CARL VON OSSIETZKY universität OLDENBURG

Efficient integration on Riemann surfaces & applications

Von der Fakultät für Mathematik und Naturwissenschaften

der Carl von Ossietzky Universität Oldenburg

zur Erlangung des Grades und Titels eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

angenommene Dissertation

von

Christian Neurohr

geboren am 28.03.1989

in Bad Kreuznach

Gutachter

Dr. Steffen Müller

Weitere Gutachter

Prof. Dr. Nils Bruin

Prof. Dr. Florian Heß

Tag der Einreichung

09.03.2018

Tag der Disputation

04.05.2018

Kurzfassung

Die vorliegende Dissertation beschäftigt sich mit effizienten Algorithmen zur numerischen Integration von Differentialformen auf Riemannschen Flächen und deren Anwendung auf Problemstellungen in der arithmetischen Geometrie.

Explizite Berechnungen auf der Jacobischen Varietät einer algebraischen Kurve sind in der Zahlentheorie & algebraischen Geometrie von Interesse. In dieser Arbeit arbeiten wir ausschließlich mit Jacobischen über den komplexen Zahlen. Die (analytische) Jacobische einer algebraischen Kurve hat die Struktur eines komplexen Torus, welcher bis auf Isomorphie durch eine Periodenmatrix der zugehörigen Riemannschen Fläche gegeben ist. Die Einträge von Periodenmatrizen sind Integrale von holomorphen Differentialformen, sogenannte Perioden, entlang einer Homologiebasis. Diese enthalten wichtige Informationen über algebraische Kurven und ihre Jacobischen. Eine konkrete Verbindung zwischen dem komplexen Torus und der Kurve wird durch die Abel-Jacobi Abbildung hergestellt; diese bildet Punkte auf der Kurve auf Perioden entlang von Wegen, die diese Punkte verbinden, ab. Im Allgemeinen sind die Werte dieser Integrale transzendente komplexe Zahlen welche nicht symbolisch berechnet werden können.

Für Anwendungen in der arithmetischen Geometrie ist es oft notwendig Perioden zu hoher numerischer Präzision zu berechnen und wünschenswert, dass die Berechnungungen beweisbar korrekt sind. In dieser Arbeit approximieren wir Perioden von kompakten Riemannschen Flächen mittels numerischer Integration unter Nutzung beweisbarer Fehlerabschätzungen.

Für den Fall allgemeiner kompakter Riemannscher Flächen, die durch ebene (möglicherweise singuläre) affine Gleichungen, über Zahlkörpern definiert, gegeben sind, haben wir Algorithmen im Computeralgebrasystem MAGMA implementiert, die Periodenmatrizen und die Abel-Jacobi Abbildung schnell und zuverlässig berechnen; hierbei können problemlos mehrere hundert Dezimalstellen an Genauigkeit erreicht werden. Dafür benutzen und kombinieren wir verschiedene Integrationsverfahren: Gauss-Legendre Quadratur, Clenshaw-Curtis Quadratur sowie doppel-exponentielle Integration.

Für den Spezialfall der superelliptischen Kurven konnten wir sogar noch bessere Ergebnisse erzielen. In Zusammenarbeit mit Pascal Molin haben wir einen komplett beweisbaren Algorithmus entwickelt der unseren Anforderungen genügt und viel schneller ist als der Algorithmus für den allgemeinen Fall. Wir nutzen die Vorteile der Geometrie superelliptischer Kurven, um Perioden bis auf mehrere tausend Dezimalstellen genau zu berechnen. Hierfür verwenden wir doppel-exponentielle Integration sowie Gauss-Jacobi Integration. Die entsprechenden Algorithmen sind nicht nur in MAGMA, sondern auch in der C-Bibliothek ARB implementiert, mit deren Ballarithmetik die Genauigkeit unserer Ergebnisse zertifiziert werden kann.

In beiden Fällen, dem allgemeinen und dem superelliptischen, erweitern die Implementierungen unserer Algorithmen die Reichweite der Kurven und der Genauigkeit für welche Perioden in angemessener Zeit und numerisch stabil berechnet werden können.

Zuletzt diskutieren wir Anwendungen unserer Algorithmen in der arithmetischen Geometrie. Unter anderem, den Bezug zur berühmten Vermutung von Birch und Swinnerton-Dyer, das Problem Isogenien zwischen Jacobischen zu finden, die Berechnung der Riemannschen Thetafunktion und die Berechnung kanonischer Höhen auf abelschen Varietäten.

Unsere Implementierungen wurden bereits benutzt um hunderttausende Endomorphismenringe für die 'L-functions and modular forms database' zu berechnen (dies geschah in Zusammenarbeit mit Jeroen Sijsling) und um neue konkrete Daten als Nachweis für Beilinson's Vermutung für K_2 von Kurven zu berechnen.

Abstract

In this thesis we are concerned with *efficient* algorithms for numerical *integration* of differential forms *on Riemann surfaces* with *applications* motivated by arithmetic geometry.

Explicit computations on the Jacobian variety of an algebraic curve are of great importance in number theory & algebraic geometry. The present work is exclusively concerned with computing on Jacobians over the complex numbers. In particular, the (analytic) Jacobian has the structure of a complex torus which is, up to isomorphism, determined by a period matrix of the compact Riemann surface associated to the algebraic curve. These period matrices are defined in terms of integrals of holomorphic differentials, so-called periods, along a homology basis, and they encode valuable information about algebraic curves and their Jacobians. An explicit connection between the complex torus and the curve is established via the Abel-Jacobi map, which maps points on the curve to periods along paths connecting them. However, the values of these integrals are, in general, transcendental complex numbers which cannot be obtained symbolically.

For applications in arithmetic geometry it is often necessary to approximate periods to high numerical precision and it is desirable to have provable results. In this work we approach the problem of computing periods of compact Riemann surfaces using numerical integration together with rigorous error bounds.

For the case of general compact Riemann surfaces given by plane (possibly singular) affine equations defined over numbers fields, we implemented algorithms in the computer algebra system MAGMA that compute period matrices and the Abel-Jacobi map fast and reliably, for hundreds of decimal digits of precision. For that purpose we employ various numerical integration schemes, namely Gauss-Legendre quadrature, Clenshaw-Curtis quadrature and double-exponential integration.

We obtained even better results in the special case of superelliptic curves. In joint work with Pascal Molin we developed a completely rigorous algorithm that achieves this goal and is even faster than the one for the general case. We take advantage of numerous simplifications due to the geometry of superelliptic curves to obtain periods with thousands of digits of precision, using double-exponential integration and Gauss-Jacobi quadrature for numerical integration. The corresponding algorithms are not only implemented in MAGMA, but also in the C-library ARB whose ball arithmetic enables us to certify the accuracy of our results.

In both cases, the general and the superelliptic, the implementations of our algorithms significantly extend the range of algebraic curves and precisions for which integration of differential forms can be performed in reasonable time while being numerically stable.

Finally, we discuss some applications of the aforementioned algorithms in arithmetic geometry. These include the famous Birch and Swinnerton-Dyer conjecture, the problem of finding isogenies between Jacobians, computing the well-known Riemann Theta function and also computing the canonical height on abelian varieties.

Our implementations have already been applied to compute hundreds of thousands of endomorphism rings (in joint work with Jeroen Sijsling) for the 'L-functions and modular forms database' and to gather new numerical evidence for Beilinson's conjecture on K_2 of curves.

Acknowledgements

Foremost, I want to thank my advisor Steffen Müller for excellent supervision, guidance and support throughout my doctoral studies. He did a great job guiding my research and encouraging me to visit international workshops and conferences, which connected me with mathematicians internationally. In many aspects I could profit from Steffen's wisdom and I was very fortunate to become his PhD student. I want to thank Florian Hess for all the efforts that he put into supporting my research. His mathematical insights, algorithmic ideas and programming support significantly influenced and motivated my mathematical work. His door was always open and whenever I needed help he took the time to deal with my problems.

I would like to thank my collaborator Pascal Molin with whom I wrote my first scientific paper and who helped me carry out the generalization of his ideas. During my numerous visits in Paris, I was able to greatly profit from Pascal's experience in mathematics, in particular from his programming skills and algorithmic expertise. I want to thank Jeroen Sijsling for continuously using my algorithms, for pointing out interesting examples and bugs in my code to me. He endorsed my work on several occasions and got me involved with the LMFDB. Moreover, I want to thank Nils Bruin for several discussions about period matrix computations and Alexey Chernov for his advice on numerical integration.

I am grateful for having had such nice colleagues, especially my office mates Matthias Junge, Pinar Kilicer and Dietrich Kuhn, for mathematical and non-mathematical discussions and for making my time in the institute for mathematics in Oldenburg so enjoyable. For valuable proofreading, I would like to thank Nelly El Ashkar, Malte Behr, Yauheniya Filipyeva, Matthias Junge, Dietrich Kuhn, Birte Kramer and Erik-Marc Schetzke.

I want to thank my family, especially my parents, for supporting my career and for being there for me in difficult times. Without them I would not be the person I am today. Finally, I'm deeply grateful for having Yauheniya as my partner, who has been sharing her life with me for so many years. I'm sincerely thankful for her unconditional support and for her being a steady source of motivation and inspiration.

I wish to thank the Carl von Ossietzky University Oldenburg for funding my position as part of the PhD program OLCRYPT. Moreover, I acknowledge financial support from the DAAD (Deutscher Akademischer Austauschdiesnt) under grant 57212102.

Contents

1 Introduction **1**
- 1.1 Outline of the thesis . . . 4
- 1.2 Existing work . . . 5
- 1.3 Complexity . . . 6
- 1.4 Main results & contributions . . . 9
  - 1.4.1 Compact Riemann surfaces . . . 9
  - 1.4.2 Superelliptic curves . . . 10
  - 1.4.3 Applications . . . 11
  - 1.4.4 Software packages . . . 11

2 Theoretical background **12**
- 2.1 Algebraic function fields . . . 12
- 2.2 Algebraic curves . . . 13
  - 2.2.1 Algebraic varieties . . . 14
  - 2.2.2 Regular functions and morphisms . . . 15
  - 2.2.3 Non-singular curves . . . 15
  - 2.2.4 Maps between algebraic curves . . . 16
- 2.3 Divisors & Differentials . . . 17
  - 2.3.1 Divisors . . . 17
  - 2.3.2 Differentials . . . 18
  - 2.3.3 Constant field extension . . . 19
- 2.4 Riemann surfaces . . . 20
  - 2.4.1 Basic definitions . . . 20
  - 2.4.2 Properties of holomorphic maps . . . 21
  - 2.4.3 Monodromy & homotopy . . . 22
  - 2.4.4 Two constructions of Riemann surfaces . . . 24
- 2.5 Compact Riemann surfaces and algebraic curves . . . 25
  - 2.5.1 Coverings of the projective line . . . 26
  - 2.5.2 Constructing the fundamental group . . . 27
  - 2.5.3 Algebraic curves and their normalizations . . . 27
- 2.6 Integration on Riemann surfaces . . . 30
  - 2.6.1 Differential forms . . . 30
  - 2.6.2 Integration of 1-forms along paths . . . 32
- 2.7 Intersection theory . . . 34
- 2.8 Divisors and meromorphic functions . . . 35
- 2.9 The Abel-Jacobi map . . . 35
  - 2.9.1 Period matrices . . . 37

3 Numerical integration methods **38**
3.1 A versatile error bound . . . 39
3.2 Gaussian quadratures . . . 41
3.2.1 Gauss-Jacobi quadrature . . . 42
3.2.2 Gauss-Legendre quadrature . . . 44
3.2.3 Gauss-Chebyshev quadrature . . . 48
3.3 Clenshaw-Curtis quadrature . . . 49
3.4 Double-exponential integration . . . 53
3.4.1 Adaptive double-exponential integration . . . 58
3.5 A priori comparison . . . 60
3.6 Outlook . . . 61

4 Computing period matrices & the Abel-Jacobi map: general case **62**
4.1 The Riemann surface . . . 62
4.1.1 Holomorphic map to the projective line . . . 63
4.1.2 Exceptional points . . . 63
4.1.3 Ordering of the sheets . . . 64
4.2 Holomorphic differentials . . . 65
4.3 Fundamental group . . . 67
4.3.1 Choices . . . 68
4.3.2 Alternatives . . . 69
4.3.3 Types of paths & parametrizations . . . 70
4.4 Root approximation methods . . . 71
4.5 Analytic continuation & local monodromy . . . 74
4.5.1 Monodromy representation . . . 77
4.6 Computing a homology basis . . . 82
4.6.1 The Tretkoff algorithm . . . 82
4.6.2 Symplectic reduction . . . 85
4.7 Numerical integration . . . 85
4.7.1 Concatenation . . . 86
4.7.2 Gauss-Legendre & Clenshaw-Curtis quadrature . . . 87
4.7.3 Double-exponential integration . . . 88
4.7.4 Integration algorithm . . . 89
4.7.5 Improving integration paths . . . 90
4.7.6 Comparison of integration methods (I) . . . 93
4.8 Strategy for the period matrix . . . 97
4.8.1 Big and small period matrices . . . 99
4.8.2 Classifying integration paths . . . 100
4.8.3 Comparison of integration methods (II) . . . 101
4.8.4 Comparison with other implementations . . . 103
4.9 Computing the Abel-Jacobi map . . . 105
4.9.1 Moving between sheets . . . 106
4.9.2 Reaching non-critical points . . . 106
4.9.3 Integration into non-singular, critical points . . . 109
4.9.4 Moving points by strong approximation . . . 109
4.9.5 Adaptive double-exponential integration . . . 112
4.9.6 Reduction modulo the period lattice . . . 114
4.9.7 Strategy for the Abel-Jacobi map . . . 115
4.9.8 Alternatives . . . 115
4.10 Precision issues . . . 116
4.10.1 Bounding sizes of numbers . . . 116
4.10.2 Bounding differentials . . . 117

4.11 Symbolic integration . . . 118
4.11.1 Integration of differentials . . . 119
4.11.2 Practical issues and experiments . . . 119
4.11.3 Existing work . . . 120
4.12 Outlook . . . 120

5 Computing period matrices & the Abel-Jacobi map: superelliptic case 122
5.1 Superelliptic curves . . . 123
5.1.1 Definition & properties . . . 123
5.1.2 Complex roots and branches of the curve . . . 124
5.1.3 Cycles and homology . . . 127
5.1.4 Differential forms . . . 128
5.2 Strategy for the period matrix . . . 130
5.2.1 Periods of elementary cycles . . . 131
5.2.2 Numerical integration . . . 132
5.2.3 Minimal spanning tree . . . 132
5.2.4 Symplectic reduction . . . 132
5.3 Intersections . . . 133
5.4 Computing the Abel-Jacobi map . . . 138
5.4.1 Between ramification points . . . 139
5.4.2 Reaching non-ramification points . . . 140
5.4.3 Points at infinity . . . 141
5.5 Numerical integration . . . 146
5.5.1 Gauss-Chebyshev quadrature . . . 146
5.5.2 Gauss-Jacobi quadrature . . . 148
5.5.3 Double-exponential integration . . . 150
5.6 Computational aspects . . . 152
5.6.1 Complexity analysis . . . 152
5.6.2 Precision issues . . . 155
5.6.3 Implementation tricks . . . 155
5.6.4 Further ideas . . . 156
5.7 Examples, timings and comparison . . . 158
5.7.1 Big period matrix . . . 158
5.7.2 Abel-Jacobi map . . . 164
5.8 Outlook . . . 165

6 Applications 167
6.1 The Birch and Swinnerton-Dyer conjecture . . . 167
6.2 Endomorphism rings . . . 168
6.2.1 Interface with the LMFDB . . . 169
6.2.2 Isogenies between Jacobians . . . 169
6.3 Riemann Theta functions . . . 170
6.3.1 Reduced small period matrix . . . 171
6.3.2 Computing canonical heights . . . 172
6.4 Numerical verification of Beilinson's conjecture . . . 172
6.4.1 L-functions of algebraic curves . . . 172
6.4.2 K-theory and Beilinson's conjecture . . . 173
6.4.3 Strategy for checking Belinson's conjecture . . . 175
6.4.4 Numerical integration . . . 175
6.4.5 Examples of de Jeu, Dokchitser & Zagier . . . 176
6.4.6 Example of Busch . . . 179
6.4.7 Examples of Liu & de Jeu . . . 181

Appendices **190**

A Miscellaneous **191**
A.1 Closest point on an ellipse 191
A.2 Source code for the Tretkoff algorithm 192

B Declaration **199**

Chapter 1

Introduction

Riemann surfaces are objects of great interest and occur in various branches of number theory and mathematical physics; they are named after the famous mathematician Bernhard Riemann. [1] Historically, in the 19th century, the theory of Riemann surfaces arose from the study of algebraic functions and of their integrals, the so-called *abelian integrals*. The simplest examples are *elliptic integrals* of the form

$$\int \frac{\mathrm{d}x}{\sqrt{p(x)}}$$

where p is a polynomial of degree 3 or 4 with no multiple roots. These elliptic integrals occur, for example, as the circumference of an ellipse. The integral can be written as the integral of the *differential form* $\mathrm{d}x/y$ on the *elliptic curve* $C : y^2 = p(x)$, i.e.

