (RKF), Bulirsch-Stoer[19] (BS), and the Extrap[20] ODE-solver by E. Hairer and G. Wanner [**HNW93**]. BS and Extrap are adaptive solvers using an extrapolating scheme. They allow dynamic changes of the integration order. For analytic functions, these methods are superior to single-step-methods[21] like RKF. However, single-step-methods are more robust in discontinuities. Only the classical Runge-Kutta has a fixed step size and no error control. It is useful in situations where precision is less important then performance, as in a graphics application.

A generic problem in ODEs can be reduces to the study of a set of N coupled first-order differential equations

$$\frac{d}{dt} y_i(t) = f_i(t, y_1(t), \ldots, y_N(t)) ,$$

for the functions $y_i : \mathbb{R} \to \mathbb{R}$, with $i = 1, \ldots, N$, where the functions $f_i : \mathbb{R} \times \mathbb{R}^N \to \mathbb{R}$ are given. Similar to the function interfaces in Section 5.3.2.1 is the interface `ODE` defining and ordinary differential equation. It consists of two methods

(1) `public int getNumberOfEqations()`, and
(2) `public void eval(double t, double[] y, double [] f ).`

The first method is self-explanatory. For the given time `t` and position `y,` the second method provides the values of the functions f_i, which are written into `f`.

As an example we like to give the implementation of the mathematical pendulum: $\ddot{x} = -\sin x - c\,\dot{x}$. This second order ODE is equivalent to

$$\begin{aligned} \dot{x} &= v , \\ \dot{v} &= -\sin x - c\,v , \end{aligned}$$

[15] *numericalMethods.calculus.rootFinding.Newton*
[16] *numericalMethods.calculus.rootFinding.Broyden*
[17] *numericalMethods.calculus.odeSolving.RungeKutta*
[18] *numericalMethods.calculus.odeSolving.RungeKuttaFehlberg*
[19] *numericalMethods.calculus.odeSolving.BulirschStoer*
[20] *numericalMethods.calculus.odeSolving.Extrap*
[21] Unfortunately we do not have multi-step methods yet [**SB90**, p. 113ff].

which is a first order ODE in the phasespace (x, v). An implementation could like:

```
ODE pendulum = new ODE() {
     double c = 42;
     public double eval( double t, double[] y, double [] f) {
            double x = y[0];
            double v = y[1];
            f[0] = v;
            f[1] = -Math.sin(x)-c*x;
     }
     public int getNumberOfEquations() {
            return 2;
     }
};
```

To solve an ODE using one of the adaptive solvers you can simply call the static method

solve(ODE ode, double [] y, double t0, double t1, double tol),

which is defined for all the corresponding classes. The method integrates the ODE *ode* with the initial values given in *y* from *t0* to *t1*. On output *y* contains the values at time *t1* which should lie within the error tolerance *tol*. To solve the mathematical pendulum with given initial values *x* and *v* at time *t0* for the final time *t1* and an error tolerance *tol* using the Extrap solver one writes:

```
double [] y = new double[] {x,v};
Extrap.solve( pendulum, y, t0, t1, tol );
```

Figure 5.3.3 shows a snap-shot of a webstart application visualizing planar vector fields. The application was generated using *Java Oorange*[22] and uses different ODE solvers of this package. This and other applications using the jtem library can be downloaded from *http://www.math.tu-berlin.de/geometrie/lab/* .

[22] *Java Oorange* is a rapid prototyping environment for scientific software; *http://www.oorange.de* .

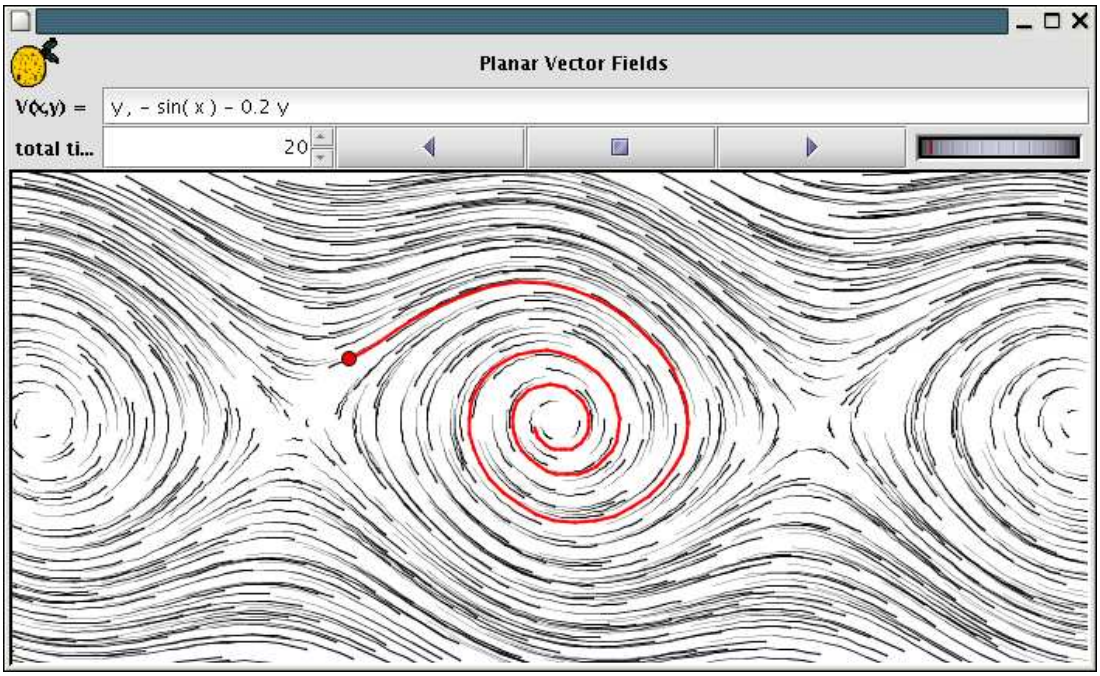


FIGURE 5.3.3. Snap-shot of a webstart application visualizing planar vector fields. The application can be downloaded from *http://www.math.tu-berlin.de/geometrie/lab/calculus.shtml* .

5.3.2.6. *numericalMethods.calculus.integration.* Numerical integration is a well-studied field and thus there are many algorithms available. Among these are the classical formulas for equally spaced abscissas: the closed Newton-Cotes formulas. The class *NewtonCotes* provides all classical quadrature formulas up to order 9. In situation when you have specific information about the function or the integral's accuracy is less significant the Newton-Cotes formulas can be a good choice, because the integration method incorporates almost no additional costs. However, in many situations the fixed step-size is a huge drawback. Thus, modern methods always use a *variable* or *adaptive* step-size which automatically incorporates an error prediction. The algorithms in the *odeSolving* package use this concepts and since integration is just the easiest case of an ODE, we can provide efficient integrators based on the adaptive ode solvers.

The example code below integrates the holomorphic Weierstrass data

$$\begin{aligned} g &= \exp z, \\ \zeta &= i\,dz \end{aligned}$$

representing the simple helicoid. For the complex arithmetic it uses the class *Complex* of the *mfc* project, see Section 5.6 for details.

```
package helicoid.example;
import numericalMethods.calculus.integration.ExtrapIntegrator;
import numericalMethods.calculus.function.RealVectorValuedFunctionOfOneVariable;
import mfc.field.Complex;
public class HelicoidIntegration {
     final Complex P = new Complex();
     final Complex Q = new Complex();
     RealVectorValuedFunctionOfOneVariable helicoid
           = new RealVectorValuedFunctionOfOneVariable() {
           public int getNumberOfFunctions() {
                return 3; // three coordinate functions to integrate
           }
           public void eval( double x, double[] values, int offset ) {
                Complex z     = P.plus( Q.minus(P).times(x) );
                Complex g     = z.exp();
                Complex zetta = Complex.I;
                Complex phi1  = g.invert().minus( g ).times( zetta );
                Complex phi2  = g.invert().plus ( g ).times( zetta ).timesI();
                Complex phi3  = zetta.times(2);
                values[ offset     ] = phi1.re;
                values[ offset + 1 ] = phi2.re;
                values[ offset + 2 ] = phi3.re;
           }
     };
     /**
     * Integrates Weierstrass data on segment given
     * by endpoints P and Q with prescribed precision eps.
     */
     public double [] integrate( Complex P, Complex Q, double eps ) {
           this.P.assign(P);
           this.Q.assign(Q);
           double [] integral = new double[3];
           ExtrapIntegrator.integrate( integral, helicoid, 0, 1, eps );
           return integral;
     }
}
```