$$\int \mathrm{d}x/y \quad \text{where} \quad y^2 = p(x)\,.$$

In fact, for a fixed value t_0, the multi-valued function defined by

$$t \mapsto \int_{t_0}^{t} \frac{\mathrm{d}x}{\sqrt{p(x)}}$$

and their inverses have been of great interest to mathematicians. In the case of elliptic integrals an explicit inverse could be constructed, namely the well known double periodic Weierstrass $\wp$-function. Its double periodicity reflects the double-valued character of the square root and the non-vanishing of the *periods* of the differential form $\mathrm{d}x/y$, i.e. of the integrals

$$\int_{\gamma} \frac{\mathrm{d}x}{\sqrt{p(x)}},$$

where γ is a closed path in $\mathbb{C}$ (or rather on C). Later on, Abel and Jacobi started to consider more general *hyperelliptic integrals* of the form

$$\int \frac{a(x)}{\sqrt{p(x)}}\mathrm{d}x$$

where $a(x)$ is a rational function and p a polynomial of degree $d \geq 3$ with no multiple roots. The expression

$$\omega = \frac{a(x)}{\sqrt{p(x)}}\mathrm{d}x$$

[1] The first part of this introduction was inspired by [10, II.1]

defines a *meromorphic differential* (i.e. it might have *poles*) on the *Riemann surface* C_f defined by the *affine equation*

$$f = 0 \quad \text{with} \quad f = y^2 - p(x) \in \mathbb{C}[x, y]\,.$$

This differential extends to a meromorphic differential on the associated *compact Riemann surface* $\hat{C}_f$ (see §2.5.3), whose *genus* is $g = \lfloor \frac{d+1}{2} \rfloor - 1$. The differential is *holomorphic* (i.e. it has no pole) on $\hat{C}_f$ if and only if $a(x)$ is a polynomial of degree at most $g-1$. Hence, the differential forms

$$w_i = \frac{x^{i-1}}{y}\mathrm{d}x\,, \quad i = 1, \ldots, g$$

and their integrals $\int \omega$ are the natural generalizations of the elliptic case. Again, one would attempt to invert the multi-valued functions

$$t \mapsto \int_{t_0}^{t} \frac{a(x)}{\sqrt{p(x)}}\mathrm{d}x$$

where $a(x)$ is a polynomial of degree $\leq g-1$. However, as soon as $g \geq 2$, the set of periods

$$\left\{ \int_\gamma \frac{a(x)}{\sqrt{p(x)}}\mathrm{d}x \;\mid\; \gamma \text{ closed path on } C_f \right\}$$

is a dense subset of $\mathbb{C}$ (at least for a generic polynomial $a(x)$). Therefore, elementary functions cannot occur as an inverse of hyperelliptic integrals. It was Jacobi who discovered this and he formulated the *inversion problem* as follows: For g given points $t_1, \ldots, t_g$ find an inverse to the function

$$(t_1, \ldots, t_g) \mapsto (I_1, \ldots, I_g)$$

with values in $\mathbb{C}^g$ given by

$$I_i = \int_{t_0}^{t_1} \omega_i + \cdots + \int_{t_0}^{t_g} \omega_i\,, \quad i = 1, \ldots, g\,.$$

However, there are ambiguities in the definition of these integrals, namely

- the square root $\sqrt{p(x)}$ is a multi-valued function; this problem is solved by looking at the t_j as points $P_j = (t_j, \sqrt{p(t_j)})$ on the Riemann surface C_f;
- the integrals I_i are only well-defined when considered modulo the periods $\int_\gamma \omega_i$, where γ is a closed path on C_f.

Denoting the vector of holomorphic differentials by $\bar{\omega} = (\omega_1, \ldots, w_g)$ leads to the definition of the *period lattice*

$$\Lambda = \left\{ \int_\gamma \bar{\omega} \;\mid\; \gamma \text{ closed path on } C_f \right\} \subset \mathbb{C}^g\,,$$

which is a lattice in $\mathbb{C}^g$. Now, we have an analytic map

$$C_f \times \cdots \times C_f \to \mathbb{C}^g/\Lambda,\ (P_1, \ldots, P_g) \mapsto (I_1, \ldots, I_g) \mod \Lambda\,.$$

Jacobi proved that this map is surjective (Theorem 2.9.3), while Abel determined its kernel (Theorem 2.9.2). The complex torus $\mathbb{C}^g/\Lambda$ is called the *Jacobian* of the compact Riemann surface $\hat{C}_f$.

In this thesis we are interested in numerically computing superelliptic integrals (which generalize hyperelliptic integrals by allowing m-th roots instead of only square roots). For

such integrals we developed a very fast and reliable algorithm that is explained in Chapter 5. We also present algorithms for the integration of differential forms in the case of general *compact Riemann surfaces*, which is described in Chapter 4.

A Riemann surface X is a topological space of dimension one that is locally homeomorphic to the complex plane. If X is compact, it is diffeomorphic to a g-holed torus (see Figure 1.1), where g is a non-negative integer called the *genus* of X.

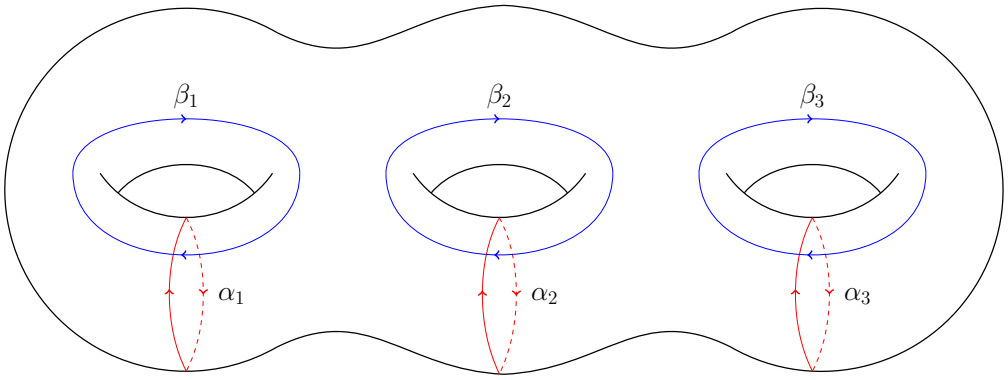


Figure 1.1: Canonical homology basis of a compact Riemann surface of genus 3.

Exploiting the local structure of X it is possible to generalize the concept of integration from contour integration of usual complex valued functions in the complex plane to integration of *differential forms* along *smooth paths* on Riemann surfaces. Generally, we will consider a basis $\bar{\omega} = (\omega_1, \ldots, \omega_g)$ for the *space of holomorphic differentials* $\Omega^1(X)$ with corresponding *period lattice*

$$\Lambda = \left\{ \int_\gamma \bar{\omega} \mid \gamma \in H_1(X, \mathbb{Z}) \right\} \subset \mathbb{C}^g \,.$$

The *first homology group* $H_1(X, \mathbb{Z})$ is an abelian group of rank $2g$, consisting of certain equivalence classes of closed paths on X, called *cycles*. Now, for a fixed base point $P_0 \in X$, integration on X is a well-defined map modulo the period lattice

$$X \to \mathbb{C}^g/\Lambda, \; P \mapsto \int_{P_0}^{P} \omega \mod \Lambda \,,$$

referred to as *Abel-Jacobi map*. It links a Riemann surface to the *complex torus* $\mathbb{C}^g/\Lambda$, which is called the *Jacobian* of X and denoted $\mathrm{Jac}(X)$.

Integrating differential forms along cycles makes it possible to define *period matrices*. In particular, integrating a basis of holomorphic differentials $\bar{\omega}$ along a *canonical homology basis* (α_i, β_i) of $H_1(X, \mathbb{Z})$ defines a *big period matrix*, while taking a basis of differentials that is dual to the α-cycles results in a *small period matrix* $\tau \in \mathcal{H}_g$ in the *Siegel upper-half space* of complex symmetric matrices with positive imaginary part. One of the most important functions in the theory of Riemann surfaces, the *Riemann Theta function*, is defined in terms of a small period matrix

$$\Theta : \mathbb{C}^g \times \mathcal{H}_g, \; (z, \tau) \mapsto \sum_{v \in \mathbb{Z}^g} \exp(\pi i v^T \tau v) \exp(2\pi i v^T z) \,.$$

Theta functions are objects of central interest in many areas of mathematics and physics. In the context of number theory and cryptography, period matrices appear in

the Birch and Swinnerton-Dyer conjecture [34], as a tool to identify isogenies between Jacobians [93], in particular endomorphisms [21], or to find equations for algebraic curves having prescribed complex multiplication [91], [51]. The Abel-Jacobi map may be used to identify divisors of meromorphic functions, or in conjunction with Theta functions to compute archimedean canonical local heights on abelian varieties [67]. Integration of differential forms on Riemann surfaces is also required to compute the regular pairing for the second K-group of algebraic curves [29]. For such applications one often needs to identify exact relations between periods, e.g. using the LLL-algorithm, for which it might be necessary to compute these integrals to large precision (hundreds, or even thousands of decimal digits). Moreover, it is desirable to have provable algorithms that yield rigorous results.

Riemann surfaces also appear in the theory of partial differential equations. In particular, the authors of [24] and [36] consider the Korteweg-de Vries (KdV) equation, the non-linear Schrödinger (NLS) equation, the Kadomtsev-Petviashvili (KP) equation and the Sine-Gordon (SG) equation. Solutions of such differential equations can often be described in terms of period matrices, the Abel-Jacobi map and the Riemann Theta function.

This work deals with *efficient integration* of differential forms *on Riemann surfaces and* it is motivated by its *applications* in number theory. In particular, we focus on computing a basis for the period lattice, big and small period matrices and the Abel-Jacobi map. Although much work has already been done on computations with Riemann surfaces, most of the existing algorithms lack the ability of achieving high numerical precision in reasonable time, do not provide rigorous error bounds or are simply prone to errors and numerical instabilities. Our goal was to have an implementation that significantly improves this situation.

1.1 Outline of the thesis

In order to provide a general map for the reader, we outline the structure of this thesis here. An overview over already existing work related to the topic is given in Section 1.2. In Section 1.3 we lay the foundation for analyzing complexity, which makes up a significant part of our work, while our main results and contributions are presented in Section 1.4.

A brief review of the necessary theoretical background is the topic of Chapter 2. We introduce algebraic function fields in Section 2.1 and algebraic curves in Section 2.2 quite generally, but only as far we need them. The rest of the chapter is devoted to Riemann surfaces and their relation to algebraic curves. Naturally, we focus on the aspects that will be algorithmically relevant for us.

In Chapter 3 we revisit methods for numerical integration, namely Gauss quadratures in Section 3.2, Clenshaw-Curtis quadrature in Section 3.3 and double-exponential integration in Section 3.4, in several variations. We are particularly interested in their applicability in a multi-precision setting, rigorous error bounds, efficient ways of computing the integration schemes and the corresponding complexities.

We explain in detail our work on computing period matrices and the Abel-Jacobi map for compact Riemann surfaces in Chapter 4. The focus lies primarily on arbitrary precision, efficiency and rigor. The corresponding algorithms are presented quite explicitly, accompanied by complexity analyses and timings. We compare the performance of different integration methods applied to the problem of integrating differential forms and test our period matrix algorithm in MAGMA [9] against other existing multi-precision implementations.

Chapter 5 is based on a paper [65] that is joint work with Pascal Molin. We consider the same problem as before, but in the special case of superelliptic Riemann surfaces.

We take advantage of numerous simplifications that arise from the special form of the input to obtain an extremely fast, rigorous algorithm that computes period matrices and the Abel-Jacobi map for such Riemann surfaces. We give a detailed complexity analysis, compare the performance of different integration methods in MAGMA and ARB [47] and the performance of other implementations.

The final part, Chapter 6, explains how our algorithms can be applied to problems occurring in active areas of mathematical research, in particular number theory. We point out the relation of period matrices to the Birch and Swinnerton-Dyer conjecture in Section 6.1 and explain how they can be used to compute isogenies between Jacobians, in particular endomorphisms, in Section 6.2. One of the most important and direct applications of period matrices is the Riemann Theta function, discussed in Section 6.3, which is actually defined with respect to a period matrix. Lastly, in Section 6.4, we apply our algorithms to integrate meromorphic differentials which play a role in the K-theory of algebraic curves. We numerically verify the Beilinson conjecture for many new hyperelliptic examples and compute regulators of higher genus non-hyperelliptic curves.

1.2 Existing work

Computing with Riemann surfaces is not a new topic and much work has been done in the recent years. Here, we want to give an overview of existing methods and algorithms and their capabilities.

For small genera, in particular 1 and 2, methods based on isogenies (AGM [22], Richelot [11], Borchardt mean [55]) make it possible to compute small period matrices and the Abel-Jacobi map to arbitrary precision in quasi-linear time. However, these techniques rely on relations between small period matrices and the Theta function, and are thus inherently exponential in the genus.

For modular curves, the modular symbols machinery and term-wise integration of expansions of modular forms give excellent algorithms for integration of differential forms [61, §3.2].

For hyperelliptic curves of arbitrary genus, there is a MAGMA package due to van Wamelen [93] that computes period matrices and the Abel-Jacobi map. It also provides functionality for analytic Jacobians in general, e.g. Theta functions, isogenies between Jacobians and endomorphism rings of Jacobians) and can be used in conjunction with our methods. However, it is limited in terms of precision (less than 2000 digits) and some bugs are experienced on certain configurations of branch points.

For compact Riemann surfaces given by irreducible affine equations, there is the MAPLE package *algcurves*, that was developed by Deconinck and van Hoeij [27], whose approach we are going to pick up and refine in Chapter 4. It seems that their package is no longer maintained properly and the code produces errors quite frequently. There is also a re-implementation of their methods in SAGE due to Swierczewski [82], the package *abelfunctions*. These packages compute period matrices, the Abel-Jacobi map [24], Theta functions [25] and the Riemann constant vector [26]. However, we found that these packages are not suitable for high precision purposes.

We also mention the MATLAB packages owed to Frauendiener and Klein for hyperelliptic curves [36] and for compact Riemann surfaces [37]. Their work is motivated by applications in physics, in particular solutions of partial differential equations. They approach Riemann surfaces entirely numerical and they are neither interested in having rigorous algorithms nor high precision.

There is another more recent implementation that computes period matrices starting from an affine equation in SAGE, but not the Abel-Jacobi map. It is due to Nils Bruin

and Alexandre Zotine and generalizes van Wamelen's approach for hyperelliptic curves to general compact Riemann surfaces. Their algorithm works quite well for moderate precisions and well-behaved examples.

1.3 Complexity

Throughout this thesis we will analyze the complexity of the algorithms that we implemented in terms of binary operations. Since our approach is largely based on multi-precision arithmetic of real (or complex) numbers, most of the run time is spent on multiplication and division of such numbers, while additions and subtractions only play a minor role. In the following, we denote by $D \in \mathbb{Z}_{\geq 0}$ the precision (a number of digits) to base $e = \exp(1)$, while $D_{10} \in \mathbb{Z}_{\geq 0}$ denotes a number of decimal digits. The complexity of an algorithm, given in Big-O notation (see (1.1)), is independent of the base, as the number of digits with respect to different bases only differ by a constant. Whenever we give absolute run times of algorithms depending on the precision, it will be a number of decimal digits.

By *computing* or *approximating* a real (or complex) number x *to precision* D we mean computing a number $\tilde{x} \in \mathbb{R}$ such that $\tilde{x} \in]x - e^{-D}, x + e^{-D}[$ (or $\tilde{x} \in \mathbb{C}$ that lies inside a disc with radius e^{-D} around x) , i.e.

$$|x - \tilde{x}| < e^{-D} .$$

We call $\tilde{x}$ an *approximation* to x and $|x - \tilde{x}|$ the *approximation error*. This notion of approximation readily generalizes to real (complex) vectors and matrices by requesting that the worst approximation among the entries is still to precision D or, in other words, that the maximal approximation error is smaller than e^{-D}. Analogous terminology is used for D_{10} and $10^{-D_{10}}$.

For the sake of completeness, we define the *Big-O notation* for functions of several variables. Let f, g be real-valued functions on a subset $U \subset \mathbb{R}^n$; we write

$$g(x) = O(f(x)) \;\Leftrightarrow\; \exists\, c, d > 0 : \forall\, \tilde{x} \in U \text{ with } \|\tilde{x}\|_\infty \geq d : |f(\tilde{x})| \leq c\,|g(\tilde{x})| \; . \tag{1.1}$$

We consider the following complexities for multi-precision numbers of size D digits, as given in [12]. Operations can always be thought of as binary operations (although any other base is fine).

- Addition and checking equality are linear in the precision and take $O(D)$ operations.
- Multiplication, denoted by $\mathcal{M}(D)$, depends on the algorithm that is used; for our complexity analyses we assume that $\mathcal{M}(D) := O(D \log^{1+\varepsilon} D) = O(D \log D \log\log D)$ using the Schönhage-Strassen algorithm for a suitable $\varepsilon > 0$.
- The cost of divisions is $\mathcal{M}(D)$ using Newton-Raphson division, but they are strictly (in MAGMA between 2 to 3 times) more expensive than multiplications.
- Evaluating elementary functions costs $\mathcal{T}(D) := O(\log D \mathcal{M}(D)) = O(D \log^{2+\varepsilon} D)$ using the arithmetic-geometric mean (AGM); this includes $\exp, \log, \sin, \cos$ and their inverses as well as the hyperbolic functions $\sinh, \cosh$ and their inverses.
- We assume the cost of computing m-th roots to be $\mathcal{T}(D)$ using $\exp(\frac{1}{m}\log(\cdot))$; but they could also be computed in $O(\log m \mathcal{M}(D))$ using Newton's method.
- Computing complex roots of a univariate polynomial of degree m takes $\mathcal{R}(m, D) := O(m \log^2 m (\log^2 m + \log D)\mathcal{M}(D))$ operations using an algorithm of Pan (see §1.3);

- while evaluation of univariate polynomials of degree m costs $O(m\mathcal{M}(D))$.

Since the big O-notation ignores constants, we keep in mind that complex multiplications are considered about 2 times more expensive than real ones.

Moreover, we assume that $m \times m$-Matrix multiplication and inversion have complexity $O(m^\omega)$ where $\omega < 2.373$ is the matrix multiplication constant.

Newton's method Newton's method is a very important tool in real and complex arbitrary-precision arithmetic (for more details see [12, §4.2]) and will be used throughout this thesis. Thus, it is crucial for our analysis to shed some light on the complexity of this method.

Let f be a (real or complex-valued) function and η a simple zero of f, i.e. such that $f(\eta) = 0, f'(\eta) \neq 0$ and f is differentiable in a neighborhood of η. For an initial approximation $z^{(0)}$ to η, Newton's method is defined by iteratively applying the well-known formula

$$z^{(k+1)} = z^{(k)} - f(z^{(k)})/f'(z^{(k)}) \quad (k \geq 0). \tag{1.2}$$

If $|z^{(0)} - \eta|$ is sufficiently small, $z^{(k)}$ is expected to converge to η with *order of convergence* at least two, which means

$$|z^{(k+1)} - \eta| = O(|z^{(k)} - \eta|^2).$$

If this is the case we need at most $O(\log D)$ iterations to approximate η to precision D. In the case where f is a complex polynomial the notion of 'good initial approximation' can be made explicit, see for example [72, Chapter 2].

If every iterative step is performed in precision D, Newton's method has complexity $O(\log D\mathcal{F}(D))$ where $\mathcal{F}(D)$ is the cost of calculating $f(z^{(k)})/f'(z^{(k)})$ in precision D. However, if we are able to evaluate f and f' at z at variable precision, we can start the iterative process at a suitable low precision and double it in each step such that only the last step is performed with precision D numbers. Since $\mathcal{F}(D)$ has to grow at least linearly with D, using a geometric series, the computational cost is reduced to

$$\begin{aligned} &\mathcal{F}(D) + \mathcal{F}(D/2) + \mathcal{F}(D/4) + \dots \\ &\leq \mathcal{F}(D)(1 + 1/2 + 1/4 + \dots) \\ &\leq 2\mathcal{F}(D) = O(\mathcal{F}(D)). \end{aligned} \tag{1.3}$$

The complexity $O(\mathcal{F}(D))$ for Newton's method is assumed for the complexities of divisons and m-th roots, as given above, and for any other applications of Newton's method in this thesis.

Finding complex roots In several places we need to determine the complex roots of a univariate polynomial of degree m up to some desired precision D. Much work has been done on this topic and we will not go into much depth here but give some relevant references; for this we follow [55, §9.2.1]. A nice overview over existing root-finding methods can be found in [62]. In MAGMA we use the command 'Roots' to compute polynomial roots numerically. Here one can choose between the algorithms 'Schönhage','Laguerre' and 'NewtonRaphson'. In practice, we will always use the Schönhage algorithm [78] (which is also the default option) which has complexity $O(m^3 \log m + m^2 D)$. The implementation of this algorithm in MAGMA is due to Gourdon and is discussed in detail in his thesis [41]. An even better complexity is claimed by Pan in [71], namely

$$\mathcal{R}(m, D) = O(m \log^2 m(\log^2 m + \log D)\mathcal{M}(D)),$$

but since it is not clear whether this algorithm is actually implemented anywhere, this result seems to be rather theoretical. Nonetheless, we are going to use the complexity $\mathcal{R}(m, D)$ for our complexity analysis. It is not our goal to improve the computation of complex roots in this work, but since we cannot really avoid it is good to be aware of its impact on the complexity.

Precision loss For our complexity analyses, we will not take into account precision loss due to rounding errors produced by the floating-point arithmetic in MAGMA. This is not an issue in practice and can be prevented by throwing in a small number of guard digits. Nonetheless, we do care about predictable precision loss that is caused by overflow occurring during the computations. The goal is to compute, a priori, a (minimal) *working precision* (or *internal precision*) $\tilde{D} > D$ such that the end result is correct to precision D. However, the ways in which the coefficients of the affine equations $f(x, y) = 0$ induce precision loss are heavily interwoven with several other choices and cannot be easily identified explicitly. In the general case, we analyze precision loss in terms of the input (and our choices) as far as possible in Section 4.10. In the case of superelliptic curves, where the situation is clearer, we give a short exposition on precision loss in §5.6.2. We will also comment on this issue in Section 1.4 below.

Heuristic assumptions Here we want to clarify the quality of the complexity statements made in this thesis that involve multi-precision arithmetic. In order to make useful statements we had to compromise by making some *heuristic assumptions* which are listed below. In particular, the complexity statements of Chapter 4 and Section 5.6, as well as Theorem 1.4.1, Corollary 1.4.2, Theorem 1.4.3, Theorem 1.4.4 and Theorem 1.4.5 are understood to be *heuristic* in the following sense:

- the expression 'to precision D' means that the algorithm achieves a mathematical error smaller than e^{-D};
- we do not take into account numerical errors due to rounding or cancellation;
- we do not differentiate between working precision $\tilde{D}$ and output precision D, but assume that for a fixed polynomial $f \in \mathbb{C}[x, y]$ as input, we have that
$$\tilde{D} - D = O(1) \quad \text{in } D\,.$$

These heuristic assumptions reflect what is happening in practice and enable us to formulate comprehensible complexity statements that remain mathematically meaningful. We provide evidence that $\tilde{D} - D$ is independent of D in Section 4.10 and §5.6.2.

Note that we usually have as input an exact polynomial $f \in K[x, y]$ over a number field K and a desired output precision D. Before the numerical part of the computation starts, we estimate the working precision $\tilde{D}$ that is actually required to achieve precision D. For all further considerations we assume that the input, be it $f \in \mathbb{C}[x, y]$ or some $P \in \mathbb{C}^2$, is always given to sufficiently large precision $\tilde{D}$.

Computational setup The algorithms described in this thesis are part of software packages for the computer algebra system MAGMA [9]. More precisely, we used MAGMA versions 2.22-4 and 2.23-5. In fact the reader may assume that, unless mentioned otherwise, we did not rely on already existing MAGMA functionality, but used our own implementations. The period matrix algorithm described in Chapter 5 has also been implemented in the C-library ARB [47], in version 2.12.0. For the timings in §4.8.4 we used MAPLE 17 [60] and SAGE 8.0 [84]. Furthermore, all computations, for which timings are given in this work, were carried out on an Intel Xeon(R) CPU E3-1275 V2 3.50GHz processor using a single core.

1.4 Main results & contributions

In this section we want to present the main results of this thesis and the impact it already had or will have on mathematical research. Informally speaking, our algorithms vastly increase the range of algebraic curves for which period matrices can be computed in reasonable time. Compared to other existing multi-precision implementations that rely on numerical integration, we gain a huge speed-up on period matrix computations and the Abel-Jacobi map in every genus and for every precision. This is mostly due to efficient numerical integration of differential forms, using different integration methods and applying them according to their strengths and weaknesses. A lesser role, which is still worth mentioning, is very careful coding and lots of algorithmic optimization that went into our implementations. Moreover, our algorithms are numerically stable and robust (they do not rely on the niceness of the input). In the case of superelliptic curves we obtain a completely rigorous algorithms that enables us to compute periods with thousands of correct digits, even for very high genus.

Several of the methods that we apply have already been used by other authors. However, we give detailed complexity analyses for all significant algorithms presented in this thesis. In the following, we present the most important of these complexity results. Recall from Section 1.3 that we assume $\varepsilon > 0$ to be chosen such that multiplication of precision D numbers has computational complexity $\mathcal{M}(D) = O(D \log^{1+\varepsilon} D)$.

1.4.1 Compact Riemann surfaces

Our work on compact Riemann surfaces is described in Chapter 4. Most importantly, we implemented a fast and reliable algorithm (see Algorithm 4.8.1) for computing period matrices of compact Riemann surfaces, given by an irreducible affine equation $f(x, y) = 0$ defined over a number field, to arbitrary precision. Assuming a basis of holomorphic differentials as given, we obtain the following result (see Corollary 4.8.3 for the proof).

Theorem 1.4.1. *Let X be a compact Riemann surface of genus $g > 0$ defined by an affine equation $f(x, y) = 0$, $f \in \mathbb{C}[x, y]$ irreducible of total degree d, and $(\omega_1, \ldots, \omega_g)$ a standard basis* (4.8) *of the space of holomorphic differentials $\Omega^1(X)$. There is an algorithm that (heuristically) computes a basis of the period lattice Λ to precision $D > 0$ using*

$$O(c(f)d^2D^2(\log D + d^4)\log^{1+\varepsilon} D + d^7 D) \quad \textit{operations,}$$

where $c(f) = 1/r$, for some $r > 0$ (see §4.7.2), depends on the configuration of exceptional values (4.2)*. In particular, r depends on their absolute values and absolute and relative distances between them, see Section 4.7.*

Corollary 1.4.2. *Fixing one affine equation $f(x, y) = 0$ and all related parameters, we (heuristically) compute a basis of the period lattice Λ to precision $D > 0$ using*

$$O(D^2 \log^{2+\varepsilon} D) \quad \textit{operations.}$$

This is precisely the cost of performing one numerical integration using Clenshaw-Curtis quadrature (Section 3.3). Note that, while Clenshaw-Curtis is rarely a good option in practice, it yields the best complexity results.

The predictable precision loss is discussed in Section 4.10. It depends on the exceptional values, the basis of differentials and several other choices made during the algorithm. We can prevent precision loss due to overflow by setting the internal precision to $\tilde{D} = D + \max\{\tilde{D}_1, \tilde{D}_2\}$ where $\tilde{D}_1$ and $\tilde{D}_2$ are given by equations (4.40) and (4.42). The precision loss for the Abel-Jacobi map can be handled analogously.

The second important result is our arbitrary-precision implementation of the Abel-Jacobi map of compact Riemann surfaces. With X and $\bar{\omega}$ as in Theorem 1.4.1, we obtain the following complexity result (see Corollary 4.9.3 for the proof).

Theorem 1.4.3. *For all but finitely many points $P = (x_P, y_P) \in X$, we (heuristically) compute $\int_{P_0}^{P} \bar{\omega}$ to precision $D > 0$ using*

$$O(c(f, x_P) D^2 (\log D + d^3) \log^{1+\varepsilon} D + d^4 D) \quad \textit{operations,}$$

where $c(f, x_P) = 1/r$, for some $r > 0$ (see §4.7.2), depends on the distances between x_P and the exceptional values (4.2).

We also compute the Abel-Jacobi map for all the points not included in this complexity statement. This is discussed extensively in Section 4.9.

1.4.2 Superelliptic curves

Chapter 5 is largely based on the paper 'Computing period matrices and the Abel-Jacobi map of superelliptic curves' [65] which is joint work with Pascal Molin. The paper is accepted by the journal Mathematics of Computation. There, we specifically compute period matrices and the Abel-Jacobi map of compact Riemann surfaces associated to affine equations of the form

$$y^m = p(x), \quad m > 1, \, p \in \mathbb{C}[x] \text{ separable of degree } \deg(p) = n \geq 3. \tag{1.4}$$

They can be viewed as a generalization of hyperelliptic curves and we refer to them as *superelliptic curves*. Using the advantages that are offered by the specific geometry of superelliptic curves, we obtain the following complexity result, which is a direct consequence of Theorem 5.6.1.

Theorem 1.4.4. *Let C be a superelliptic curve of genus g defined by an equation* (1.4). *There is an algorithm that (heuristically) computes a basis of the period lattice Λ to precision $D > 0$ using*

$$O(c(p)\, n\, D^2 \log^{k+\varepsilon} D(g + \log D)) \quad \textit{operations, with } \begin{cases} k = 1, \textit{ if } m = 2, \\ k = 2, \textit{ if } m > 2, \end{cases}$$

where $c(p) = 1/r$, for some $r > 0$ (see §5.5.1 and §5.5.3), depends on the roots of p. In particular, r depends on their absolute values and absolute and relative distances between them, see Section 5.5.

The precision loss due to overflow is discussed in §5.6.2. Up to a constant factor that depends on the (absolute distances between the) roots of p, we lose $O(g)$ digits of precision. Similarly, we can estimate the precision loss occurring for the Abel-Jacobi map.

Finally, we give a partial result for the Abel-Jacobi map of superelliptic curves here. The complexity statement for the complete Abel-Jacobi map is given in Theorem 5.6.3.

Theorem 1.4.5. *For all but finitely many points $P = (x_P, y_P) \in C$, we (heuristically) compute $\int_{P_0}^{P} \bar{\omega}$ to precision $D > 0$ using*

$$O(c(p, x_P) D^2 \log^{2+\varepsilon} D(g + \log D) + ngD) \quad \textit{operations,}$$

where $c(p, x_P) = 1/r$, for some $r > 0$ (see §5.5.3), depends on the distances between x_P and the roots of p.

1.4.3 Applications

Here we only indicate the applications of our algorithms have already been carried out during this work. Further possible applications and how they might be realized are described in Chapter 6.

Endomorphism rings As described in Section 6.2, our period matrix algorithms were used to compute the endomorphism rings of Jacobians for many genus 3 curves (hyperelliptic & non-hyperelliptic) that are going to appear in the LMFDB [58]. This was done in collaboration with Jeroen Sijsling, using the methods of Costa et al. [21].

Numerical verification of Beilinson's conjecture In Section 6.4 we present our work related to Beilinson's conjecture. In order to compute the regulator pairing for the second K-group of algebraic curves, one has to integrate a certain meromorphic differential which can be done with a minor tweak to our integration methods for holomorphic differentials. Based on the works of [29], [14] and [23], we numerically verify Beilinson's conjecture for many new hyperelliptic examples and also provide numerical data for regulators of higher genus non-hyperelliptic curves.

1.4.4 Software packages

All algorithms presented in this work are implemented and tested extensively. The author has written two software packages for the computer algebra system MAGMA [9]. More precisely,

- one for the case of general compact Riemann surfaces, which contains all the algorithms that are described in the Chapters 3 and 4; it will be available as part of the MAGMA distribution in the near future;
- one for the case of superelliptic curves that realizes the ideas of Chapter 5; there is second implementation in ARB [47] due to Pascal Molin with only minor contributions from the author's side; both are publicly available on github at [66].

Chapter 2

Theoretical background

In this chapter we introduce the basic theory for the mathematical structures that will occur throughout this work, namely *Riemann surfaces*, *algebraic curves* and *algebraic function fields*. Although we start this chapter by introducing algebraic function fields and algebraic curves, the structures that are most central to this work are Riemann surfaces and will be introduced last. As it turns out, the isomorphism classes of these objects are closely related through category equivalences.

For the theory of Riemann surfaces presented in Section 2, we rely on the books of Miranda [63], Farkas & Kra [33], Bost [10] and the lecture notes of Bobenko [7]; a great source to read up about plane algebraic curves and their relation to Riemann surfaces is the book [13] by Brieskorn and Knörrer.

We are in the situation where we could formulate the theoretical part entirely in the language of Riemann surfaces, i.e. over the (algebraically closed) field of complex numbers, but in our applications we deal with algebraic curves defined over number fields, which are not algebraically closed. Since we want to properly introduce algebraic curves over non-algebraically closed fields of characteristic zero, we decided to follow the book of Niederreiter and Xing [70], which provides this kind of introduction whilst avoiding the scheme-theoretic language used by other standard sources for algebraic geometry, such as [57] or [43]. Additionally, in Chapter 4, we rely on MAGMA's function fields functionality on several occasions, which requires defining polynomials to be defined over exact fields. Therefore, it is compelling to introduce algebraic function fields over non-algebraically closed fields as well. In addition to [70], we recommend the book 'Algebraic function fields and codes' of Stichtenoth [80] for further details.

In the following we will assume that K is a field of characteristic zero (everything could be done in more generality, e.g. for perfect fields, but we do not need that in this thesis). Moreover, we denote by $\overline{K}$ a fixed algebraic closure of K.

2.1 Algebraic function fields

We briefly introduce the theory of algebraic function fields as far as it is useful to us. Later on, we will solely consider function fields defined by affine equations $f(x, y) = 0$ where $f \in K[x, y]$ is geometrically irreducible and K is either a number field or $\mathbb{C}$.

An *algebraic function field* F/K of one variable over K is an extension field $F \supseteq K$ such that F is a finite algebraic extension of the rational function field $K(x) = \mathrm{Quot}(K[x])$. Function fields are often represented as a simple algebraic field extension $F = K(x, y)$ where $f(y) = 0$ for an irreducible polynomial $f \in K(x)[y]$. The field $\tilde{K} := \{\, x \in F \mid z \text{ algebraic over } K \,\}$ is called the *constant field* of F/K; we say that K is the *full constant field* of F if $\tilde{K} = K$.