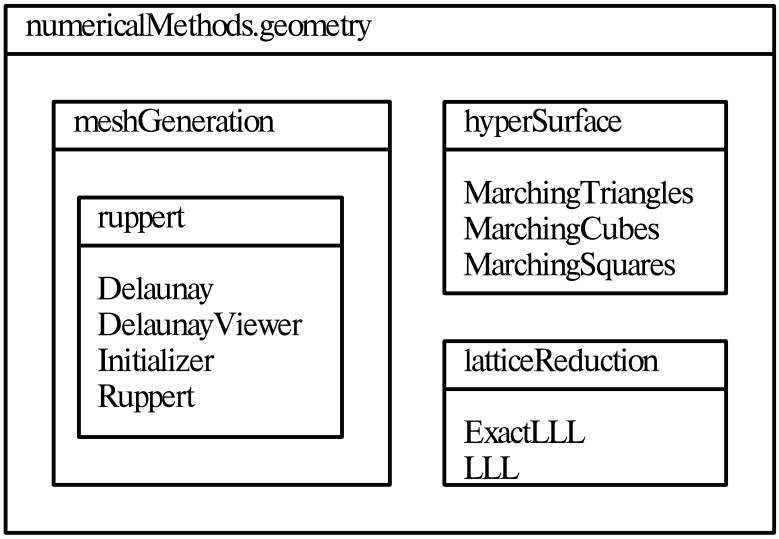


FIGURE 5.3.4. UML diagram of classes and packages in *numerical.geometry*.

5.3.3. numericalMethods.geometry. This package is a potpourri of routines and solutions for geometrical tasks. Unfortunately, only three of this packages have been released so far (Figure 5.3.4). Two of these, *meshGeneration* and *latticeReduction*, are of particular importance for the computation of helicoids with handles.

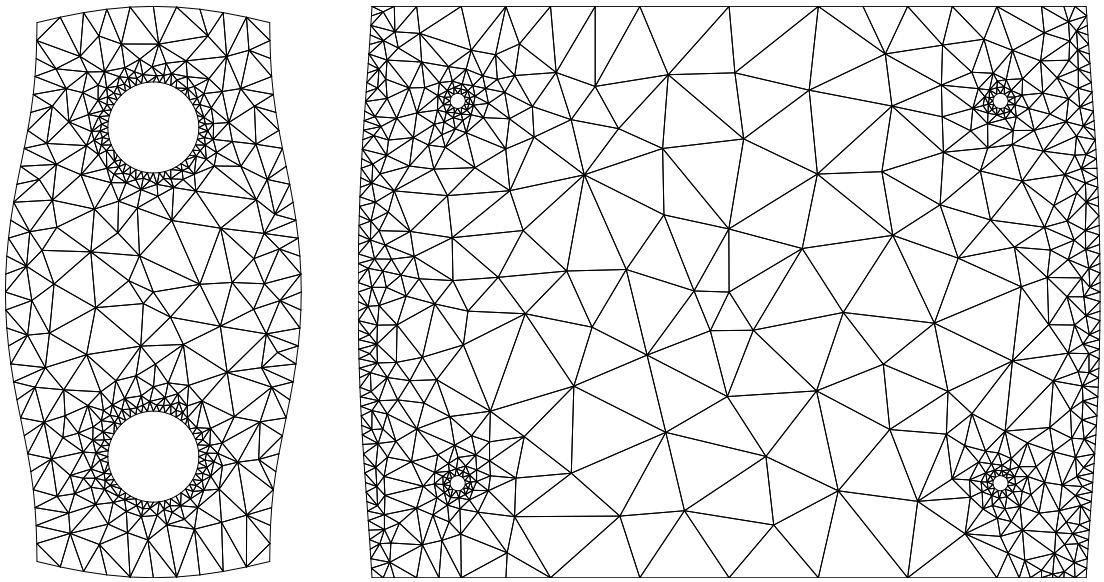
5.3.3.1. *numericalMethods.geometry.latticeReduction.* The LLL[23] lattice reduction algorithm [**LLL82**] is the technical core of the implementation of Siegel's reduction algorithm[24] , which itself is crucial for the computation of Riemann theta functions, see Chapter 3 and Section 5.6.2.

5.3.3.2. *numericalMethods.geometry.meshGeneration.* This package has currently one publicly available sub-package, called *ruppert*, that implements algorithms for two-dimensional quality mesh generation and construction of Delaunay triangulations, constrained Delaunay triangulations, and Voronoï diagrams. The class have been implemented by Heller [**Hel02**]. His code follows the ideas of *triangle*[25], a C program by Jonathan R Shewchuk. The program allows the triangulation of non-simply connected plain regions with polygonal boundary. The core is Ruppert's algorithm for two-dimensional quality mesh generation [**Rup95**], which "is perhaps the first theoretically guaranteed meshing algorithm to be truly satisfactory in practice". It produces meshes with no small angles, using relatively few triangles (though the density of triangles can be increased

[23] *numericalMethods.geometry.latticeReduction.LLL*

[24] *riemann.theta.SiegelReduction*

[25] see *http://www-2.cs.cmu.edu/~quake/tripaper/triangle0.html* for information about the C program *triangle*, which is freely available at *http://www.cs.cmu.edu/~quake/triangle.html* and from Netlib.



(a) S_+ for He $_1$ (b) $S_{-,2}$ for He $_2$

FIGURE 5.3.5. Triangulations of the fundamental domain of Schottky groups associated to Helicoids with handles.

under user control) and allowing the density of triangles to vary quickly over short distances. Figure 5.3.5 shows the result of the algorithms for two domains associated to the Schottky data S_+ and $S_{-,2}$ generating the surfaces $\mathcal{H}e_1$ and $\mathcal{H}e_2$ (Sections 4.4.1 and 4.4.3).

5.4. The mfc Project

The mfc (Mathematical Foundation Classes) project provides high level classes for basic mathematical purposes: complex numbers, polynomials, special groups, or lie-algebras . In addition, the package also contains GUI components, editors and panels implementing the associated *Java Beans* interfaces (Figure 5.4.1).

The APIs of all the classes are very similar and easy to use. Internally, all data is administered with publicly accessible primitive data typesdue to performance reasons. The four complex entries of a complex 2-by-2 matrix[26] $\begin{pmatrix} a & b \\ c & d \end{pmatrix}$ are represented by eight doubles: `aRe, aIm, bRe, bIm, cRe, cIm, dRe, dIm`. This is sometimes cumbersome, but creating five instead of a single instance can have a major impact as we will see later in this section. Anyhow, the APIs of these classes offer a convenient front-end, which do not force the user to deal with their internal structure.