A *place* P of the function field F/K is the maximal ideal of some valuation ring $\mathcal{O}$ of F/K. Every element $t \in P$ such that $P = t\mathcal{O}$ is called *local parameter* for P. The valuation ring

$$\mathcal{O} = \{ z \in F \mid z^{-1} \notin P \}$$

is uniquely determined by the place P and we call $\mathcal{O}_P := \mathcal{O}$ the *valuation ring of* P. Moreover, we denote by M_F the set of places of F/K.

Let $P \in M_F$ and t be a local parameter for P. We associate to P a discrete valuation $v_P : F \to \mathbb{Z} \cup \{\infty\}$ via

$$v_P(z) = \begin{cases} n, & \text{if } z = t^n u \text{ for some } u \in \mathcal{O}_P^*, \\ \infty, & \text{if } z = 0. \end{cases}$$

Note that, among other properties, the discrete valuation v_P satisfies the *triangle inequality*

$$v_P(y + z) \geq \min\{v_P(y), v_P(z)\}\,.$$

In case of equality (i.e. whenever $v_P(y) \neq v_P(z)$) it is called *strict triangle inequality.*

Let P be a place of F. We call $F_P := \mathcal{O}_P/P$ the *residue class field* of P. The map from F to $F_P \cup \{\infty\}$, $x \mapsto x(P)$ is called *residue class map* with respect to P. By [70, Theorem 1.5.13.] the residue class field of every place of F/K is a finite field extension of K. So we define the *degree of* P by

$$\deg(P) := [F_P : K]\,.$$

A place of degree one is called *rational place* of F/K.

The following result [70, Theorem 1.5.18.] will be a useful tool for the computation of the Abel-Jacobi map, see §4.9.4. and §5.4.3.

Theorem 2.1.1 (Approximation theorem)**.** *Let $P_1, \ldots, P_n$ be distinct places of F/K. Then, for any elements $x_1, \ldots, x_n \in F$ and integers $m_1, \ldots, m_n \in \mathbb{Z}$ there exists an element $z \in F$ such that*

$$v_{P_i}(z - x_i) = m_i \quad \text{for } i = 1, \ldots, n.$$

Definition 2.1.2. An algebraic function field F'/K' is called an *algebraic extension* of F/K if $F' \supset F$ is an algebraic extension of F/K and $K' \supset K$. The algebraic extension F'/K' of F/K is called *constant field extension* if $F' = FK'$, where FK' is the composite field of F and K'.

Consider an algebraic extension F'/K' of F/K. A place $P' \in \mathbb{P}_{F'}$ is said to *lie over* $P \in M_F$ if $P \subset P'$. We also say that P *lies under* P' and write $P'|P$. A place P of F *splits completely* in the extensions F'/F if there are exactly $[F' : F]$ places of F' lying over P.

2.2 Algebraic curves

We are now going to introduce *algebraic curves* over fields of characteristic zero, in quite some generality, as irreducible algebraic varieties of dimension one. As we already mentioned earlier, our goal here is to emphasize the connection between algebraic curves and algebraic function fields. We continue to denote by K a field of characteristic zero and by $\overline{K}$ a fixed algebraic closure of K.

2.2.1 Algebraic varieties

For this brief introduction to algebraic varieties we follow [70, Chapter 2]. Roughly speaking, in this thesis an algebraic variety is the zero locus of a finite number of polynomials in a finite number of variables. The variety is defined over K if the coefficients of the polynomials are in K.

More precisely, we define *affine (resp. projective) n-space over $\overline{K}$* to be

$$\mathbb{A}^n(\overline{K}) = \{\, (a_1, \ldots, a_n) \mid a_i \in \overline{K} \,\},$$
$$\mathbb{P}^n(\overline{K}) = \{\, [a_0 : a_1 : \ldots : a_n] \mid a_i \in \overline{K}, \text{not all } a_i = 0 \,\}$$

where

$$[a_0 : a_1 : \ldots : a_n] = \{\, (\lambda a_0, \lambda a_1, \ldots, \lambda a_n) \mid \lambda \in \overline{K}^* \,\}.$$

Likewise, we define the set of *K-rational points* of $\mathbb{A}^n(\overline{K})$ (resp. $\mathbb{P}^n(\overline{K})$) as

$$\mathbb{A}^n(K) = \{\, (a_1, \ldots, a_n) \in \mathbb{A}^n(\overline{K}) \mid a_i \in K \,\},$$
$$\mathbb{P}^n(K) = \{\, [a_0 : a_1 : \ldots : a_n] \in \mathbb{P}^n(\overline{K}) \mid a_i \in K \,\}.$$

The *absolute Galois group* of K, denoted by

$$G_K := \mathrm{Gal}(\overline{\mathrm{K}}/\mathrm{K}),$$

acts on a $\overline{K}$-rational point $P = (a_1, \ldots, a_n) \in \mathbb{A}^n(\overline{K})$ via

$$\sigma(P) = (\sigma(a_1), \ldots, \sigma(a_n)),$$

and analogously one can check that this action is well-defined for $P \in \mathbb{P}^n(\overline{K})$. The orbit of a $\overline{K}$-rational point P under this action

$$\{\, \sigma(P) \mid \sigma \in \mathrm{Gal}(\overline{\mathrm{K}}/\mathrm{K}) \,\}$$

is called *K-closed point*. Two points in a K-closed point are said to be *K-conjugate*. An affine (resp. projective) *algebraic set* is the locus of common zeros $Z(S) \subset \mathbb{A}^n(\overline{K})$ (resp. $\mathbb{P}^n(\overline{K})$) of a finite set of polynomials $S \subset \overline{K}[x_1, \ldots, x_n]$ (resp. homogeneous polynomials in $\overline{K}[x_0, x_1, \ldots, x_n]$). The algebraic set $Z(S)$ is *defined over $K \subset \overline{K}$* if the polynomials in S are defined over K. The n-spaces $\mathbb{A}^n(\overline{K})$ and $\mathbb{P}^n(\overline{K})$ become topological spaces when equipped with the *Zariski topology*, that is, the closed subsets are taken to be the algebraic sets.

A non-empty affine (resp. projective) irreducible algebraic subset $V = Z(S)$ (defined over K) is then called an *affine (resp. projective) algebraic variety (defined over K)* and denoted V/K. The condition that V is irreducible is equivalent to the ideal $\langle S \rangle$ being a prime ideal. For any algebraic variety V/K and field $L \subset \overline{K}$ we define the *L-rational points of V* as

$$V(L) := \{\, P \in V \mid P \text{ is an } L\text{-rational point} \,\}.$$

An affine (resp. projective) algebraic variety (defined over K) of dimension 1 (as a topological space) is called an *affine (respectively projective) algebraic curve (defined over K)*, and will be denoted C/K. For an affine variety V/K we define the *coordinate ring* of V as the ring

$$K[V] = K[x_1, \ldots, x_n]/\langle S \rangle,$$

which is an integral domain, and the *K-rational function field* of V as the quotient field

$$K(V) = \mathrm{Quot}(K[V]).$$

For a projective variety V defined over K there always exists an affine variety $U \subset \mathbb{A}^n(\overline{K})$ (obtained by setting $a_i = 1$ for some i) such that $V \cap \mathbb{A}^n(\overline{K}) = V \cap U \neq \varnothing$ is an affine variety defined over K. Thus, we can define the *K-rational function field* to be

$$K(V) = K(V \cap \mathbb{A}^n(\overline{K})).$$

2.2.2 Regular functions and morphisms

A function $f : V \to \overline{K}$ on an affine algebraic variety $V \subset \mathbb{A}^n(\overline{K})$ is called *regular at* P, if there exists a neighborhood W of P in the Zariski topology such that $f = g/h$ on W for polynomial $g, h \in \overline{K}[x_1, \ldots, x_n]$ with $h(Q) \neq 0$ for all $Q \in W$. If $V \subset \mathbb{P}^n(\overline{K})$ is a projective variety, a point $P \in V$ belongs to $V \cap U$ for an affine variety $U \subset \mathbb{A}^n(\overline{K})$ and $f : V \to \overline{K}$ is called *regular at* $P \in V$, if the restriction of f to $V \cap U$ is regular at P. The *ring of regular functions at* P is defined as

$$\mathcal{O}_P = \mathcal{O}_P(V) := \{ f : V \to \overline{K} \mid f \text{ regular at } P \} / \sim$$

where $f \sim g$ if and only if they coincide on some non-empty open subset of V.

Let V, W be two algebraic varieties over $\overline{K}$. A *morphism* $\varphi : V \to W$ is a continuous map such that for every open subset $U \subset W$ with $\varphi^{-1}(U) \neq \varnothing$ and every regular function f on U, the composite function $f \circ \varphi$ is regular on $\varphi^{-1}(U)$.

Example 2.2.1. Let $V \subset \mathbb{P}^n(\overline{K})$, $W \subset \mathbb{P}^m(\overline{K})$ be projective varieties and $f_0, \ldots, f_m \in \overline{K}[x_0, \ldots, x_n]$ homogeneous polynomials of equal degree such that

$$[f_0(P) : \ldots : f_m(P)] \in W \quad \text{for every } P \in V.$$

Then the map defined by

$$\varphi : V \to W,\ P \mapsto [f_0(P) : \ldots : f_m(P)]$$

is a morphism.

2.2.3 Non-singular curves

In the following a *curve* C/K denotes an affine (or projective) algebraic curve defined over K. We say that C is *non-singular* (or *smooth*) at a point $P \in C(\overline{K})$ if and only if the local ring $\mathcal{O}_P$ at P is a discrete valuation ring. Otherwise, we say that P is a *singular point* of C. If C is non-singular at every point, we say that C is *non-singular* (or *smooth*). Note that, by [70, Theorem 3.1.7.], the set of singular points of a curve C is finite.

Combining several statements from [70, Chapter 2], we obtain that the K-rational function field $K(C)$ of a curve C/K is an algebraic function field in one variable over K and the full constant field of $K(C)$ is indeed K.

For a non-singular point P of a curve C, a local parameter of the discrete valuation ring $\mathcal{O}_P$ is called *local parameter* at P. Furthermore, the valuation of the ring $\mathcal{O}_P$ is denoted by v_P.

Let now C/K be a non-singular projective curve. Then, by [70, Theorem 3.1.12], the map

$$P \mapsto \mathcal{O}_P = \mathcal{O}_P(C)$$

yields a one-to-one correspondence between the $\overline{K}$-rational points of C and the discrete valuation rings with quotient field $\overline{K}(C)$.

Moreover, for each two points P and Q on a projective curve C/K that belong to the same K-closed point of C and satisfy $Q = \sigma(P)$ for some $\sigma \in \mathrm{Gal}(\overline{\mathrm{K}}/\mathrm{K})$, by [70, Lemma 3.1.13], we have that

$$\mathcal{O}_Q(C) = \sigma(\mathcal{O}_P(C)) .$$

2.2.4 Maps between algebraic curves

A rational map $\varphi : C_1/K \to C_2/K$ between projective curves is *defined over* K if

$$\varphi \circ \sigma = \sigma \circ \varphi \quad \text{for all } \sigma \in G_K,$$

that is, if $\varphi(\sigma(P)) = \sigma(\varphi(P))$ for all $P \in \mathrm{dom}(\varphi)$ and all $\sigma \in G_K$. By [70, Theorem 3.2.3] every non-constant rational defined over K induces a homomorphism between K-rational function fields

$$\varphi^* : K(C_2) \to K(C_1),\ f \mapsto f \circ \varphi$$

that fixes K.

Two non-singular projective curves defined over K are K-isomorphic if and only if their function fields are K-isomorphic (see [70, Theorem 3.2.7]). In that case the curves are called *birationally equivalent*.

Definition 2.2.2. An algebraic curve C/K that is an algebraic subset of $\mathbb{A}^2(\overline{K})$ (resp. $\mathbb{P}^2(\overline{K})$) is called *affine (resp. projective) plane algebraic curve.*

For curves defined over $\overline{K}$ we have that, up to birational equivalence, every algebraic curve can be realized as an affine plane algebraic curve (see [70, Corollary 2.5.26]).

As shown by [70, Theorem 3.2.8], for every projective curve C/K there is a non-singular projective curve C'/K such that the following properties hold:

- there exists a birational map $\varphi' : C' \to C$;
- if φ'' is a another birational map from a non-singular projective curve C''/K to C, then there exists a unique isomorphism $\psi : C' \to C''$ (defined over K) such that $\varphi'' \circ \psi = \varphi'$.

Definition 2.2.3. [Non-singular projective model] The pair (C', φ') is unique up to K-isomorphisms and thus called the *non-singular projective model* of C.

We can now conclude by stating the equivalence of the categories of algebraic function fields in one variable and non-singular projective curves, as given by [70, Theorem 3.2.9].

Theorem 2.2.4. *There is a one-to-one correspondence between K-isomorphism classes of non-singular projective curves over K and K-isomorphism classes of algebraic function fields of one variable with full constant field K, induced by the map*

$$C/K \mapsto K(C)\,.$$

Category equivalence We want to emphasize that in this section (see Theorem 2.2.4) we established that, for a field of characteristic zero, there is a category equivalence between the categories

- of smooth projective curves C/K with non-constant regular maps and
- algebraic function fields in one variable over K with homomorphisms of K-algebras.

In fact, this is true over any field, see for example [57, Proposition 3.13].

2.3 Divisors & Differentials

Recall from Section §2.2.3, that for a given non-singular projective curve C/K, its K-rational function field $K(C)$ is an algebraic function field of one variable with full constant field K. Conversely, Theorem 2.2.4 shows that, given an algebraic function field F/K of one variable with full constant field K, there exists a non-singular projective curve C_F/K such that $F \cong K(C_F)$. This makes it possible to use statements on curves for function fields with a proper interpretation.

In the following we start from an algebraic function field F/K of one variable with full constant field K. The theory of divisors of F that we are going to introduce now can be developed in an equivalent fashion in the language of non-singular projective curves. In particular, there is a one-to-one correspondence between the places of F/K and the K-closed points of the curve C_F/K. We denote by M_F the set of places of F.

2.3.1 Divisors

The *divisor group* of F/K, denoted by $\mathrm{Div}(F/K)$ or $\mathrm{Div}(F)$, is the free abelian group generated by the places of F, that is, a *divisor* in $\mathrm{Div}(F)$, also called a divisor of F, is a formal sum

$$D = \sum_{P \in M_F} v_P P$$

with coefficients $v_P = v_P(D) \in \mathbb{Z}$ and $v_P \neq 0$ for only finitely many $P \in M_F$. Two divisors are added by adding the corresponding coefficients. A divisor D is called *effective*, written as $D \geq 0$, if $v_P(D) \geq 0$ for all $P \in M_F$. For two divisors D_1, D_2 we write $D_1 \geq D_2$ if $D_1 - D_2 \geq 0$. The *support* $\mathrm{supp}(D)$ of D is the finite set given by

$$\mathrm{supp}(D) := \{\, P \in M_F \mid v_P(D) \neq 0 \,\}.$$

The *degree* $\deg(D)$ of the divisor is defined by

$$\deg(D) := \sum_{P \in M_F} v_P \deg(P).$$

The degree deg defines a group homomorphism from $\mathrm{Div}(F)$ to $\mathbb{Z}$. The kernel of this homeomorphism is a subgroup of $\mathrm{Div}(F)$, the group of divisors of degree zero, denoted by $\mathrm{Div}^0(F)$, that is,

$$\mathrm{Div}^0(F) := \{\, D \in \mathrm{Div}(F) \mid \deg(D) = 0 \,\}.$$

By [70, Corollary 3.3.2], every element $x \in F^*$ has only finitely many zeros and finitely many poles, so we may associate to it the so-called *principal divisor*

$$\mathrm{div}(x) = \sum_{P \in M_F} v_P(x) P \in \mathrm{Div}^0(F)$$

which is a divisor of degree zero by [70, Corollary 3.4.3]. The group of principal divisors of F, defined as

$$\mathrm{Prin}(F) = \{\, \mathrm{div}(x) \mid x \in F^* \,\},$$

forms a subgroup of $\mathrm{Div}^0(F)$.

Let $D, D' \in \mathrm{Div}(F)$ be divisors of F. Then, the relation

$$D \sim D' \quad \Leftrightarrow \quad D = D' + \mathrm{div}(x) \quad \text{for some } x \in F^*.$$

defines an equivalence relation on $\mathrm{Div}(F)$, called *linear equivalence*.

Definition 2.3.1. For a divisor $D \in \mathrm{Div}(F)$ we define the *Riemann-Roch space associated to* D by

$$L(D) = \{\, x \in F^* \mid \mathrm{div}(x) + D \geq 0 \,\} \cup \{0\}\,.$$

In particular, for a divisor $D = \sum m_i P_i - \sum n_j Q_j$ with $m_i, n_j > 0$, the space $L(D)$ contains all the elements $x \in F$ such that

- x has a zero of order at least n_j at each Q_j, i.e. $v_{Q_j}(x) \geq n_j$ and
- x has poles only at the places P_i and of order at most m_i, i.e. $v_{P_i}(x) + m_i \geq 0$.

It is easy to prove that $L(D)$ is a finite dimensional K-vector space and that for equivalent divisors $D \sim D'$ we have $L(D) \cong L(D')$; we denote by $\ell(D) = \dim_K L(D)$ its dimension.

Definition 2.3.2. A divisor $D \in \mathrm{Div}(F)$ is called *non-special* if and only if

$$\ell(D) = \deg(D) + 1 - g\,.$$

Otherwise, D is called *special*.

Note that, by [80, Remark 1.6.11.], any divisor D of degree $\deg(D) > 2g - 2$ is non-special and, if D is non-special and $D' \geq D$ then D' is non-special as well.

2.3.2 Differentials

For an introduction to differentials of algebraic function fields we follow the lecture notes of Stoll [81, Chapter 6].

Definition 2.3.3. Let A be a K-algebra and M an A-module. A *K-derivation* from A to M is a K-linear map $\delta : A \to M$ such that

$$\delta(xy) = x\delta(y) + y\delta(x)\,.$$

Moreover, a pair (Ω, d) consisting of an A-module Ω and a K-derivation $d : A \to \Omega$ is called *universal* K-derivation of A, if for every K-derivation $\delta : A \to M$ there exists an A-linear map $\Phi : \Omega \to M$, such that $\delta = \Phi \circ d$. In this case Ω is called *differential module* of A; its elements being called *K-differentials* of A.

A differential module Ω of A can be constructed as the kernel of the map

$$\mu : A \otimes_K A \to A,\ \mu(a \otimes b) = a \cdot b\,,$$

i.e. we obtain $\Omega = \ker(\mu)$ as a submodule of $A \otimes_K A$ and the universal derivation is given by $\mathrm{d}a = (1 \otimes a) - (a \otimes 1)$. It follows directly from the definition that (Ω, d) is unique up to isomorphisms of A-modules. Thus, we may also write $\Omega_{A/K}$ for such Ω.

For an algebraic function field F/K in one variable, $\Omega_{F/K}$ is a one dimensional F-vector space. Let $x \in F$ such that $F/K(x)$ is a finite algebraic extension. Then $\Omega_{F/K}$ is generated by $\mathrm{d}x$, which can easily be seen from

$$\Omega_{F/K} = F \otimes_{K(x)} (K(x) \otimes_{K[x]} K[x]\mathrm{d}x) = F\mathrm{d}x\,,$$

see also [81, Theorem 6.5].

Let $P \in M_F$ be a place and t_P a local parameter at P. Then, by [81, Corollary 6.8], for every differential $\omega \in \Omega_{F/K}$ there exists a unique $f \in F$ such that $\omega = f\mathrm{d}t_P$. In this situation we define

$$v_P(\omega) := v_P(f)\,.$$

Now by [81, Proposition 6.14], we have that for any $\omega \in \Omega_{F/K} \setminus \{0\}$

$$\operatorname{div}(w) := \sum_{P \in M_F} v_P(\omega) P$$

is a well-defined divisor of F. So, for a divisor $D \in \operatorname{Div}(F)$, we can define

$$\Omega(D) = \{\, \omega \in \Omega_{F/K} \mid \operatorname{div}(\omega) \geq D \,\}.$$

Note that, for any differential ω there is an isomorphism

$$L(\operatorname{div}(\omega) - D) \to \Omega(D),\ x \mapsto x \cdot \omega\,.$$

In particular, $\Omega(D)$ is a finite-dimensional K-vector space. A differential $\omega \in \Omega_{F/K}$ is called *regular* or *holomorphic*, if $\operatorname{div}(\omega) \geq 0$ and we call

$$\Omega(0) = \{\, \omega \in \Omega_{F/K} \mid \operatorname{div}(\omega) \geq 0 \,\}.$$

the *space of holomorphic differentials*.

Definition 2.3.4. (Genus) We define the *genus* g of the algebraic function field F/K as the dimension

$$g = \dim_K \Omega(0)\,.$$

The *genus* of an algebraic curve C/K can now be defined as the genus of the associated algebraic function field $K(C)$.

Note that for any divisor coming from a differential $0 \neq \omega \in \Omega_{F/K}$ we have that $\deg(\operatorname{div}(\omega)) = 2g - 2$. Such divisors are called *canonical divisors* and they are all linearly equivalent. The famous Riemann-Roch theorem can now be stated as

Theorem 2.3.5 (Riemann-Roch)**.** *Let $D \in \operatorname{Div}(F)$. Then,*

$$\dim_K L(D) = \deg(D) - g + 1 + \dim_K \Omega(D)\,.$$

We want to stress that the space of holomorphic differentials can be computed as the Riemann-Roch space of the canonical divisor $\operatorname{div}(\mathrm{d}x)$, because

$$\Omega(0) \cong L(\operatorname{div}(\mathrm{d}x))\,.$$

2.3.3 Constant field extension

Later on, in Chapter 4 and for several applications in Chapter 6, it happens that we are given an algebraic curve C/K where K is a number field (mostly $K = \mathbb{Q}$). In practice, we want take advantage of this exactness by first executing certain computations over K and then transporting the data into $\mathbb{C}$ using a complex embedding $\iota : K \to \mathbb{C}$.

Note that the embedding ι induces a constant field extension

$$\mathbb{C}(C) \overset{\iota}{\cong} K(C) \otimes_K \mathbb{C}\,.$$

Every place P of $K(C)$ of degree $d = \deg(P)$ splits completely in the function field extension $\mathbb{C}(C)$, i.e. there are exactly places $P_1, \ldots, P_d$ lying over P. For every divisor $D = \sum v_P(D)P \in \operatorname{Div}(K(C))$ there exists a corresponding divisor $D' = \sum v_P(D) \sum_{P_i|P} P_i \in \operatorname{Div}(\mathbb{C}(C))$ such that $\deg(D) = \deg(D')$. Moreover, for every canonical divisor $D = \operatorname{div}(\omega)$ of $K(C)$, D' is a canonical divisor of $\mathbb{C}(C)$. Every basis of $L(D)$ as a K-vector space is also a basis of $L(D')$ as a $\mathbb{C}$-vector space. In particular, this is true for the space of holomorphic differentials

$$\Omega_{\mathbb{C}(C)}(0) \cong \Omega_{K(C)}(0) \otimes_K \mathbb{C}\,.$$

These statements can be proved by first moving from K to an algebraic closure $\overline{K}$. This is an algebraic constant field extension for which all the above statements are true by [80, Theorem 3.6.3]. The complex embedding ι extends to a complex embedding $\bar{\iota} : \overline{K} \to \mathbb{C}$ which can then be used to move from $\overline{K}$ to $\mathbb{C}$, obviously preserving all data as claimed.

2.4 Riemann surfaces

From here on we assume that the reader is familiar with the basics of topology, complex analysis and global analysis. As already mentioned in the beginning of this chapter, for this introduction to Riemann surfaces, our main sources are [10] and [63]; other sources are [33], [7] and [13].

2.4.1 Basic definitions

We follow [10, I.1] and [63, I] for the basic definitions.

Definition 2.4.1. A Riemann surface is defined as a complex manifold of complex dimension 1, which means that it is a topological space in which a neighborhood of any point looks like the complex plane. More precisely, a *Riemann surface* X is a (connected) Hausdorff space X such that for any point $P \in X$, we are given an open neighborhood U of P and a homeomorphism

$$\Phi : U \to \Phi(U) \subset \mathbb{C}$$

from U to an open domain in $\mathbb{C}$. We call the pair (U, Φ) *holomorphic chart* on X and the map Φ is called *local coordinate at* P; we say Φ is *centered at* $P \in U$ if $\Phi(P) = 0$. These homeomorphisms are required to be *compatible* in the following way: if the neighborhoods U_1 and U_2 of two charts (Φ_1, U_1) and (Φ_2, U_2) overlap, then

$$\Phi_2 \circ \Phi_1^{-1} : \Phi_1(U_1 \cap U_2) \to \Phi_2(U_1 \cap U_2)$$

is holomorphic. The function $\Phi_2 \circ \Phi_1^{-1}$ is called *transition function*. More generally, if Ψ is any holomorphic function defined in a neighborhood of $z = \Phi(P)$ and $\Psi'(z) \neq 0$, then $\Psi \circ \Phi$ defines the same complex structure on a neighborhood of P, as Φ is also a local coordinate at P. We say that X is a *compact Riemann surface* if it is compact as a topological space.

All Riemann surfaces in this thesis will be connected (except for those appearing in §2.4.4, but they are of minor importance).

As one can easily derive from their definition, Riemann surfaces may also be viewed as differentiable real manifolds. For compact orientable 2-manifolds there exists the following classification

Proposition 2.4.2. *[63, Proposition I.1.23] Every Riemann surface is an orientable path-connected 2-dimensional $\mathcal{C}^\infty$ real manifold. Every compact Riemann surface is diffeomorphic to the g-holed torus, for some unique integer $g \geq 0$.*

The non-negative integer g is a fundamental invariant of X and is called the *(topological) genus* of the compact Riemann surface g.

Obviously, any open domain in $\mathbb{C}$ is a Riemann surface. The most basic compact Riemann surface is the *projective line* or *Riemann sphere*

$$\mathbb{P}^1 := \mathbb{P}^1(\mathbb{C}) = \mathbb{C} \cup \{\infty\}.$$

The genus of $\mathbb{P}^1$ is 0 and, as a topological space, it is the one point compactification of $\mathbb{C}$. A complex structure on $\mathbb{P}^1$ is easily found: consider the open subsets $U_1 = \mathbb{P}^1 \setminus \{\infty\}$ and $U_2 = \mathbb{P}^1 \setminus \{0\}$ with corresponding local coordinates

$$\Phi_1 : U_1 \to \mathbb{C},\ z \mapsto z \quad \text{and} \quad \Phi_2 : U_2 \to \mathbb{C},\ z \mapsto \begin{cases} 1/z, & \text{if } z \neq \infty, \\ 0, & \text{if } z = \infty. \end{cases}$$

Functions and maps

The complex structure on a Riemann surface makes it possible to transfer the standard notions from the theory of functions in one complex variable from the complex plane to the Riemann surface:

- a complex valued function $f : U \to \mathbb{C}$ defined on an open subset $U \subset X$ is *holomorphic* if and only if for any $P \in U$ and any local coordinate Φ at P the function $f \circ \Phi^{-1}$ is holomorphic near $z = \Phi(P)$. Moreover, we say that f has a zero of order n at P if and only if $f \circ \Phi^{-1}$ has a zero of order n at z; we write $v_P(f) = n$.

- analogously one defines *meromorphic* functions on Riemann surfaces and the order of a pole of a meromorphic function; we write $v_P(f) = -n$, if f has a pole of order n at z;

- a continuous map $\varphi : X \to Y$ between two Riemann surfaces is *holomorphic* if and only if for any holomorphic chart $\Phi : U \to \mathbb{C}$ in Y, the map $\Phi \circ \varphi : \varphi^{-1}(U) \to \mathbb{C}$ is holomorphic. We call a holomorphic map between Riemann surfaces an *isomorphism* or *biholomorphic*, when it is a homeomorphism and its inverse is holomorphic.

Holomorphic maps to the projective line

Note that any meromorphic function $f : X \to \mathbb{C}$ on X defines a holomorphic map to the projective line $\mathbb{P}^1$ via

$$\varphi_f : X \to \mathbb{P}^1,\ P \mapsto \begin{cases} f(P), & \text{if } P \text{ is not a pole of } f, \\ \infty, & \text{if } P \text{ is a pole of } f. \end{cases} \tag{2.1}$$

Such maps play an important role in the classification of compact Riemann surfaces, as we will see later on.

2.4.2 Properties of holomorphic maps

In the following let X and Y be Riemann surfaces. The morphisms between these objects, the *holomorphic maps*, were already defined in §2.4.1 above. Following [63, II.4] we establish some global properties of holomorphic maps between Riemann surfaces.

First, we state that locally every holomorphic map is simply the power map (see [63, Lemma II.4.1]).

Proposition 2.4.3 (Local normal form)**.** *Let $\varphi : X \to Y$ be a non-constant holomorphic map that is defined at $P \in X$. Then there is a unique integer $m > 0$ with the following property: for every chart (Φ_2, U_2) on Y centered at $\varphi(P)$, there exists a chart (Φ_1, U_1) on X centered at P such that*

$$(\Phi_2 \circ \varphi \circ \Phi_1^{-1})(z) = z^m\,.$$

Definition 2.4.4. Let $\varphi : X \to Y$ be a non-constant holomorphic map. The *ramification index of φ at P*, denoted $e_P(\varphi)$, is the unique integer $m > 0$ such that there are local coordinates near P and $\varphi(P)$ with φ having the local normal form $z \mapsto z^m$.

A point $P \in X$ is a *ramification point* for φ if $e_P(\varphi) > 1$. A point $Q \in Y$ is a *branch point* for φ if it is the image of a ramification point for φ. We denote the set of branch points by $\mathcal{B} \subset Y$ and the set of ramification points by $\mathcal{R} \subset X$. Moreover, we say that the holomorphic map φ is *ramified* if there exists a ramification point, and *unramified* otherwise.

Let $\varphi : X \to Y$ be a non-constant holomorphic map between compact Riemann surfaces. By [63, Proposition II.4.8] we have that the sum of ramification indices of φ at the fiber above $Q \in Y$

$$d_Q(\varphi) = \sum_{P \in \varphi^{-1}(Q)} e_P(\varphi)$$

is independent of Q and thus constant. So, we may define the *degree of* φ as the integer

$$\deg(\varphi) := d_Q(\varphi)\,.$$

Using this one can prove (see [63, Proposition II.4.12]) that the sum of the orders of a non-constant meromorphic function f on a compact Riemann surface X is zero, i.e.

$$\sum_{P \in X} v_P(f) = 0\,.$$

Now we are in the position to state an important formula that relates the degree and the ramification indices associated to a holomorphic map with the genera of the compact Riemann surfaces, namely the

Theorem 2.4.5. (Riemann-Hurwitz formula) [63, Theorem II.4.16]
Let $\varphi : X \to Y$ be a non-constant holomorphic map of degree $\deg(\varphi) = m$ between compact Riemann surfaces of respective genera g_X and g_Y. Then

$$2g_X - 2 = m(2g_Y - 2) + \sum_{P \in X} (e_P(\varphi) - 1)\,. \tag{2.2}$$

In particular, in the case $Y = \mathbb{P}^1$ we have $g_Y = 0$ and the formula becomes

$$g_X = \frac{1}{2} \sum_{P \in X} (e_P(\varphi) - 1) - m + 1\,. \tag{2.3}$$

2.4.3 Monodromy & homotopy

For an introduction to the concepts of covering maps and their monodromy representations we follow [63, Section III.4].

Covering spaces & homotopy group

Let V be a connected real manifold. A *covering space* (or simply *covering*) of V is a continuous surjective map $\varphi : U \to V$ of connected real manifolds such that for each point $x \in V$ there is a neighborhood $W \subset V$ of x such that $\varphi^{-1}(W)$ consists of a disjoint union of open sets $U_\alpha \subset U$, each mapping homeomorphically onto W via φ.

The open sets U_α, which can be thought of as homeomorphic 'copies' of W in U, are called the *sheets* over W.

A *path* on V is a continuous map $\gamma : [-1, 1] \to V$. If V is a Riemann surface, we additionally require a path on V to be a piecewise $\mathcal{C}^\infty$ function. The points $\gamma(-1)$ and $\gamma(1)$ are called the *endpoints* of the path; we refer to $\gamma(-1)$ also as the *starting point*. The path γ is called *closed* whenever $\gamma(-1) = \gamma(1)$. Moreover, we denote by $-\gamma$ the *reverse path* defined by $-\gamma : [-1, 1] \to X,\ t \mapsto \gamma(-t)$.

Let γ_1 and γ_2 be two paths on X such that $\gamma_1(1) = \gamma_2(-1)$, then we call γ_1 and γ_2 *compatible* and define their *concatenation* as the path

$$\gamma_2 \cdot \gamma_1 : [-1, 1] \to X,\ t \mapsto \begin{cases} \gamma_1(2t + 1), & t \in [-1, 0], \\ \gamma_2(2t - 1), & t \in [0, 1]. \end{cases}$$

Further, two paths γ_0 and γ_1 on V are called *homotopic* if there exists a continuous function $H : [-1,1] \times [0,1] \to X$ (a *homotopy*) such that for each $s \in [0,1]$, $\gamma_s(t) = H(t,s)$ is a path on V, and such that $\gamma_s(\pm 1) = \gamma_0(\pm 1) = \gamma_1(\pm 1)$. In fact, this defines an equivalence relation and we write $\gamma_0 \sim \gamma_1$ for homotopic paths.

For a base point $x_0 \in V$, we define the *fundamental group* of V as

$$\pi_1(V, x_0) := \{ \text{ closed paths on } V \text{ based at } x_0 \} / \sim \ .$$

Path-lifting property A covering space $\varphi : U \to V$ enjoys the so-called *path-lifting property*: for any path $\gamma : [-1,1] \to V$ and any preimage $P \in \varphi^{-1}(\gamma(-1))$ there is a path $\tilde{\gamma}$ on U such that $\tilde{\gamma}(-1) = P$ and $\varphi \circ \tilde{\gamma} = \gamma$. This property is crucial to our approach of computing period matrices.