[26] *mfc.matrix.Complex2By2*

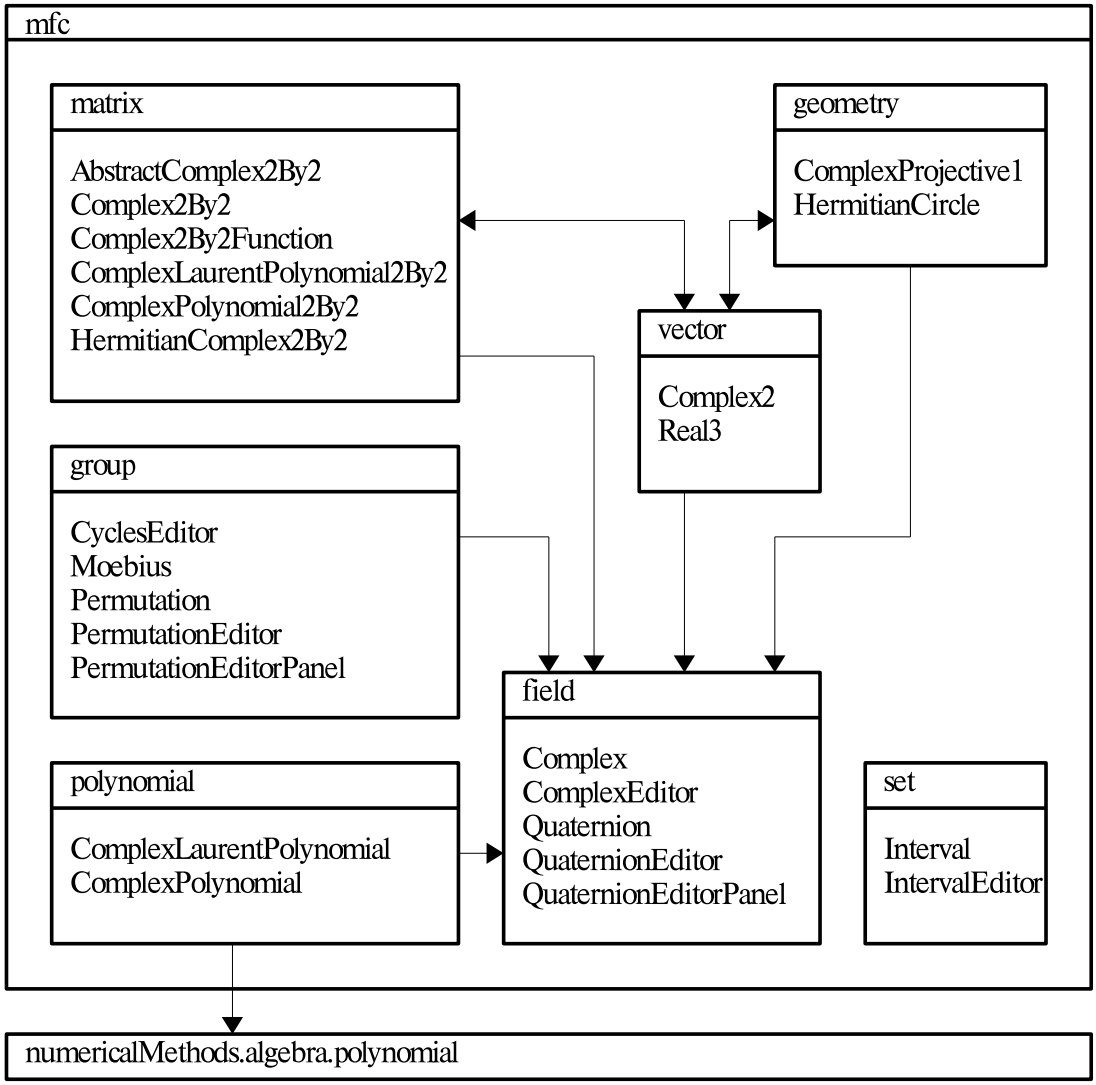


FIGURE 5.4.1. UML diagram of classes and packages in *mfc*.

An instance of a mathematical class of this project usually represents an element of an algebraic category: group, algebra, field, or vector space. Thus the nature of these classes is to represent a value, in Java normally implemented as an immutable type. Hence instances can not be changed; their state is totally defined at the time they are created. Many of the basic *Java Foundation Classes*[27] are of this kind: *String*, *Double*, or *Color*. Immutable types have many advantages, they allow convenient and safe implementations. Nevertheless the types of this package are mutable, thus representing a variable rather then a constant. The

[27]The *Java Foundation Classes* is an extensive collection of standard classes which are incorporated in the Java language.

reason for this is that in a computational setup performance is always critical and the creation of objects in large amounts has to be avoided.

Even our objects are mutable they still provide functions which allow for a programming style similar to an immutable implementation. To multiply to complex numbers `a` and `b`, for example, and to store the result in `c`, you can either call

(1) `c = a.times( b ),` or
(2) `c.assignTimes( a, b ).`

The first method returns the result, the product, in a newly created instance. Its reference is stored in `c`. Thus, this method could be also defined on an immutable type. The second function assigns to the instance `c` the product, which is certainly only possible with a mutable type. The immutable style is much easier to read especially if you have nested expressions. We want to give another example: If you need to compute $d = a \cdot (b + c)$ you can write

(1) `d = a.times( b.plus(c) )`, or
(2) `d.assignPlus( b, c ); d.assignTimes( a ) .`

Of course, (1) us much easier to read than (2). The only problem with the first expression is that it creates an unnecessary instance for the intermediate result $b + c$. This can have a dramatic negative impact on the performance, as the following shows: Let us compare two implementations for the exponential series

$$\exp z = \sum_{i=0}^{\infty} \frac{z^i}{i!}.$$

```
public Complex exp( Complex z ) {
     Complex sum = new Complex( 1 );
     Complex term = new Complex( z );
     for( int i=2; i<=N; i++ ) {
           term = term.times( z ).divide( i );
           sum = sum.plus( term );
     }
     return sum;
}
```

The implementation above creates $3N - 1$ instances compared to 2 of the implementation below, which is 6 times faster ($N > 50$)[28] then the first.

[28]measured on Linux system with a Java(TM) 2 Runtime Environment, Standard Edition (build 1.4.2-b28) with Java HotSpot(TM) Client VM (build 1.4.2-b28, mixed mode).

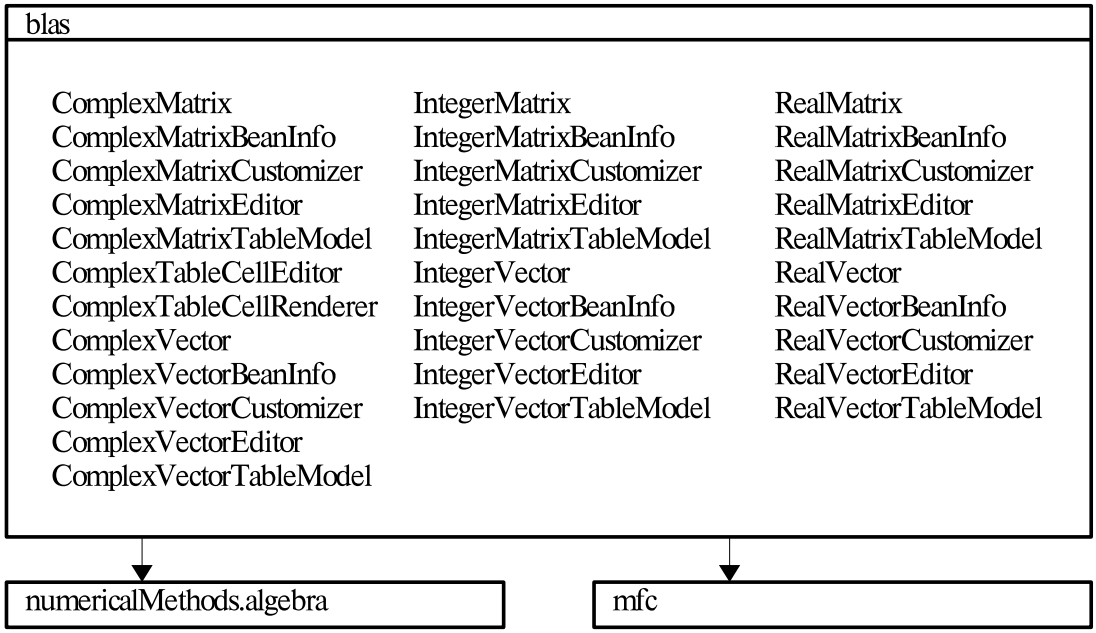


FIGURE 5.5.1. UML diagram of classes and packages in *blas*.

```
public Complex exp( Complex z ) {
     Complex sum = new Complex( 1 );
     Complex term = new Complex( z );
     for( int i=2; i<=N; i++ ) {
           term.assignTimes( z );
           term.assignDivide( i );
           sum.assignPlus( term );
     }
     return sum;
}
```

Suns 1.4. Java Hot Spot Compiler *inlines* all method calls in the loop, which makes the code very efficient and comparable highly performing than a plain C[29] implementation.

5.5. The blas Project

This package is a basic linear algebra system for integer, real, and complex matrices and vectors. It mainly wraps the functionality of the algebra section of the *numericalMethod*s project and provides a convenient front-end, including Bean conform editors and panels, following the same principles as the *mfc* project. Figure 5.4.1 shows an UML class and package diagram of the *blas* project.

[29]measured on Linux system with the GCC 3.3.2 C compiler.

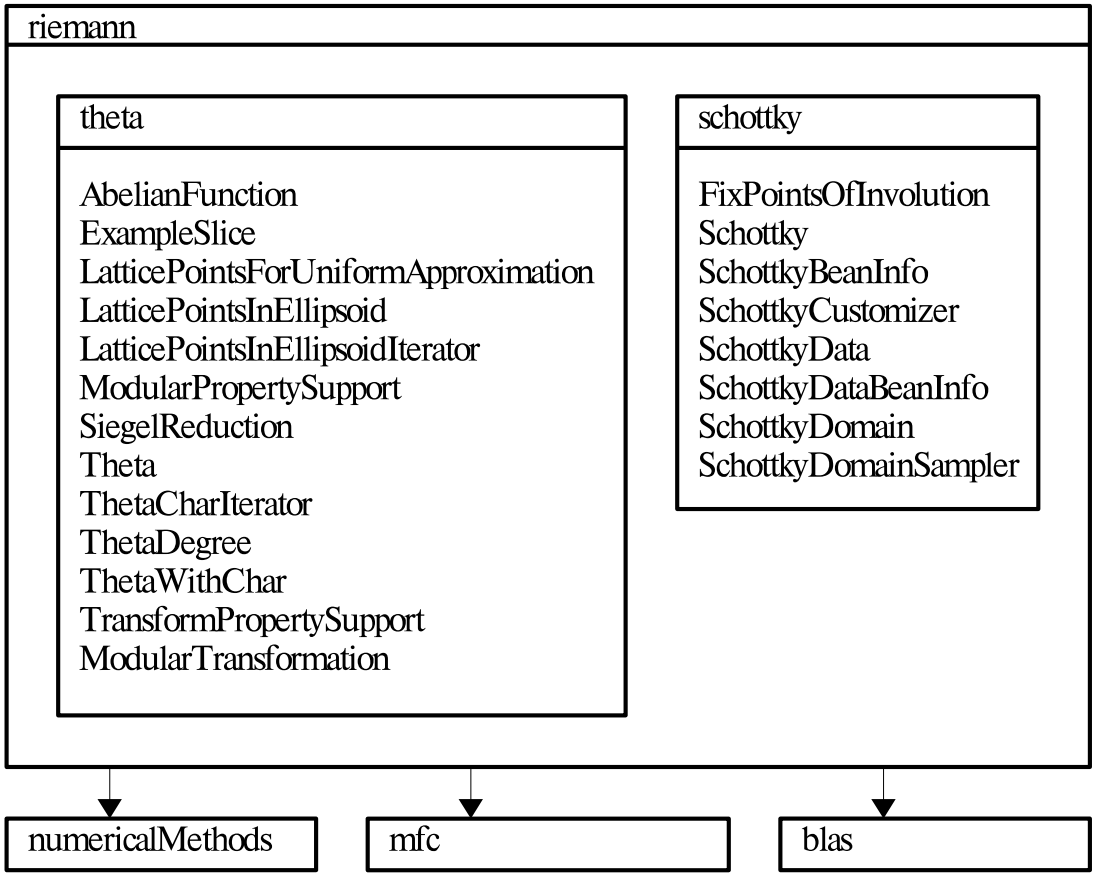


FIGURE 5.6.1. UML diagram of classes and packages in *riemann*.

One of its major advantages is the wide range of operations that mix the different types. For example by writing `a.times( b )` you multiply any instances of the types *IntegerVector, RealVector, ComplexVector, IntegerMatrix, RealMatrix, or ComplexMatrix* with any other instance of these types plus the scalar type *int, double* and *Complex*.

Vectors and matrices are internally represented as arrays of ordinary types (int and double). These arrays are one-dimensional for vectors and two-dimensional for matrices and can be accessed by and easily passed to external routines.

5.6. The riemann Project

The riemann project currently contains two public sub-packages (Figure 5.6.1). Both packages implement the current state of research in this field. A third package concerning algebraic curves and coverings is subject of a major refactoring and will be published as soon as possible.

5.6.1. riemann.schottky. This package implements the numerical methods and algorithms presented in Chapter 2 for the evaluation of certain automorphic functions and forms in the context of Schottky uniformization. A classical theorem states that for any Riemann surface $\mathcal{R}$ exists a Schottky group G such that $\mathcal{R}$ is conformally equivalent to the quotient Ω/G, where Ω denotes the set of discontinuity of G. A Schottky group is a free, finitely generated, discontinuous group G that is purely loxodromic, i.e., a Schottky group of rank N, which equals the genus of the associated Riemann surface, can always be generated by N loxodromic transformations $\sigma_1, \ldots, \sigma_N$. Further a loxodromic transformation σ_i can be defined by its fixed points A_i and B_i and the loxodromic factor μ_i, with $|\mu_i| < 1$.

Thus all Schottky groups of rank N can be associated to a list of $3N$ complex values

$$S = \{A_1, B_1, \mu_1, \ldots, A_N, B_N, \mu_N\} \,,$$

which is called the Schottky data. Not any $3N$ complex numbers define a Schottky group, but there exists a convenient sufficient criterion which is related to the notion of *iso-classical* Schottky groups. For any loxodromic transformation exists a unique pair of circles having the same radii, which are mapped by the transformation onto each other. These circles are called *isometric circles*. The differential of the transformation takes on the absolute value 1 one the circle. A Schottky group is called iso-classical if all the isometric circles of the generators are exterior to each other. This can be easily checked for a set of generators and it is easy to see that such a set generates a free, finitely generated, discontinuous group, which is therefore a Schottky group. The exterior of the isometric circles of a classical Schottky group define a fundamental domain for the quotient Ω/G, i.e., it maps conformally one-to-one to the associated Riemann surface. It is not clear that all Riemann surfaces can be uniformized by a iso-classical Schottky group. Thus, the fact that the numerics only deals with iso-classical Schottky groups means a limitation.

The strength of this approach is that functions and differentials of the Riemann surface Ω/G must be automorphic on Ω. For normalized[30] differentials of first kind exist closed representations as (-2)-dimensional Poincare theta series:

$$\omega_n(z) = \sum_{\sigma \in G_n \backslash G} \left(\frac{1}{\sigma(z) - B_n} - \frac{1}{\sigma(z) - A_n} \right) (\gamma_\sigma z + \delta_\sigma)^{-2} \, dz \,, \tag{5.6.1}$$

where G_n denotes the subgroup of G generated by the generator γ_n and $G_n \setminus G$ defines the cosets. Elementary computations deliver further formulas for the

[30]The a-cycles coincide with isometric circles.

integrals of first kind

$$\Omega_n(z) = \int_\infty^z \omega_n = \sum_{\sigma \in G/G_n} \log \frac{z - \sigma(B_n)}{z - \sigma(A_n)} \tag{5.6.2}$$

and the period matrix

$$B_{nm} = \delta_{nm} \log \mu_n + \sum_{\sigma \in G_m \backslash G/G_n, \sigma \neq id} \log \{B_m, A_m, \sigma(B_n), \sigma(A_n)\}, \tag{5.6.3}$$

where the *curly* brackets indicate the cross-ratio $\{a,b,c,d\} = \frac{a-c}{a-d}\frac{b-d}{b-c}$. The series above do not always converge and it is challenging to evaluate them in a stable manner.

The numerics in this package allow the evaluation of several other series:

(1) Normalized differentials and integrals of third kind having simple poles at A and B with residues -1 and 1:

$$\omega(z) = \sum_{\sigma \in G} \left(\frac{1}{\sigma(z) - B} - \frac{1}{\sigma(z) - A} \right) (\gamma_\sigma z + \delta_\sigma)^{-2} dz \tag{5.6.4}$$

and

$$\Omega(z) = \int_\infty^z \omega = \sum_{\sigma \in G} \log \frac{z - \sigma(B)}{z - \sigma(A)}. \tag{5.6.5}$$

(2) Series of the form

$$\Sigma_k(z,w) = \sum_{\sigma \in G} \sigma(z)^k - \sigma(w)^k \tag{5.6.6}$$

(3) The constants given by the series of the form

$$V_{n,k} = \sum_{\sigma \in G_n \backslash G} \sigma(A_n)^k - \sigma(B_n)^k \tag{5.6.7}$$

(4) The constant

$$\gamma = \sum_{\sigma \in G} \frac{1}{\gamma_\sigma^2} \tag{5.6.8}$$

The major drawback of this method is that the series above, which we will simply refer to as *Schottky series*, do not converge in the general case and that there is no general criteria in sight. However, for many special cases criteria exist.

In Chapter 2 we presented sufficient criteria that can be easily evaluated by a computer. The (integral) series (5.6.2), (5.6.3), (5.6.5), (5.6.6), and (5.6.7) converge if the limit q_∞^Ω of the monotonously decreasing series q_k^Ω is smaller then 1. The (differential) series (5.6.1), (5.6.4), and (5.6.8) have similar critirea. They converge if the limit of the monotonously decreasing series (q_k^ω) is smaller than 1. The computation of $q_l^\star$ involves to generate all group elements up to word length $l+1$. For groups of higher order one should therefore choose carefully how many

terms of the series $(q_l^\star)$ are used to check the criteria. The monotone series $(q_l^\star)$ converge very fast and in our experience, one should give up if the third term is still not smaller one. The series can only be evaluated stably, if it converges sufficiently fast, which only is the case if the limit is significant smaller then one.