Monodromy of a finite covering

Let $\varphi : U \to V$ be a connected covering space of finite degree m, also called *m-sheeted covering* (i.e. all points of V have exactly m preimages under φ), and let $x_0 \in V$ be a base point. The *fiber above* x_0 is the set of preimages

$$\varphi^{-1}(x_0) = \{ P_1, \ldots, P_m \} .$$

By the path-lifting property, every closed path based at x_0 can be lifted to exactly m different paths $\tilde{\gamma}_1, \ldots, \tilde{\gamma}_m$, where $\tilde{\gamma}_i$ is the unique lift starting at P_i, i.e. $\tilde{\gamma}_i(-1) = P_i$ for $i = 1, \ldots, m$. The endpoints $\tilde{\gamma}_i(1)$ also lie above x_0 and form the entire preimage, i.e.

$$\varphi^{-1}(x_0) = \{ \tilde{\gamma}_i(-1) \mid i = 1, \ldots, m \} = \{ \tilde{\gamma}_i(1) \mid i = 1, \ldots, m \} .$$

Consequently, there exists a permutation $\sigma_\gamma \in \mathrm{Sym(m)}$ such that $\sigma(i) = j$ precisely when $\tilde{\gamma}_i(-1) = \tilde{\gamma}_j(1)$ and σ_γ solely depends on the homotopy class of γ. This induces a group homomorphism

$$\mathrm{Mon}(\varphi) : \pi_1(V, x_0) \to \mathrm{Sym(m)}, \ [\gamma] \mapsto \sigma_\gamma \tag{2.4}$$

We call $\mathrm{Mon}(\varphi)$ the *monodromy representation* of the covering map $\varphi : U \to V$. The monodromy representation depends on the choice of the base point $x_0 \in V$ and the ordering of the sheets above x_0, but only up to simultaneous conjugation. Therefore, the conjugacy class of its image in Sym(m) is uniquely determined.

Monodromy of a holomorphic map

Let now $\varphi : X \to Y$ be a holomorphic map between compact Riemann surfaces as introduced in Section 2.4.1. Generally we can expect φ to be ramified, so it will not be a 'true' (unramified) covering map. In this case we call the holomorphic map φ a *ramified covering*. However, we can define open sets $V = Y \setminus \mathcal{B}$ and $U = X \setminus \varphi^{-1}(\mathcal{B})$ by removing the branch points of φ from Y and the preimages of the branch points from X (this includes the ramification points $\mathcal{R}$). Then, V and U will still be Riemann surfaces, which we will call *Riemann surface with punctures*, see Definition 2.5.1.

Now, for any $v \in V$ the preimage $\varphi^{-1}(v)$ consists of m distinct points. Therefore, the restriction $\varphi_{|U} : U \to V$ is a covering map of degree m. This covering map now has a monodromy representation $\mathrm{Mon}(\varphi) : \pi_1(V, x_0) \to \mathrm{Sym(m)}$ which is called the *monodromy representation* of the holomorphic map φ. Since X is connected, so is the open subset U and, by [63, Lemma III.4.4], this implies that the image of $\pi_1(V, x_0)$ under $\mathrm{Mon}(\varphi)$ must be a transitive subgroup of Sym(m).

Suppose that above the branch point $x \in \mathcal{B} \subset Y$ there are k preimages $P_1, \ldots, P_k$ with $e_{P_j}(\varphi) = e_j$. Let $\gamma \in \pi_1(V, x_0)$ be a path *encircling* x, i.e. $\gamma = (-\alpha)\beta\alpha$ where α is a

path joining x_0 and $\tilde{x} \in W \setminus \{x\}$ where W is a small neighborhood of x, β a small circle around x that is based at $\tilde{x}$ and contained in W, and $-\alpha$ the reverse path of α. Then, the permutation σ_γ has cycle structure $(e_1, \ldots, e_k)$ (see [63, Lemma III.4.6]); we call $\sigma_x := \sigma_\gamma$ the *local monodromy* at $x \in \mathcal{B}$.

2.4.4 Two constructions of Riemann surfaces

Here, we consider two ways of constructing Riemann surfaces that are important in this work. For this we closely follow [10, Section I.2].

Algebraic curves The Riemann surfaces occurring in this thesis are precisely the Riemann surfaces associated to *algebraic curves* defined over the complex numbers, as already introduced in Section 2.2. The easiest examples of an complex algebraic curves are given by irreducible non-constant polynomials $f \in \mathbb{C}[x, y]$.

Denote by $C_f : f = 0$ the *affine plane curve*

$$C_f := C_f(\mathbb{C}) = \{ (x, y) \in \mathbb{C}^2 \mid f(x, y) = 0 \} \tag{2.5}$$

and by S_f its *singular locus*, defined by

$$S_f := \{ P \in C_f(\mathbb{C}) : \partial_x f(P) = \partial_y f(P) = 0 \} .$$

Then, the non-singular part of the affine curve

$$\tilde{C}_f := C_f \setminus S_f \tag{2.6}$$

is easily checked to have the structure of a one dimensional complex submanifold of $\mathbb{C}^2$, i.e. a Riemann surface. The connectedness of $\tilde{C}_f$ is a consequence of f being irreducible.

Holomorphic charts on C_f are obtained from the implicit function theorem [63, II,Theorem 2.1]. Let $P \in C_f$ be a point such that $\partial_y f(P) \neq 0$. Then there is an open neighborhood U of P such that the projection $\varphi_{\mathrm{x}} : U \to \mathbb{C}$, $(x, y) \mapsto x$ is a homeomorphism from U to its image, giving us a local coordinate at P. Analogously, if $\partial_x f(P) \neq 0$ then we can take the projection $\varphi_{\mathrm{y}} : U \to \mathbb{C}$, $(x, y) \mapsto y$ as local coordinate.

Example 2.4.6. Consider a polynomial $f := y^m - p(x)$ where $m > 1$ and $p \in \mathbb{C}[x]$ has no multiple roots. Then, f is irreducible, $S_f = \varnothing$ and C_f is a Riemann surface. We call algebraic curves defined by such polynomials *superelliptic curves*; they will be the topic of Chapter 5.

Analytic continuation The second construction of Riemann surfaces uses the well-known concept of *analytic continuation*. Let $x_0 \in \mathbb{C}$ be a point and y_0 be a germ of holomorphic functions at x_0, i.e. a series

$$y_0(z) = \sum_{i=0}^{\infty} a_i (z - x_0)^i, \quad a_i \in \mathbb{C},$$

with a positive radius of convergence. The Riemann surface of y_0 is the largest connected Riemann surface, unramified over $\mathbb{C}$, to which the germ y_0 may be extended as a holomorphic function.

We now describe a formal construction of this Riemann surface: consider the set $\mathcal{O}$ consisting of pairs (x, y) where $x \in \mathbb{C}$ and y is a germ of holomorphic functions at x, and consider the projection onto the x-coordinate

$$\varphi_x : \mathcal{O} \to \mathbb{C}, \ (x, y) \mapsto x .$$

There exists a unique structure of a (highly non-connected) Riemann surface on $\mathcal{O}$ which satisfies the following conditions: for any $(x, y) \in \mathcal{O}$, if the radius of convergence of y is $r > 0$ and if, for any $\tilde{x} \in D(x; r) := \{ z \in \mathbb{C} \mid |x - z| < r \}$ (the open disc of radius r around x), we denote by $y_{\tilde{x}}$ the Taylor series of y at $\tilde{x}$, the map

$$D(x; r) \to \mathcal{O},\ \tilde{x} \mapsto (\tilde{x}, y_{\tilde{x}})$$

is a biholomorphic map from $D(x; r)$ onto its image. The Riemann surface structure on $\mathcal{O}$ makes the map φ_{x} a holomorphic map, that is locally biholomorphic (i.e. unramified). Moreover, the map

$$\varphi_{\mathrm{y}} : \mathcal{O} \to \mathbb{C},\ (x, y) \mapsto y(x)$$

is easily seen to be holomorphic.

The Riemann surface $X \subset \mathcal{O}$ of y_0 is now defined as the connected component of (x_0, y_0) in $\mathcal{O}$. By construction, X is a covering space of the complex plane via the restriction

$$\varphi_x : X \to \mathbb{C}$$

and therefore, a Riemann surface. Moreover, the holomorphic function φ_{y} extends the germ y, which can be seen as a germ of holomorphic functions on a neighborhood of (x, y) in X (identified with a neighborhood of x in $\mathbb{C}$ by φ_{x}).

The link between this construction of the analytic continuation and its more classical description is made by the following observation: a germ $(\tilde{x}, \tilde{y})$ belongs to X if and only if there exists a finite sequence (x_i, y_i), $0 \leq i \leq N$, of germs such that $(x_0, y_0) = (x, y)$, $(x_N, y_N) = (\tilde{x}, \tilde{y})$ and such that the open discs of convergence of y_{i-1} and y_i intersect and y_i and y_{i-1} coincide on this intersection.

The most important example of analytic continuation for us is provided by algebraic functions. Assume y is a germ of an algebraic function, i.e. that there exists a non-zero polynomial $f \in \mathbb{C}[x, y]$ such that

$$f(\tilde{x}, y(\tilde{x})) = 0$$

for any $\tilde{x}$ in a neighborhood of x. The polynomial f may taken to be irreducible and then the map

$$X \to \mathbb{C}^2,\ P = (x, y) \mapsto (\varphi_{\mathrm{x}}(P), \varphi_{\mathrm{y}}(P)) = (x, y(x))$$

establishes an isomorphism between X and the open subset

$$\{ P \in C_f \mid \partial_y f(P) \neq 0 \} \subset C_f$$

of the affine plane curve $C_f : f = 0$. In fact, we rely on this construction in Section 4.5, in order to lift smooth paths from the complex plane to Riemann surfaces given by affine plane curves.

2.5 Compact Riemann surfaces and algebraic curves

In this section we construct compact Riemann surfaces from algebraic curves. For this we are first going to introduce the notion of a Riemann surface with punctures.

Definition 2.5.1. Let X be a Riemann surface. We say that X is a *Riemann surface with punctures* if there exists an open subset $U \subset X$ such that

(C1) there exists a biholomorphic map from U onto a disjoint finite union of punctured discs $\{ 0 < |z| < 1 \}$;

(C2) $X \setminus U$ is compact.

For such X one easily obtains a new Riemann surface $\hat{X}$ by gluing X and a disjoint finite union of unpunctured discs $\{\,|z| < 1\,\}$. The resulting $\hat{X}$ is compact, contains X as a subset, and the complement $\hat{X} \setminus X$ is finite. Moreover, $\hat{X}$ is well-defined as it is characterized by these properties.

2.5.1 Coverings of the projective line

Let $\mathcal{L}$ be a finite subset of $\mathbb{P}^1$ and $\varphi : X \to \mathbb{P}^1 \setminus \mathcal{L}$ be a covering map of finite degree. Then X is a compact Riemann surface with punctures. If we choose for every $x \in \mathcal{L}$ a neighborhood $U_x \subset \mathbb{P}^1$ that is disjoint from $\mathcal{L} \setminus \{x\}$ and such that the U_x's are pairwise disjoint and biholomorphic to open discs, then the open subset

$$U = \bigcup_{x \in \mathcal{L}} \varphi^{-1}(U_x \setminus \{x\}) \subset X$$

clearly satisfies the condition (C2).

Since any covering of finite degree of a punctured disc is biholomorphic to a disjoint union of punctured discs, the set U satisfies (C1) as well. As a consequence, one obtains a compact Riemann surface $\hat{X}$ and the covering $\varphi : X \to \mathbb{P}^1 \setminus \mathcal{L}$ extends to a holomorphic map between compact Riemann surfaces

$$\hat{\varphi} : \hat{X} \to \mathbb{P}^1$$

whose ramification points are a subset of $\varphi^{-1}(\mathcal{L})$, i.e. $\hat{\varphi}$ is a ramified covering.

Remark 2.5.2. Let $S \subset X$ be a finite subset. Note that

$$X \text{ is compact R.S. with punctures} \quad \Leftrightarrow \quad X \setminus S \text{ is a R.S. with punctures.}$$

In that case, $\hat{X}$ may be identified with $\hat{Y}$ where $Y = X \setminus S$. More generally, let $\varphi : X \to Y$ be a non-constant holomorphic map between Riemann surfaces. If Y is a compact Riemann surface with punctures, then so is X.

Monodromy representation The fundamental group of $V = \mathbb{P}^1 \setminus \mathcal{L}$ with base point x_0 is a free group $\pi_1(V, x_0)$ on n generators $[\gamma_1], \ldots, [\gamma_n]$ subject to the single relation

$$[\gamma_1] \cdot \ldots \cdot [\gamma_n] = 1\,.$$

Each $[\gamma_i]$ is the homotopy class of a closed path based at x_0 encircling x_i exactly once. Therefore, we obtain a monodromy representation

$$\mathrm{Mon}(\varphi) : \pi_1(V, x_0) \to \mathrm{Sym(m)}$$

by choosing the n permutations $\sigma_i := \sigma_{\gamma_i}$ such that

$$\sigma_1 \cdot \ldots \cdot \sigma_n = \mathrm{id}\,.$$

By [63, Corollary III.4.10] we have a one-to-one correspondence between

- isomorphism classes of holomorphic maps $\hat{\varphi} : \hat{X} \to \mathbb{P}^1$ of degree m whose branch points lie in $\mathcal{L}$ (as constructed above);
- conjugacy classes of n-tuples $(\sigma_1, \ldots, \sigma_n)$ of permutations in Sym(m) such that

  $$\sigma_1 \cdot \ldots \cdot \sigma_n = \mathrm{id}\,,$$

  and the subgroup generated by the σ_i's is transitive.

In the following, if generators $\gamma_1, \ldots, \gamma_n$ are explicitly given, we define the monodromy representation of a holomorphic map $\varphi : X \to \mathbb{P}^1$ simply as the set of images of these generators, i.e.

$$\mathrm{Mon}(\varphi) := \{\, \sigma_i \neq \mathrm{id} \mid i = 1, \ldots, n \,\}\,.$$

2.5.2 Constructing the fundamental group

Let X be a compact Riemann surface, $\varphi : X \to \mathbb{P}^1$ be an m-sheeted ramified covering and $\mathcal{L}$ be a finite set containing the branch points of φ, i.e. $\mathcal{B} \subset \mathcal{L} \subset \mathbb{P}^1$. Suppose we computed the monodromy representation $\mathrm{Mon}(\varphi) = \{\, \sigma_i \neq \mathrm{id} \mid i = 1, \ldots, n \,\}$ with respect to a base point $x_0 \notin \mathcal{L}$. Denote by S the free group generated by $\mathrm{Mon}(\varphi)$. We can then obtain a generating set for the fundamental group $\pi_1(X \setminus \varphi^{-1}(\mathcal{L}), P_0)$ of the punctured Riemann surface, where P_0 lies over x_0 (corresponding to the index 1), as the stabilizer subgroup $\mathrm{Stab}(S, 1)$.

In order to obtain the fundamental group $\pi_1(X, P_0)$ we have to factor out the *holes*, i.e. closed paths on $X \setminus \varphi^{-1}(\mathcal{L})$ based at P_0 that become trivial on the unpunctured Riemann surface X. The cycle structure $(e_1, \ldots, e_k)$ of $\sigma_i = \tau_1 \cdot \ldots \cdot \tau_k$ corresponds to the ramification indices of φ above x_i. Since S is transitive as subgroup of $\mathrm{Sym(m)}$, for every cycle τ_j of order e_j we can find $\alpha \in \mathrm{Sym(m)}$ such that $\alpha \tau_j^{e_j} (-\alpha) \in \mathrm{Stab}(S, 1)$.

Factoring out the subgroup generated by such elements will yield the fundamental group $\pi_1(X, P_0)$, as they correspond to cycles that will become trivial. This process is summarized in the diagram below.

$$\begin{array}{ccccc} \pi_1(\mathbb{P}^1 \setminus \mathcal{L}, x_0) & \dashrightarrow & \pi_1(X \setminus {\varphi_{\mathrm{x}}}^{-1}(\mathcal{L}), P_0) & \dashrightarrow & \pi_1(X, P_0) \\ \updownarrow & & \updownarrow & & \updownarrow \\ S = \langle \sigma_1, \ldots, \sigma_n \rangle & \xrightarrow{\text{subgroup}} & \mathrm{Stab}(S, 1) & \xrightarrow{\text{factor group}} & \mathrm{Stab}(S, 1)/\langle \text{holes} \rangle \end{array}$$

This construction is extremely useful: once we know a generating set for $\pi_1(X, P_0)$, the homology group is easily obtained as the abelianization. In particular, everything can be computed using permutations, i.e. in terms of a monodromy representation.

It is used in the Tretkoff algorithm which computes a homology basis of X from a monodromy representation $\mathrm{Mon}(\varphi)$ (see §4.6.1 or Section A.2). Moreover, in the case of superelliptic curves this construction yields the proof of Theorem 5.1.6, which provides an explicit generating set for $\pi_1(X, P_0)$.