The front-end for the evaluation of the Schottky series is provided by the class *Schottky* which extends the class *SchottkyData*. The main purpose of the class *SchotttkyData* is to provide a nice front end to configure Schottky data and to check whether the data generates a classical Schottky group. *SchottkyData* provides a huge variety of methods allowing the manipulation of a fixed number of generators. We recommend to create the Schottky data using this class and finally passing the data to an instance of the evaluation class *Schottky*. To create the Schottky data

$$S_{-,2} = \left\{A, -A, \mu, -\overline{A}, \overline{A}, \mu\right\}$$

with $A, \mu \in \mathbb{C}$ for the Helicoid $\mathcal{H}e_2$ (Section 4.4.3) you can either write

```
schottkyData = new SchottkyData( new Complex[]
    { A, A.neg(), mu,A.conjugate().neg(), A.conjugate(), mu } )
```

or configure the object sequentially and write

```
schottkyData = new SchottkyData( 2 );
schottkyData.setA(  0, A );
schottkyData.setB(  0, A.neg() );
schottkyData.setMu( 0, mu );
schottkyData.setA(  1, A.conjugate().neg() );
schottkyData.setB(  1, A.conjugate() );
schottkyData.setMu( 1, mu );
```

Before you pass the Schottky data to the evaluator class `Schottky`, you should check whether the data generates a classical schottky group and call

```
schottkyData.isClassical()
```

Otherwise you run the risk of getting a *RuntimeException*[31] when you pass the data to the evaluator, which you can do at its instanciatino:

```
schottky = new Schottky( schottkyData );
```

or afterwards with

```
schottky.setUniformizationData( schottkyData )
```

At this moment some precomputation steps are performed including the generation of all elements of the Schottky group up to word length 2. All other elements are computed and stored in a tree the first time they are needed for the evaluation of a term. Once elements are added to the tree, they are not removed unless the entire instance is destroyed. When the group is changed, i.e., the Schottky data changes, the instances for the group elements are merely updated (but only on demand). This is crucial for the performance, because the tree can get very big.

[31] *RuntimeException* is a basic *Exception* type, consult the documentation of the Java foundation classes for details.

We had examples where we dealt with several hundred thousand group elements. The default accuracy for the evaluation of the Schottky series is 10^{-7}, which is mutable bean property[32]of class *Schottky*.

Before you start to compute the Schottky series you should assure that the implemented algorithms are able to evaluate them. According to the discussion above we have different criteria for "differential" and "integral" series, which you can query by calling

schottky.isDifferentialSeriesEvaluable() [33]

and

schottky.isIntegralSeriesEvaluable() [34].

The methods checks for $q_l^\star < C$, with the default values $l = 2$ and $C = 0.75$. These default values are also mutable properties of the class *Schottky*, but they should only be altered with care (see discussion above). Now you are ready to evaluate the Schottky series.

To evaluate the period matrix you can simply write

ComplexMatrix B = schottky.getPeriodMatrix().

The Abelian differential ω_n are evaluated at the position z in the fundamental domain F by

Complex r = schottky.abelianDifferentialOf1stKind(z, n)

and the integral of first kind by

Complex r = schottky.abelianIntegralOf1stKind(z, n).

Because of performance reasons the code does not check whether the position z is valid. If the the argument z is in one of the isometric circles the algorithm can fail and the code will throw a *RuntimeException*, but first it will create the maximum number of possible group elements (default: 200,000). This is certainly to be avoided; if your are not sure that an argument is valid, you should test this with

schottky.isInFundamentalDomain(z)

in forehand.

All methods for evaluating Schottky series are overwritten with versions that allow the prescription of a specific accuracy, e.g. ,

ComplexMatrix B = schottky.getPeriodMatrix(0.001)

evaluates the period matrix with an accuracy of 0.001 instead of the default accuracy. A different type of overwritten methods enables to provide an instance for the result, which helps to relieve the garbage collector. The call

schottky.abelianDifferentialOf1stKind(r, z, n)

evaluates ω_n at z and returns the result in r, for example.

---

[32]Property is notion in the context of java programming and related to "Bean" conventions, see [**Eng97**].

[33]the Bean convention allows for boolean Properties (apart form the standard "get") only the singular "is" as prefix and no "are", which would lead to better readable method name in this case.

[34]There is also an abbreviation to test both at once: *isSeriesEvaluable().*

The subject of the next Section is the *riemann.theta* package. In Section 5.6.3 we combine *riemann.schottky* and *riemann.theta* and give example implementations for some real mathematical problems.

5.6.2. riemann.theta. This package implements the methods and algorithms presented in Chapter 3 for computing riemann theta functions including those with characteristics. An important ingredient for the computation of Riemann theta function are modular transformations and Siegel´s Reduction algorithm for which the package also offers public interfaces, see Figure 5.6.1 for a UML class diagram.

5.6.2.1. *Computing Riemann theta functions.* The Riemann theta function is a complex-valued function of g complex variables and defined by

$$\theta\left(z\,|B\right) = \sum_{n\in\mathbb{Z}^g} e^{\frac{1}{2}\langle n,B\cdot n\rangle+\langle z,n\rangle},$$

where $z \in \mathbb{C}^g$ and B is a symmetric g-dimensional matrix with strictly negative definite real part. Be aware that there are many different conventions of writing the theta functions and that you therefore might have to perform a scaling of the arguments to adapt it to yours.

The complex matrix B is usually the period matrix of a given Riemann surface and computing it is a hole different story. The Schottky uniformization as described in the previous Section offers one way of calculating it.

Suppose you have a valid period matrix B: To evaluate Riemann theta functions create an instance of the central class of this package:

```
theta = new Theta( B ).
```

The period matrix is the most important property of the class *Theta.* You can change it by calling

```
theta.setPeriodMatrix( B ).
```

This will trigger some pre-calculation steps. The implementation is optimized to evaluate the Riemann theta function many times for a fixed period matrix. Thus, setting the period matrix takes much longer then evaluating the function. Unfortunately, it is not possible to be more precise, because the expenditure depends on the period matrix (especially its genus) and the configuration of the evaluating instance.

```
Complex r = theta.theta( z )
```

evaluates the Riemann theta function at the argument $z \in \mathbb{C}^g$. Following the philosophy of the whole project, there exist also overloaded methods which allow to prescribe the result as a parameter:

```
theta.theta( z, r ).
```

These functions are not particularly useful because the Riemann theta function grows exponentially and in practice you might easily leave the range of the double precession representation. It is therefore necessary to separate the exponential

growth from the oscillating part of the function:

$$\theta\left(z\,|B\right) = e^{f(z|B)} \cdot \theta_{\Sigma}\left(z\,|B\right) ,$$

with $f\left(z\,|B\right)$ being a quadratic function in z, see Chapter 3.

theta.theta(z, f, o)
evaluates the Riemann theta function in the form from above with $f = f\left(z\,|B\right)$ and $o = \theta_{\Sigma}\left(z\,|B\right)$. The accuracy of the evaluation always refers to the oscillating part *o* only and is by default 10^{-7}. In almost all applications theta functions are used to compute Abelian functions, which can be expressed as ratios of theta functions. Thus, the exponential factors usually almost cancel. Typical examples are of the form:

$$f\left(z\right) = \frac{\theta\left(z+a\,|B\right)}{\theta\left(z+b\,|B\right)} = e^{f(z+a|B)-f(z+b|B)} \cdot \frac{\theta_{\Sigma}\left(z+a\,|B\right)}{\theta_{\Sigma}\left(z+b\,|B\right)} .$$

Since the vectors $a, b \in \mathbb{C}^g$ are relatively small the exponential factor is about 1. An implementation of this function could look like:

```
package helicoid.example;
import mfc.field.Complex;
import blas.ComplexMatrix;
import blas.ComplexVector;
import riemann.theta.Theta;
public class SimpleAbelianFunction {
     ComplexMatrix B;
     ComplexVector a, b;
     Theta theta;
     public SimpleAbelieanFunction( ComplexMatrix B,
                                    ComplexVector a,
                                    ComplexVector b ) {
           this.B = new ComplexMatrix(B);
           this.a = new ComplexVector(a);
           tihs.b = new ComplexVector(b);
           theta=new Theta(B);
     }
     public Complex valueAt( ComplexVector z ) {
           theta.theta( z.plus(a), f_a, o_a );
           theta.theta( z.plus(b), f_b, o_b );
           return Complex.exp( f_a.miuns( f_b ) )
                              .times( o_a ).divide( o_b );
     }
}
```

The implementation is not optimal if the function needs to be evaluated many times, because it creates six unnecessary instances for intermediate results. This can be avoided by using the mutable programming style presented in Section 5.6.

The API also offers functions for the first and second derivative of the theta function with respect to z.

Complex r = dTheta.theta(z, X)
computes the partial derivative of theta in direction of X. Again this method is overloaded by a version that separates the exponential growth of the derivative from the oscillating part:

$$D_X\theta\,(z\,|B\,) = e^{f(z|B)} \cdot (D_X\theta)_\Sigma\,(z\,|B\,)\ ,$$

with f being the same quadratic function as for the Riemann theta function itself.
dTheta.theta(z, X, f, o, dxo)
evaluates the function and the derivative simultaneously with $f = f\,(z\,|B\,)$, $o = \theta_\Sigma\,(z\,|B\,)$, and $dxo = (D_X\theta)_\Sigma\,(z\,|B\,)$. Evaluating the function causes almost no extra cost and in many applications, e.g., logarithmic derivatives, you need the function value in addition to its derivative. Evaluating the second derivative works similar.

The error estimates for the derivative of the Riemann theta function are much more complicated than for the function itself. Therefore, we implemented the derivatives of the approximation instead of approximating the derivative. In almost 10 years experience in computing theta functions we never encountered any practical problems caused by inaccuracies of the derivatives, but in theory badly conditioned scenarios are likely to occur for large arguments z.

There are three important boolean properties that configure the evaluation algorithm of the Riemann theta function:

(1) `fillFactorErrorUsed`
(2) `uniformApproximationUsed`
(3) `siegelReductionPerformed`

By default all properties are set to `true`, but in certain situations different configurations may lead to better results.

The fill factor error (FFE) is a heuristic value to sharpen the error estimates for theta functions, which can drastically reduce the number of terms needed to approximate the oscillating part, see [**DHB**$^+$**04**] for details. The drawback of using the FFE is that the prescribed accuracy is not guaranteed anymore. Tests have shown that in practice the FFE is usually good enough. When accuracy is your concern rather than performance you might want to set this property to `false`.

The second property allow for the choice between uniform and pointwise approximation. For genus smaller than five uniform approximation is usually faster than pointwise. For higher genus pointwise approximation seems better, but that always depends on the specific case. The uniform approximation consumes some pre-calculation time, thus, if you only need to evaluate the theta function a few times for a fixed period matrix the pointwise approximation can also perform better for small genus. If performance is an important issue in your application you should try both possibilities; consult Section 3.3.3 for a more complete discussion.

The last propertiy controls whether Siegel's reduction algorithm is performed when the period matrix is set. Switching it off is only sensible in situations where you already know that your period matrix is not reducible by Siegel's algorithm. But this will only have a significant impact if you have no more than a few function evaluations for a fixed period matrix.

5.6.2.2. *Siegel's Reduction algorithm and Modular Transformations.* Siegel's reduction algorithm and modular transformation of Riemann matrices are closely related to Riemann theta functions, which just as the elliptic functions and the Jacobian theta functions have a modular transformation property. The modular transformation is a transformation on the Riemann (period) matrix B:

$$B' = \frac{1}{2\pi i} \left(a\, B + 2\pi i\, b\right) \left(c\, B = 2\pi i\, d\right)^{-1} .$$

B and B' define the "same" theta function up to an affine transformation of the argument z and an overall scaling factor. The computation of these transformations and factors is performed by the class *ModularTransformationSupport.*

Siegel's reduction algorithm finds an element of the modular group $\mathrm{Sp}\,(2g, \mathbb{Z})$, a modular transformation, which transforms a given period matrix into the fundamental domain of Riemann matrices. The classes *SiegelReduction, ModularTransformation*, and *ModularTransformationSupport* are members of the package *riemann.theta.* We illustrate the usage of these classes by means of an example from section 4.4.2.

For the asymmetric Example 0 (Table 5) we have

$$\begin{aligned} \mu &= -0.0118611871544 - 0.0033906555595\, i \qquad \text{and} \\ B = \log \mu &= -.39520911583988 - 2.8631569862600\, i\,. \end{aligned}$$

$|B| < 2\pi$ and therefore it is not in the fundamental domain of the modular group[35]. But to check whether B is a rhombic torus we need a representative of the fundamental domain. Siegel's reduction algorithm determines the modular transformation

$$\sigma = \begin{pmatrix} 1 & 1 \\ -1 & 0 \end{pmatrix}$$

that yields

$$B' = -6.3061076416625 - 2.1752178192847\, i\,,$$

which has an absolute value bigger then 2π and an absolute value of its imaginary part smaller then π. Thus B' is in the fundamental domain and not rhombic.

[35]Integrals of first kind are normalized along a-cycles to $2\pi i$.

```
package helicoid.example;
import mfc.field.Complex;
import blas.ComplexMatrix;
import riemann.theta.ModularPropertySupport;
import riemann.theta.ModularTransformation;
import riemann.theta.SiegelReduction;
public class SiegelReductionAndTransformation {
     public void main( String [] arg ) {
           // mu of the asymmetric example #0
           Complex mu = new
           Complex(-0.0118611871544, -0.0033906555950 );
           // creating a "period matrix", e.g.  setting
           the modulus
           ComplexMatrix B = new ComplexMatrix(1);
           B.set( 0,0, mu.log() );
           // create an instance of the Siegel-reduction
           algorithm for B SiegelReduction
           siegelReduction = new SiegelReduction(B);
           // query the reduced period matrix herefor
           ComplexMatrix rB1 =
           siegelReduction.getReducedPeriodMatrix();
           // query the modular transformation realizing
           the reduction
           ModularTransformation mt =
                siegelReduction.getModularTransformation();
           // performing a modular transformation of a
           period matrix rB1 and rB2 are equal
           ComplexMatrix rB2 =
                ModularPropertySupport.transformPeriodMatrix( B, mt );
     }
}
```

5.6.3. Examples. We like to give two mathematical challenging example implementations which combine the capabilities of the *riemann.schottky* and *riemann.theta* packages.

5.6.3.1. *Calculating the products a^2,b^2, and ab of the helicoidal spinors.* The code below computes the products a^2,b^2, and ab of the helicoidal spinors following the ideas of Section 4.3.1. For the sake of simplicity we used the inefficient immutable programming style (Section 5.6). This causes the creation of many instances storing intermediate results. This should be avoided using the mutable style, because the spinors are evaluated very often in the process of surface integration.

```
package helicoid.example;
import mfc.field.Complex;
import riemann.schottky.*;
import riemann.theta.*;
import blas.*;
public class ProductsOfHelicoidalSpinors {
     Schottky schottky;
     ThetaWithChar eta, delta;
     ThetaCharIterator deltaIterator;
     ComplexVector V, halfV, zero;
     Complex etaOfHalfV;
     public ProductsOfHelicoidalSpinorsCalculator(
            double [] schottkyData, ComplexVector [] bCycles,
            IntegerVector alpha, IntegerVector beta, double eps ) {
            schottky = new Schottky( schottkyData, eps );
            ComplexMatrix periodMatrix = schottky.getPeriodMatrix( bCycles );
            V     = schottky.getV();
            halfV = V.divide(2);
            Theta theta = new Theta( periodMatrix, eps);
            delta = new ThetaWithChar(theta);
            eta   = new ThetaWithChar(theta);
            eta.setAlpha(alpha);
            eta.setBeta( beta);
            deltaIterator = new ThetaCharIterator(delta);
            zero = new ComplexVector( eta.dim() );
            etaOfHalfV = eta.theta( halfV );
     }
     public void calculatorProductsOfSpinors(
            Complex z, Complex aa, Complex bb, Complex ab ) {
            ComplexVector w = schottky.abelianDifferentialOf1stKind(z);
            ComplexVector W = schottky.abelianIntegralOf1stKind(z);
            Complex omega   = schottky.sigma( z, Complex.ZERO);
            deltaIterator.startOddCharIteration();
            Complex thetaDelta = delta.theta(W);
            while ( thetaDelta.absSqr() < 1e-3) {
                 if (!deltaIterator.iterateOddChar())
                      throw new RuntimeException( "could not avoid zero");
                 thetaDelta = delta.theta(W);
            }
            Complex scalar = delta.dTheta( zero, V )
                 .divide( eta.theta( V.divide(2) ).sqr() ).neg();
            Complex sqrOfC = delta.dTheta(zero,V);
            Complex sqrOfH = delta.dTheta(zero,w);
            Complex commonFactor = sqrOfC.times(sqrOfH).times( omega.exp())
                 .divide( delta.theta( W ).times( etaOfHalfV ).sqr());
            Complex etaOfWPlusHalfV  = eta.theta( W.plus(  halfV ));
            Complex etaOfWMinusHalfV = eta.theta( W.minus( halfV ));
            aa = commonFactor.times( etaOfWMinusHalfV.sqr());
            bb = commonFactor.times( etaOfWPlusHalfV.sqr());
            ab = commonFactor.times( etaOfWMinusHalfV.times(etaOfWPlusHalfV))
                 .times( eta.parityOfSpin() % 2 == 1 ? -1:1 );
     }
}
```

5.6.4. KP2 Equation - Shallow Water Waves. The equations

$$4u_{xt} + 3\left(u^2\right)_{xx} \pm u_{xxxx} - 3u_{yy} = 0$$

discovered by Kadomtsev & Petviashvili (KP) are a generalization of the Korteweg & deVries (KdV)[36] equation. For positive sign of the u_{xxxx} term it is called KP2 equation. Solutions of the KP2 equation describe the evolution of gravity-induced waves of moderate amplitude on shallow water of uniform depth when the waves are nearly one-dimensional.