2.5.3 Algebraic curves and their normalizations

Let $f \in \mathbb{C}[x, y]$ be an irreducible polynomial with $m = \deg_y(f) > 0$ and let $\tilde{C}_f$ be the non-singular part of the affine curve $C_f : f = 0$. Define the finite subset

$$\mathcal{L} = \{\, z \in \mathbb{C} \mid f(z, y) \text{ has degree } < m \text{ or has a multiple root} \,\} \subset \mathbb{C}\,.$$

The set $\mathcal{L}$, which we will later (see §4.1.2) call *exceptional values*, may be obtained as the set of roots of the discriminant (and the leading coefficients) of f as a polynomial in y, denoted $\mathrm{disc}_y(f) \in \mathbb{C}[x]$. Note that our assumptions on f guarantee that the discriminant $\mathrm{disc}_y(f)$ does not vanish identically. Moreover, we define

$$X := \{\, (x, y) \in \tilde{C}_f \mid x \notin \mathcal{L} \,\}\,,$$

$\hat{\mathcal{L}} := \mathcal{L} \cup \{\infty\}$, and consider the projection onto the x-coordinate

$$\varphi_{\mathrm{x}} : X \to \mathbb{C} \setminus \mathcal{L} = \mathbb{P}^1 \setminus \hat{\mathcal{L}},\ (x, y) \mapsto x\,.$$

Then φ_{x} is an unramified covering of degree m, and we are in the situation of §2.5.1. Therefore, X is a compact Riemann surface with punctures and we obtain a compact Riemann surface $\hat{X}$ and a holomorphic map $\hat{\varphi_{\mathrm{x}}}$ which extends φ_{x} by the following argumentation:

As $X \subset \tilde{C}_f$ and $X \setminus \tilde{C}_f$ is finite (it is included in $(\mathcal{L} \times \mathbb{C}) \cap C_f$ which is finite since f is irreducible and belongs to $\mathbb{C}[x,y] \setminus \mathbb{C}[x]$) we see that $\tilde{C}_f$ itself is a compact Riemann surface with punctures, and that the map

$$\varphi_{\mathrm{x}} : \tilde{C}_f \to \mathbb{C}, \quad (x,y) \mapsto x$$

extends to a holomorphic map

$$\hat{\varphi_{\mathrm{x}}} : \hat{C}_f = \hat{X} \to \mathbb{P}^1 .$$

We say that the compact Riemann surface $\hat{C}_f$ is *associated* to the plane affine curve $C_f : f = 0$. Connectedness of $\hat{C}_f$ results from f being irreducible. In fact, all compact Riemann surfaces can be constructed this way.

Theorem 2.5.3. *[10, Theorem I.4.2.] Let X be a compact Riemann surface.*

(1) There exists a non-constant holomorphic map

$$\varphi : X \to \mathbb{P}^1 .$$

(i.e. a non-constant meromorphic function on X).

(2) For any map φ from (1), there exists an irreducible polynomial $f \in \mathbb{C}[x,y]$ and an isomorphism $\psi : X \xrightarrow{\cong} \hat{C}_f$ such that

$$\varphi = \hat{\varphi_{\mathrm{x}}} \circ \psi .$$

The important part of Theorem 2.5.3 is really the existence of a non-constant meromorphic function on X. This fact is highly non-trivial and it is a property that is special to one dimensional compact complex manifolds.

Let us now consider general algebraic curves defined over the complex numbers, as we introduced them in Section 2.2, namely as algebraic varieties of dimension one. For the sake of simplicity, we denote

$$\mathbb{A}^n := \mathbb{A}^n(\mathbb{C}) \quad \text{and} \quad \mathbb{P}^n := \mathbb{P}^n(\mathbb{C}) .$$

for the rest of this thesis. Let $C/\mathbb{C}$ be an affine (resp. projective) algebraic curve, viewed as an irreducible algebraic subset of $\mathbb{A}^n$ (resp. $\mathbb{P}^n$). Recall that a point $P \in C$ is called *non-singular* or *smooth* if the local ring $\mathcal{O}_P$ of regular functions at P is a discrete valuation ring; we denote the set of non-singular points by $C_{\mathrm{reg}} \subset C$.

For complex algebraic curves smooth points can be characterized in the following way: $P \in C$ is smooth, i.e. $P \in C_{\mathrm{reg}}$, if there exists an open neighborhood $U \subset \mathbb{A}^n$ (resp. $\mathbb{P}^n$) such that $U \cap C$ is a complex submanifold of C. In particular, C_{reg} is a connected set of complex dimension 1 and the set of singular points $C \setminus C_{\mathrm{reg}}$ is finite.

Proposition 2.5.4. *Let C be any affine (or projective) algebraic curve. Then C_{reg} is a compact Riemann surfaces with punctures.*

The proof is a generalization of the proof for $\tilde{C}_f$. One shows that the linear projection from C_{reg} to a 'generic' line, restricted to the complement of a finite set of ramification points, is a proper finite unramified covering.

Therefore, we may consider the compact Riemann surface $\hat{C}_{\mathrm{reg}}$. In the projective case, the identity map from C_{reg} to itself extends to a holomorphic map

$$\Phi : \hat{C}_{\mathrm{reg}} \to C \subset \mathbb{P}^n .$$

The non-singular projective curve $\hat{C}_{\text{reg}}$ is called the *normalization* of C and can be obtained from C by *resolving its singularities*. The process of resolving singularities is described in detail in [13, Chapter III]. Several methods for desingularizing are known, a prominent one uses Puiseux series expansions, this approach is discussed in Section 4.11, see also [13, Section 8.3].

The pair $(\hat{C}_{\text{reg}}, \Phi)$ is exactly the *non-singular model* of the projective curve V that already occurred in Definition 2.2.3. In the affine case, say $C \subset \mathbb{A}^n$, the *projective closure* $\overline{C}$ of C in $\mathbb{P}^n$ is a projective algebraic curve and $\hat{C}_{\text{reg}}$ is the normalization of $\overline{C}$.

Hence, any complex algebraic curve C is uniquely *associated* (up to isomorphism) to a compact Riemann surface $\hat{C}_{\text{reg}}$.

Plane algebraic curves Let $f \in \mathbb{C}[x, y]$ an irreducible polynomial of degree $d > 0$. Then the affine plane curve $C_f : f = 0$ is an affine algebraic curve in $\mathbb{C}^2$. We can obtain a plane projective curve $C_F : F = 0$ as the zero locus of the *homogenization* of f

$$F(x, y, z) = z^d f(x/z, y/z) \in \mathbb{C}[x, y, z] \,.$$

The plane projective algebraic curve

$$\overline{C}_f = \{\, [x : y : z] \in \mathbb{P}^2 \mid F(x, y, z) = 0 \,\} \subset \mathbb{P}^2$$

is called the *projective closure* of C_f. Its normalization coincides with $\hat{C}_f$.

Theorem 2.5.5 (Genus formula)**.** *Let C be a plane projective algebraic curve of degree d with singular points $P_1, \ldots, P_k \in C$. Then the genus of the associated Riemann surface X is given by*

$$g = \frac{(d-1)(d-2)}{2} - \frac{1}{2} \sum_{i=1}^{k} v_i(v_i - 1)$$

where v_i is the multiplicity [13, p.598] corresponding to P_i.

Proof. See [13, Theorem 9.2.5] □

We already established that every algebraic curve is associated to a compact Riemann surface. As we shall see now, every compact Riemann surface is also a smooth projective curve.

The preceding theorem shows that, generally, compact Riemann surfaces of genus $g \geq 2$ cannot be realized as smooth projective curves in $\mathbb{P}^2$. It is easy to see that for any integer $g \geq 0$ there is a Riemann surface with genus g. For example any separable polynomial $p(x) \in \mathbb{C}[x]$ of degree $d \geq 3$ defines a Riemann surface (a hyperelliptic curve) of genus $g = \lfloor \frac{d+1}{2} \rfloor - 1$ via the affine equation $y^2 = p(x)$. However, the genus of a smooth plane projective curve of degree d is $g = (d-1)(d-2)/2$ and clearly not all integers are of this form. Therefore, in order to realize all compact Riemann surfaces as smooth projective curves, one has to look at curves in higher dimensional projective spaces.

Theorem 2.5.6. *[10, Theorem I.4.3] For any compact Riemann surface X, there exists a holomorphic embedding*

$$\varphi : X \to \mathbb{P}^3 \,.$$

In particular, the image $\varphi(X) \subset \mathbb{P}^3$ is a smooth algebraic curve.

For more details on how Riemann surfaces can be embedded into a projective space we refer to [63, Section V.4]. In fact, this theorem is simply a special case of the following result by Chow (see [10, Theorem I.4.4]).

Theorem 2.5.7. *Any compact connected complex submanifold of $\mathbb{P}^n$ is an algebraic subvariety of $\mathbb{P}^n$ of equal dimension.*

From this theorem it follows that the meromorphic functions on algebraic curves are precisely the rational functions.

Theorem 2.5.8. *[10, Theorem I.4.5]*

(1) Let $f \in \mathbb{C}[x, y]$ be irreducible. For any meromorphic function G on $\hat{C}_f$ there exists a rational function $H \in \mathbb{C}(x, y)$ and a finite subset $\mathcal{L} \subset \tilde{C}_f$ such that H is defined and coincides with G on $\tilde{C}_f \setminus \mathcal{L}$.

(2) Let X be a compact Riemann surface embedded in $\mathbb{P}^n$. For any meromorphic function F on X, there exist homogeneous polynomials $G, H \in \mathbb{C}[x_0, \ldots, x_n]$ of equal degree and a finite subset $\mathcal{L} \subset X$ such that for any $x = [x_0 : \cdots : x_n] \in X \setminus \mathcal{L}$,

$$H(x) \neq 0 \quad \text{and} \quad F(x) = \frac{G(x)}{H(x)}\,.$$

Category equivalence From the statements presented in Section 2.5 we can now conclude that there is in fact a one-to-one correspondence between compact Riemann surfaces and complex smooth projective curves:

As shown by Theorem 2.5.6, every compact Riemann surface can be holomorphically embedded into projective space such that its image is a smooth projective curve that is, of course, unique up to isomorphism. Conversely, in §2.5.3 we have seen that the non-singular projective model of a complex algebraic curve, which is unique up to isomorphism, does indeed define a compact Riemann surface.

In particular, in the case $K = \mathbb{C}$, the equivalence of §2.2.4 extends to a triple category equivalence between

- compact Riemann surface with non-constant holomorphic maps,
- smooth projective curves $C/\mathbb{C}$ with non-constant regular maps,
- algebraic function fields in one variable over $\mathbb{C}$ with homomorphisms of $\mathbb{C}$-algebras.

Studying and computing with Riemann surfaces, these equivalences allow us to combine and apply methods of algebraic geometry, complex analysis and field theory interactively. A more general notion of the connections between algebraic geometry and analytic geometry is developed in Serre's GAGA paper [79].

2.6 Integration on Riemann surfaces

Finally, we look at integration on Riemann surfaces which is an important part of this thesis. The objects that we integrate are called *differential forms* (in particular, 1-*forms*) and will be introduced in §2.6.1 below. Differential 1-forms are integrated along *paths* on Riemann surfaces, as introduced in §2.4.3. In this section we follow Chapter IV of [63].

2.6.1 Differential forms

Definition 2.6.1 (1-forms)**.** A *holomorphic (meromorphic)* 1-*form* on an open set $V \subset \mathbb{C}$ is an expression of the form

$$\omega = f(z)\mathrm{d}z$$

where f is a holomorphic (meromorphic) function on V. We say that ω is a holomorphic (meromorphic) 1-form in the coordinate z. A *holomorphic (meromorphic)* 1-*form* on a

Riemann surface X is a collection of holomorphic (meromorphic) 1-forms $\{\omega_\Phi\}$, one for each holomorphic chart (Φ, U), such that for overlapping neighborhoods U_1 and U_2 of two charts (Φ_1, U_1) and (Φ_2, U_2) the associated 1-form ω_{Φ_1} transforms to ω_{Φ_2} under the transition function $T = \Phi_2 \circ \Phi_1^{-1}$, i.e. if $\omega_1 = f(z)\mathrm{d}z$ on U_1 and $\omega_2 = g(u)\mathrm{d}u$ on U_2 we have

$$g(u) = f(T(u))T'(u)\,.$$

Let ω be a meromorphic 1-form defined in a neighborhood of a point P. Choosing a local coordinate centered at P, we can write $\omega = f(z)\mathrm{d}z$ where f is meromorphic at $z = 0$. The *order of ω at P*, denoted by $v_P(\omega)$, is defined as the order of f at 0.

It is easy to see that $v_P(\omega)$ is independent of the choice of the local coordinate and thus well-defined. A meromorphic 1-form ω is holomorphic at P if and only if $v_P(\omega) \geq 0$. We say that P is a *zero of ω of order n*, if $v_P(\omega) = n > 0$. Conversely, P is a *pole of ω of order n*, if $v_P(\omega) = -n < 0$. The set of zeros and poles of a meromorphic 1-form is a discrete set.

Defining meromorphic functions and 1-forms The definition of a meromorphic or holomorphic 1-form ω suggests that in order to define ω on a Riemann surface X, one must give local expressions for ω of the form $f(z)\mathrm{d}z$ in each holomorphic chart of X. In fact, one can define ω by a single formula on a single chart. This is sufficient to determine ω by the Identity Theorem for meromorphic functions and forms: if two meromorphic 1-forms agree on an open set, they must be identical.

Let $U \subset X$ be an open subset. We define the following complex vector spaces.

$$\begin{aligned}
\mathcal{O}(U) &= \{\text{ holomorphic functions defined on } U\ \},\\
\Omega^1(U) &= \{\text{ holomorphic 1-forms defined on } U\ \},\\
\mathcal{M}(U) &= \{\text{ meromorphic functions defined on } U\ \},\\
\mathcal{M}^1(U) &= \{\text{ meromorphic 1-forms defined on } U\ \}.
\end{aligned}$$

If the subset U is connected then $\mathcal{O}(U)$ is an integral domain and $\mathcal{M}(U)$ is a field. The spaces $\Omega^1(U)$ and $\mathcal{M}^1(U)$ are modules over $\mathcal{O}(U)$ an if U is connected then $\mathcal{M}^1(U)$ is a vector space over $\mathcal{M}(U)$. In particular, $\mathcal{M}(X)$ is called the *field of meromorphic functions* on X. Note that we can now describe the space of holomorphic 1-forms on X via

$$\Omega^1(X) = \{\, \omega \in \mathcal{M}^1(X) \mid v_P(\omega) \geq 0 \text{ for all } P \in X \,\}\,.$$

Plane algebraic curves Let F be an irreducible homogeneous polynomial of degree $d \geq 3$ in $\mathbb{C}[x_0, x_1, x_2]$ such that the algebraic curve in $C_F : F = 0$ in $\mathbb{P}^2$ is smooth and let $X = C_F(\mathbb{C})$ be the associated compact Riemann surface. The coordinates x_0, x_1, x_2 define meromorphic functions on X

$$x = \frac{x_0}{x_2} \quad \text{and} \quad y = \frac{x_1}{x_2}\,.$$

Then for any polynomial $b \in \mathbb{C}[x, y]$ of degree $\leq d - 3$, the meromorphic 1-form

$$\alpha(b) = \frac{b(x,y)}{\partial_y F(x,y,1)}\mathrm{d}x = -\frac{b(x,y)}{\partial_x F(x,y,1)}\mathrm{d}y$$

is holomorphic. Moreover, the map $b \mapsto \alpha(b)$ is an isomorphism between the space of polynomials of degree $\leq d - 3$ in $\mathbb{C}[x, y]$ and $\Omega^1(X)$.

If X is obtained as the normalization of an algebraic curve, it is always possible to describe $\Omega^1(X)$ in terms of the algebraic data defining this curve, as is shown by the following theorem. We refer to [13, p. 598] for the definition of an *adjoint curve* and its *order*.

Theorem 2.6.2. *Let $C_f : f = 0$ be the affine plane curve defined by an irreducible polynomial $f \in \mathbb{C}[x, y]$ with $m = \deg_y f > 0$. Let X be the compact Riemann surface associated to C_f as explained in §2.5.3. Then the holomorphic differentials on X are the differential forms*

$$\frac{b(x, y)}{\partial_y f(x, y)} \mathrm{d}x \,,$$

where $b(x, y) = 0$ is the equation of a curve of order $d - 3$ adjoint to C_f . Moreover, the vector space of holomorphic differentials $\Omega^1(X)$ has dimension g where g is the (topological) genus of X.

Proof. See [13, Theorem 9.3.1]. □

Remark 2.6.3. Note that Serre's famous GAGA paper [79] shows, that over the complex numbers (i.e. $K = \mathbb{C}$), the (algebraic) differentials of algebraic function fields (or, equivalently, of non-singular projective algebraic curves) as defined in §2.3.2 coincide with the (analytic) differential forms on Riemann surfaces as defined in this section, i.e.

$$\Omega^1(X) \cong \Omega_{\mathbb{C}(C_f)}(0) \,.$$

Corollary 2.6.4 (Equality of topological and algebraic genus)**.** *The (algebraic) genus of a smooth projective curve defined over $\mathbb{C}$, see Definition 2.3.4, equals the (topological) genus of the associated compact Riemann surface.*

2.6.2 Integration of 1-forms along paths

First note that by [63, Lemma IV.3.7] every path $\gamma : [-1, 1] \to X$ may be partitioned into a finite number of paths $\{\gamma_i\}$, such that each γ_i is $\mathcal{C}^\infty$, with image contained in a single chart domain of X, i.e. $\gamma_i([-1, 1]) \subset U_i$ for some holomorphic chart (Φ_i, U_i).

Let ω be a meromorphic 1-form on a Riemann surface X and let γ be a path on X with a partition $\{\gamma_i\}$ with holomorphic charts (Φ_i, U_i) as above. With respect to each chart Φ_i, write the 1-form as $\omega = f_i(z_i)\mathrm{d}z_i$ and consider the path $\Phi_i \circ \gamma_i : [-1, 1] \to \mathbb{C}$ to be the defining function for $z_i = z_i(t)$ for $t \in [-1, 1]$.