For Schottky data

$$S = \{A_1, B_1, \mu_1, \ldots, A_N, B_N, \mu_N\}$$

with $B_i = \bar{A}_i$ and $\mu_i \in \mathbb{R}$ yield all real non-singular finite-gap solutions of the KP2 equation and can be described by Krichever's formula [**Kri78**]

$$u\,(x, y, t) = 2\frac{\partial^2}{\partial x^2} \log\theta\,(U\,x + V\,y + W\,t + D\,|B\,) + 2c\,,$$

see also [**Dub81**]. The parameters in this formula can be given by Poincare series [**BBE$^+$94**]

$$\begin{array}{rcccl} U & = & V_{n,1} & = & \sum_{\sigma\in G_n\backslash G} \sigma\,(A_n) - \sigma\,(B_n)\ , \\ V & = & V_{n,2} & = & \sum_{\sigma\in G_n\backslash G} \sigma\,(A_n)^2 - \sigma\,(B_n)^2\ , \\ W & = & V_{n,3} & = & \sum_{\sigma\in G_n\backslash G} \sigma\,(A_n)^3 - \sigma\,(B_n)^3\ , \end{array}$$

$$c \;=\; \gamma \;=\; \sum_{\sigma\in G} \frac{1}{\gamma_\sigma^2}$$

and D is an arbitrary imaginary vector.

The code below shows an implementation of this solution using the efficient mutable programming style presented in Section 5.6.3.

[36] $u_t + \left(3u^2\right)_x + u_{xxx} = 0$

```
package helicoid.example;
import mfc.field.Complex;
import riemann.schottky.*;
import riemann.theta.*;
import blas.*;
public class KP2 {
     Schottky schottky;
     Theta theta;
     ComplexVector U, V, W, Z, T;
     Complex c;
     public KP2( Complex [] A, double [] mu, double eps ) {
            SchottkyData data = new SchottkyData(
            A.length );
            for( int i=0; i<A.length; i++ ) {
                 data.setA( i, A[i] );
                 data.setB( i, A[i].conjugate() );
                 data.setMu( i, mu[i], 0 );
            }
            if( !schottkyData.isClassical() )
                 throw new IllegalArgumentException
                            ( "schottky data is not classical");
            schottky = new Schottky( schottkyData, eps );
            if( !schottky.isSeriesEvaluable() )
                throw new IllegalArgumentException
                            ( "can not evaluate series");
            U = schottky.getV();
            V = schottky.getV(2);
            W = schottky.getV(3);
            c = schottky.gamma();
            Z = new ComplexVector( U.size() );
            T = new ComplexVector( U.size() );
            Theta theta = new Theta( schottky.getPeriodMatrix(), eps );
     }
     public Complex valueAt( double x, double y, double t ) {
            Z.assignTimes( U, x );
            T.assignTimes( V, y ); Z.assignPlus( T );
            T.assignTimes( W, t ); Z.assignPlus( T );
            Complex result = theta.ddLogTheta( Z, U, U );
            result.assignPlus(c);
            result,assignTimes(2);
            return result;
     }
}
```

Figure 5.6.2 shows a snap-shot of a webstart application visualizing shallow water waves computed as solutions of the KP2 equation using Schottky uniformization as described above. The application was generated using *Java Oorange*. This and other applications using the jtem library can be downloaded from *http://www.math.tu-berlin.de/geometrie/lab/* .

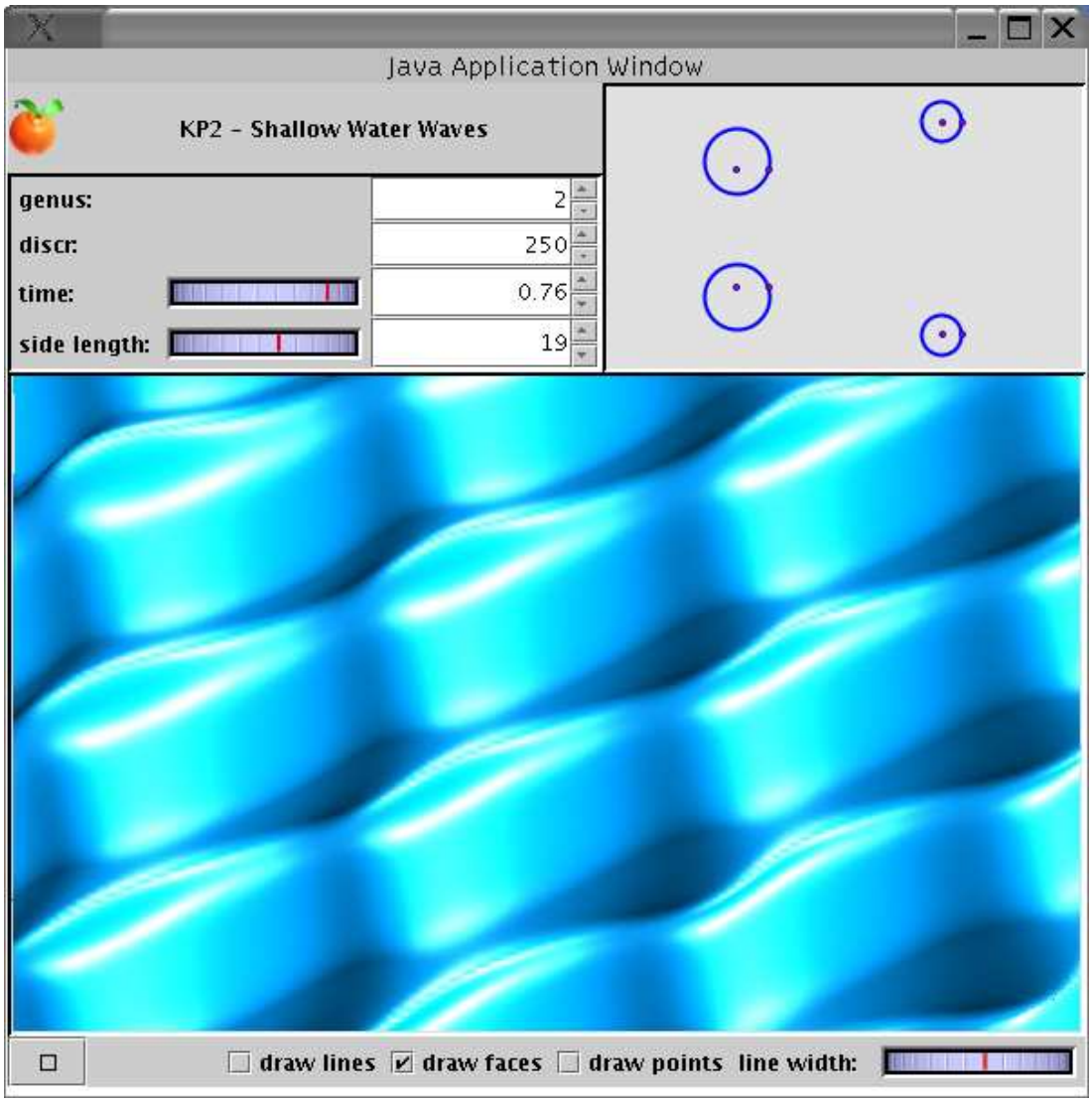


FIGURE 5.6.2. Snap-shot of a webstart application visualizing shallow water waves computed as solutions of the KP2 equation using Schottky uniformization. The application can be downloaded from *http://www.math.tu-berlin.de/geometrie/lab/ds.shtml* .

Bibliography

[Bak97] H. F. Baker. *Abel's Theroem and the Allied Therory Including the Theory of Theta Functions*. Cambridge University Press, 1897.

[BBE+94] E. D. Belokolos, A. I. Bobenko, V. Z. Enol'skii, A. R. Its, and V. B. Matveev. *Algebro-Geometric Approach to Nonlinear Integrable Equations*. Nonlinear Dynamics. Springer Verlag, 1994.

[Bob91] A.I. Bobenko. Constant mean curvature surfaces and integrable equations. *Russ. Math. Surv.*, 46(4):1–45, 1991.

[Bob94] A. I. Bobenko. Surfaces in terms of 2 by 2 matrices. Old and new integrable cases. In *Harmonic maps and integrable systems*, Aspects Math., E23, pages 83–127. Vieweg, Braunschweig, 1994.

[Bob98] A. I. Bobenko. Helicoids with handles and Baker-Akhiezer spinors. *Math. Z.*, 229(1):9–29, 1998.

[Bre02] R. P. Brent. *Algorithms for minimization without derivatives*. Dover Publications Inc., Mineola, NY, 2002. Reprint of the 1973 original [Prentice-Hall, Inc., Englewood Cliffs, NJ.

[Bur92] W. Burnside. *Proc, London Math. Soc.*, 1892.

[Col97] P. Collin. Topologie et courbure des surfaces minimales proprement plongées de $\mathbf{R}^3$. *Ann. of Math. (2)*, 145(1):1–31, 1997.

[Cos84] C. J. Costa. Example of a complete minimal immersion in $\mathbf{R}^3$ of genus one and three embedded ends. *Bol. Soc. Brasil. Mat.*, 15(1-2):47–54, 1984.

[DHB+04] B. Deconinck, M. Heil, A. I. Bobenko, M. van Hoeij, and M. Schmies. Computing Riemann theta functions. *Math. Comp.*, 73(247):1417–1442 (electronic), 2004.

[Dub81] B.A. Dubrovin. Theta functions and non-linear equations. *Russ. Math. Surv.*, 36(2):11–92, 1981.

[DvH01] B. Deconinck and M. van Hoeij. Computing Riemann matrices of algebraic curves. *Phys. D*, 152/153:28–46, 2001. Advances in nonlinear mathematics and science.

[Eng97] R. Englander. *Developing Java Beans*. O'Reilly, 1997.

[FK65] R. Fricke and F. Klein. *Vorlesungen über die Theorie der automorphen Funktionen. Band 1: Die gruppentheoretischen Grundlagen. Band II: Die funktionentheoretischen Ausführungen und die Andwendungen*, volume 4 of *Bibliotheca Mathematica Teubneriana, Bände 3*. Johnson Reprint Corp., New York, 1965.

[For29] L. R. Ford. *Automorphic Functions*. McGraw-Hill, New York, 1929.

[FS97] M. Flower and K. Scott. *UML Distilled: Applying the Standard Object Modeling Language*. Addison Wesley Longman, Reading, MA, 1997.

[GOP+97] C. Gunn, A. Ortamann, U. Pinkall, K. Polthier, and U. Schwarz. Oorange: A virtual laboratory for experimental mathematics. *Sfb 288 Preprint No. 260*, 1997.

[Hei95] M. Heil. *Numerical Tools for the study of finite gap solutions of integrable systems*. PhD thesis, Technische Universität Berlin, 1995.

[Hel02] Ullrich Heller. *Construction, Transformation, and Visualization of Willmore Surfaces*. PhD thesis, University of Massachusetts, Amherst, 2002.

[HM85] D. A. Hoffman and W. Meeks, III. A complete embedded minimal surface in $\mathbf{R}^3$ with genus one and three ends. *J. Differential Geom.*, 21(1):109–127, 1985.

[HM03] D. Hoffman and J. McCuan. Embedded minimal ends asymptotic to the helicoid. *Comm. Anal. Geom.*, 11(4):721–735, 2003.

[HNW93] E. Hairer, S. P. Nørsett, and G. Wanner. *Solving ordinary differential equations. I*, volume 8 of *Springer Series in Computational Mathematics*. Springer-Verlag, Berlin, second edition, 1993. Nonstiff problems.

[HPR01] L. Hauswirth, J. Pérez, and P. Romon. Embedded minimal ends of finite type. *Trans. Amer. Math. Soc.*, 353(4):1335–1370 (electronic), 2001.

[HW02] D. Hoffman and F. Wei. Deforming the singly periodic genus-one helicoid. *Experiment. Math.*, 11(2):207–218, 2002.

[HWK93] D. Hoffman, F. Wei, and H. Karcher. Adding handles to the helicoid. *Bull. Amer. Math. Soc. (N.S.)*, 29(1):77–84, 1993.

[Igu72] J. Igusa. *Theta functions*. Springer-Verlag, New York, 1972. Die Grundlehren der mathematischen Wissenschaften, Band 194.

[Kri78] I.M. Krichever. Algebraic curves and non-linear difference equations. *Russ. Math. Surv.*, 33(4):255–256, 1978.

[KS93] R. Kustner and N. Schmitt. The spinor representation of minimal surfacs in space. *GANG Preprint*, Preprint III.27 (1993).

[LLL82] A. K. Lenstra, H. W. Lenstra, Jr., and L. Lovász. Factoring polynomials with rational coefficients. *Math. Ann.*, 261(4):515–534, 1982.

[Mas67] B. Maskit. A characteri that are practicable in many cases as well as efficient evaluation algorithms for those series. zation of Schottky groups. *J. Analyse Math.*, 19:227–230, 1967.

[MR01] W. H. Meeks, III and H. Rosenberg. The uniqueness of the helicoid and the asymptotic geometry of properly embedded minimal surfaces with finite topology". *Preprint*, 2001.

[Mum83] D. Mumford. *Tata lectures on theta. I*, volume 28 of *Progress in Mathematics*. Birkhäuser Boston Inc., Boston, MA, 1983. With the assistance of C. Musili, M. Nori, E. Previato and M. Stillman.

[Mum84] D. Mumford. *Tata lectures on theta. II*, volume 43 of *Progress in Mathematics*. Birkhäuser Boston Inc., Boston, MA, 1984. Jacobian theta functions and differential equations, With the collaboration of C. Musili, M. Nori, E. Previato, M. Stillman and H. Umemura.

[Mum91] D. Mumford. *Tata lectures on theta. III*, volume 97 of *Progress in Mathematics*. Birkhäuser Boston Inc., Boston, MA, 1991. With the collaboration of Madhav Nori and Peter Norman.

[NM65] J. A. Nelder and R. Mead. A simplex method for function minimization. *The Computer Journal*, 7(4):308–313, 1965.

[Oss64] R. Osserman. Global properties of minimal surfaces in E^3 and E^n. *Ann. of Math. (2)*, 80:340–364, 1964.

[Oss86] R. Osserman. *A survey of minimal surfaces*. Dover Publications Inc., New York, second edition, 1986.

[PTVF92] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical recipes in C*. Cambridge University Press, Cambridge, second edition, 1992. The art of scientific computing.

[Rid] C. J. F. Ridders. *Advances in Engineering Software*, 4(2):75–76.

[Rup95] J Ruppert. A Delaunay refinement algorithm for quality 2-dimensional mesh generation. *J. Algorithms*, 18(3):548–585, 1995. Fourth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA) (Austin, TX, 1993).

[SB90] J. Stoer and R. Bulirsch. *Numerische Mathematik. 2*. Springer-Lehrbuch. [Springer Textbook]. Springer-Verlag, Berlin, third edition, 1990. Eine Einführung—unter Berücksichtigung von Vorlesungen von F. L. Bauer. [An introduction, with reference to lectures by F. L. Bauer].

[Sie89] C. L. Siegel. *Topics in complex function theory. Vol. III.* Wiley Classics Library. John Wiley & Sons Inc., New York, 1989. Abelian functions and modular functions of several variables, Translated from the German by E. Gottschling and M. Tretkoff, With a preface by Wilhelm Magnus, Reprint of the 1973 original, A Wiley-Interscience Publication.

[Val90] B. Vallée. A central problem in the algorithmic geometry of numbers: lattice reduction. *CWI Quarterly*, 3(2):95–120, 1990.

[Web] M. Weber. On the embeddedness of the genus one helicoid. Habilitationsschrift, University of Bonn, 2000.

[WHW03] M. Weber, D. Hoffman, and M. Wolf. An embedded genus-one helicoid. *Preprint*, 2003.

[WW02] M. Weber and M. Wolf. Teichmüller theory and handle addition for minimal surfaces. *Ann. of Math. (2)*, 156(3):713–795, 2002.