Definition 2.6.5 (Integration of 1-forms)**.** With the above notation, we define the *integral of ω along γ* to be the complex number

$$\int_\gamma \omega = \sum_i \int_{-1}^{1} f_i(z_i(t)) z_i'(t) \mathrm{d}t \,. \tag{2.7}$$

In particular, if the image γ is contained in the domain of a single chart $\Phi : U \to V$ and if $\omega = f\mathrm{d}z$ in this chart, then

$$\int_\gamma \omega = \int_{\Phi \circ \gamma} f \mathrm{d}z \,.$$

where the integral on the right is the usual contour integral of the path $\Phi \circ \gamma$ in $V \subset \mathbb{C}$. By [63, Lemma IV.3.9] the integral (2.7) has all the properties of the usual integral: it is $\mathbb{C}$-linear; independent of the choice of parametrization of γ; it is linear under partition; the fundamental theorem of calculus holds and integrating along the reverse path changes the sign of the integral.

The following example depicts the situation which we do encounter in the Chapters 4 and 5.

Example 2.6.6. Let X be the compact Riemann surface associated to a plane algebraic curve $C_f : f = 0$, where $f \in \mathbb{C}[x, y]$ is irreducible, and $a(x, y) \in \mathbb{C}(C_f)$ be a rational function. Moreover, let $\gamma : [-1, 1] \to \mathbb{C}$ be a smooth path and

$$\tilde{\gamma} : [-1, 1] \to X,\ t \mapsto (\gamma(t), y(\gamma(t)))$$

where $y(\gamma(t))$ is obtained via analytic continuation of the algebraic function $f(x, y(x)) = 0$ along γ. If $a(x, y)$ is defined in $\tilde{\gamma}([-1, 1])$, then with $x = \gamma(t)$ we have

$$\int_{\tilde{\gamma}} a(x, y(x))\mathrm{d}x = \int_{-1}^{1} a(\gamma(t), y(\gamma(t))\gamma'(t)\mathrm{d}t\,.$$

Recall from §2.4.3 that we have already defined paths, the concept of homotopy and the fundamental group for connected real manifolds. The same definitions can also be used for Riemann surfaces (except that we additionally require paths to be piecewise $\mathcal{C}^\infty$ functions).

Let us briefly mention that a 1-form ω is *closed* if its total derivative d (see [63, p. 114] for a Definition) is zero, i.e. $\mathrm{d}\omega = 0$, and that every holomorphic 1-form is closed. The following basic results from complex analysis concerning homotopy and contour integration carries over to Riemann surfaces (see [63, Proposition IV.3.20]):

Proposition 2.6.7. *Let γ_0 and γ_1 be homotopic paths on a Riemann surface X. Then for any closed 1-form ω on X (in particular, for holomorphic 1-forms), we have*

$$\int_{\gamma_0} \omega = \int_{\gamma_1} \omega$$

This means that the integral of a closed 1-form along a path only depends on the homotopy class of that path, and not on the path itself. Let now $\pi_1(X, P_0)$ denote the homotopy group consisting of homotopy classes of closed paths on X that are based at P_0, which we will also refer to as *cycles*. By Proposition 2.6.7, for any closed 1-form ω, the integral induces a well-defined map

$$\int : \pi_1(X, P_0) \to \mathbb{C},\ \gamma \mapsto \int_\gamma \omega\,.$$

For a fixed 1-form this map is a group homomorphism and since $\mathbb{C}$ is abelian it must factor through the abelianization $\pi_1(X, P_0)/[\pi_1, \pi_1]$ where

$$[\pi_1, \pi_1] = \{\, aba^{-1}b^{-1} \mid a, b \in \pi_1(X, P_0) \,\}\,.$$

is the commutator subgroup. In particular, $[\pi_1, \pi_1]$ is the kernel of the map (2.6.2) and integration of ω induces a well-defined group homomorphism from the quotient group $\pi_1(X, P_0)/[\pi_1, \pi_1]$ to $\mathbb{C}$.

Definition 2.6.8 (Homology group)**.** We define the *first (integral) homology group of X* as the quotient group $\pi_1(X, P_0)/[\pi_1, \pi_1]$ and denote it by $H_1(X, \mathbb{Z})$. If X is a compact Riemann surface, then $H_1(X, \mathbb{Z})$ is a free abelian group of rank $2g$ and therefore isomorphic to $\mathbb{Z}^{2g}$. Moreover, the homology group $H_1(X, \mathbb{Z})$ is independent of the base point P_0. In the following, we identify the cycles in $\pi_1(X, P_0)$ with their classes in $H_1(X, \mathbb{Z})$ (also called 1-*cycles*).

Therefore, integration of closed 1-forms along 1-cycles on X gives us a well-defined map

$$\int : H_1(X, \mathbb{Z}) \to \mathbb{C},\ \gamma \mapsto \int_\gamma \omega \tag{2.8}$$

which only depends on the homology class of $\gamma \in H_1(X, \mathbb{Z})$. Conversely, for every cycle $\gamma \in H_1(X, \mathbb{Z})$, we obtain a well-defined functional on the space $\Omega^1(X)$ of holomorphic 1-forms, given by integration

$$\int_\gamma : \Omega^1(X) \to \mathbb{C}\,. \tag{2.9}$$

Definition 2.6.9. A linear functional (2.9) that is induced by $\gamma \in H_1(X, \mathbb{Z})$ is called *period*. The set Λ of periods, called the *period lattice*, forms a subgroup of the dual space $\Omega^1(X)^*$. For a compact Riemann surface X we define the Jacobian $\mathrm{Jac}(X)$ of X as the quotient

$$\mathrm{Jac}(X) = \frac{\Omega^1(X)^*}{\Lambda}\,. \tag{2.10}$$

Later on, in Section 2.9, we give descriptions of the period lattice and the Jacobian with respect to bases of $\Omega^1(X)$ and $H_1(X, \mathbb{Z})$ that is much more useful (for practical purposes) than this construction.

2.7 Intersection theory

Let X be a compact Riemann surface of genus $g > 0$. In the following we briefly introduce intersections between cycles on X. More details can be found in [33, III.1], [10, p.105 ff.] or [7, 1.3].

For any two classes $H_1(X, \mathbb{Z})$ one can find representatives $\gamma_1, \gamma_2 \in \pi_1(X)$ which are $\mathcal{C}^\infty$ cycles without self-intersections on X and satisfy the following conditions:

- $\gamma_1([-1,1]) \cap \gamma_2([-1,1])$ is finite;
- for any $P \in \gamma_1([-1,1]) \cap \gamma_2([-1,1])$ there exist unique $t_1, t_2 \in [-1,1]$ such that $\gamma_1(t_1) = \gamma_2(t_2) = P$, and that $(\gamma_1'(t_1), \gamma'(t_2))$ is a basis of the tangent space of X at P as shown in Figure 2.1, i.e. they *intersect transversally*.

We define the *intersection number at* P as

$$(\gamma_1 \circ \gamma_2)_P = \pm 1\,,$$

depending on the orientation of the basis $(\gamma_1'(t_1), \gamma_2'(t_2))$ of the tangent space at P as shown in Figure 2.1.

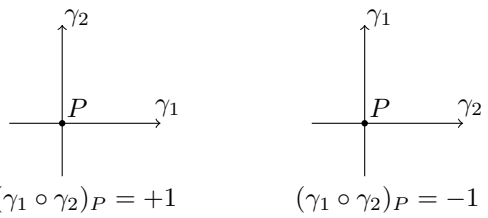


Figure 2.1: Intersection number of cycles at point P.

Definition 2.7.1. Let γ_1, γ_2 be two smooth cycles intersecting transversally in finitely many points. We define the *intersection number* of γ_1 and γ_2 to be

$$(\gamma_1 \circ \gamma_2) = \sum_{P \in \gamma_1 \cap \gamma_2} (\gamma_1 \circ \gamma_2)_P\,.$$

Theorem 2.7.2. *[7, Theorem 10] The intersection pairing is a bilinear skew-symmetric map*

$$\circ : H_1(X,\mathbb{Z}) \times H_1(X,\mathbb{Z}) \to \mathbb{Z} \quad (\gamma_1,\gamma_2) \mapsto (\gamma_1 \circ \gamma_2)$$

For a basis $\Gamma = (\gamma_1, \ldots, \gamma_{2g})$ of $H_1(X,\mathbb{Z})$ is called *homology basis*. The intersection numbers of the cycles in Γ is non-degenerate and can be represented by a matrix

$$K_\Gamma = (\gamma_i \circ \gamma_j)_{1\le i,j\le g} \in \mathbb{Z}^{2g}\,,$$

called *intersection matrix*. The following kind of homology basis will be of great importance throughout this thesis.

Definition 2.7.3. A basis $(\alpha_i,\ \beta_j)$ $(1 \le i,j \le g)$ of $H_1(X,\mathbb{Z})$ is called *canonical basis* if the intersection numbers satisfy

$$\alpha_i \circ \beta_j = \delta_{ij} \quad \text{and} \quad \alpha_i \circ \alpha_j = \beta_i \circ \beta_j = 0\,.$$

The intersection numbers of a canonical basis are represented by the intersection matrix

$$J = \begin{pmatrix} 0 & I_g \\ -I_g & 0 \end{pmatrix}.$$

Figure 1.1 shows a canonical basis for a 3-holed torus (which is diffeomorphic to a compact Riemann surface of genus $g = 3$).

2.8 Divisors and meromorphic functions

As we have already established , the theory of Riemann surfaces can be formulated equivalently in the language of smooth algebraic curves C defined over $\mathbb{C}$ or algebraic function fields in one variable over $\mathbb{C}$. Since we already introduced divisors and Riemann-Roch spaces for algebraic function fields in §2.3, we simply mention the important correspondences. For more details see [63, Chapter V].

- points on the compact Riemann surface X correspond one-to-one to places of the function field $\mathbb{C}(C)$;
- the divisors on X correspond one-to-one to divisors on $\mathbb{C}(C)$.
- the field of meromorphic functions on X is isomorphic to the function field of C, i.e. $\mathcal{M}(X) \cong \mathbb{C}(C)$;
- differential forms on X and the differentials of $\mathbb{C}(C)$, in particular $\Omega^1(X) \cong \Omega_{\mathbb{C}(C)}(0)$;
- the order v_P of a meromorphic function (or a differential form) at a point $P \in X$ and the valuation $v_{P'}$ of a function field element (or a differential) at the place P' that corresponds to P.

2.9 The Abel-Jacobi map

In this section we follow the exposition of [90, Section 2] as well as [63, Chapter VIII].

Let X be a compact Riemann surface of genus $g > 0$. Recall from §2.6.2 that for any meromorphic 1-form ω on X the integral (2.8) along $\gamma \in H_1(X,\mathbb{Z})$ is well-defined. In particular, this is true for holomorphic 1-forms.

Definition 2.9.1. Let $\bar{\omega} = (\omega_1, \ldots, \omega_g)$ be a basis of the space $\Omega^1(X)$ of holomorphic 1-forms. With respect to that basis we define the *period lattice* Λ of X

$$\Lambda = \left\{ \int_\gamma \bar{\omega} \mid \gamma \in H_1(X, \mathbb{Z}) \right\} \subset \mathbb{C}^g,$$

where $\int_\gamma \bar{\omega} = \left(\int_\gamma \omega_1, \ldots, \int_\gamma \omega_g \right)^T$. Then the Jacobian (2.10) of X is isomorphic to the complex torus

$$\mathrm{Jac}(X) \cong \mathbb{C}^g / \Lambda \,.$$

In fact, $\mathrm{Jac}(X)$ is a abelian variety of dimension g (see [10, p.189]), i.e. as a complex manifold it can be embedded in a projective space.

Let $P_0 \in X$ be a base point on X. With respect to P_0 we define the *Abel-Jacobi map* as the map

$$\mathcal{A}(\cdot, P_0) : X \to \mathrm{Jac}(X),\ P \mapsto \int_{P_0}^{P} \omega \mod \Lambda \,. \tag{2.11}$$

Note that $\mathcal{A}$ is independent of the path taken from P_0 to P. By linearity of the integral we can extend this map from X to the group $\mathrm{Div}(X)$ of divisors on X. For $D = \sum_{P \in X} v_P P \in \mathrm{Div}(X)$ we define

$$\mathcal{A}(\cdot, P_0) : \mathrm{Div}(X) \to \mathrm{Jac}(X),\ D \mapsto \sum_{P \in X} v_P \mathcal{A}(P, P_0) \,. \tag{2.12}$$

Restricting the Abel-Jacobi map to subgroup of degree zero divisors $\mathrm{Div}^0(X)$ removes the dependence of the base point.

We are ready to state the famous theorem by Abel that gives a characterization of divisors of meromorphic functions on X in terms of the Abel-Jacobi map.

Theorem 2.9.2 (Abel's theorem)**.** *Let X be a compact Riemann surface of genus g and $D \in \mathrm{Div}^0(X)$ a degree zero divisor on X. Then, D is the divisor of a meromorphic function (a principal divisor) if and only if its image under the Abel-Jacobi map is in the period lattice, i.e.*

$$D \in \mathrm{Prin}(X) \quad \Leftrightarrow \quad \mathcal{A}(D) \equiv 0 \mod \Lambda \,.$$

The surjectivity of the Abel-Jacobi map is due to Jacobi [33, p.92]:

Theorem 2.9.3 (Jacobi inversion)**.** *Every element $v \in \mathrm{Jac}(X) = \mathbb{C}^g/\Lambda$ is the image of an divisor $D \in \mathrm{Div}(X)$ of degree g under $\mathcal{A}$. In particular, the Abel-Jacobi map* (2.12) *is surjecive.*

Combining Abel's and Jacobi's theorems provides an explicit isomorphism of abelian groups

$$\mathcal{A} : \mathrm{Div}^0(X) / \mathrm{Prin}(X) \to \mathrm{Jac}(X) \,.$$

In particular, for a complex smooth algebraic curve $C/\mathbb{C}$ such that $X = C(\mathbb{C})$, via this isomorphism we can identify the algebraic Jacobian with the analytic Jacobian

$$\mathrm{Div}^0(C/\mathbb{C}) / \mathrm{Prin}(C/\mathbb{C}) \cong \mathrm{Jac}(X) \,.$$

Remark 2.9.4. The group of divisors modulo principal divisors is called *Picard group* and denoted by $\mathrm{Pic}(X) = \mathrm{Div}(X)/\mathrm{Prin}(X)$. Likewise, if we denote by $\mathrm{Pic}^0(X)$ the subgroup of $\mathrm{Pic}(X)$ formed by the classes of degree zero divisors, then we can state the aforementioned isomorphism as

$$\mathrm{Pic}^0(X) \cong \mathrm{Jac}(X) \,.$$

2.9.1 Period matrices

Let (α_i, β_j) for $1 \leq i, j \leq g$ be a canonical basis for the homology group $H_1(X, \mathbb{Z})$, see Definition 2.7.3 and, as before, $\bar{\omega} = (w_1, \ldots, w_g)$ a basis for $\Omega^1(X)$. With respect to these bases we define period matrices

$$\Omega_A = \left(\int_{\alpha_j} \omega_i \right)_{1 \leq i,j \leq g} \quad \text{and} \quad \Omega_B = \left(\int_{\beta_j} \omega_i \right)_{1 \leq i,j \leq g} .$$

We call the concatenated matrix

$$\Omega = (\Omega_A, \Omega_B) \in \mathbb{C}^{g \times 2g}$$

such that $\Lambda = \Omega \mathbb{Z}^{2g}$ a *big period matrix*. If one takes as basis of differentials the dual basis of the cycles α_i, the matrix becomes

$$\Omega_A^{-1} \Omega = (I_g, \tau),$$

where $\tau = \Omega_A^{-1} \Omega_B \in \mathbb{C}^{g \times g}$ is called a *small period matrix*.

Riemann's bilinear relations The following two statements about period matrices are known as *Riemann's bilinear relations*.

(1) The period matrices Ω_A and Ω_B satisfy $\Omega_A^T \Omega_B = \Omega_B^T \Omega_A$.

(2) The small period matrix τ lies in the Siegel upper half-space $\mathcal{H}_g$ of symmetric complex matrices with positive definite imaginary part.

For proofs we refer to Lemmas 4.5 and 4.7 of [63].

In particular, the $2g$ columns of Ω are linearly independent over $\mathbb{R}$ which implies that, as an abelian group, the Jacobian is isomorphic to $2g$ copies of $\mathbb{R}/\mathbb{Z}$, i.e.

$$\mathrm{Jac}(X) \cong (\mathbb{R}/\mathbb{Z})^{2g} .$$

We are going to use this identification for representing the image of the Abel-Jacobi map in Sections 4.9 and 5.4.

Chapter 3

Numerical integration methods

In this chapter we discuss several numerical integration methods that we are going to apply to our problem of integrating differential forms on Riemann surfaces. The focus lies on achieving high numerical precision (hundreds or thousands of digits) and using error bounds that depend mainly on the holomorphicity of the integrand.

While we did not invent any new integration method, we analyzed their complexity, applicability and performance in a particular multiprecision setting. All of the following methods were implemented in MAGMA which had none of this functionality:

- Gauss-Jacobi quadrature §3.2.1
- Gauss-Legendre quadrature §3.2.2
- Gauss-Chebyshev quadrature §3.2.3
- Clenshaw-Curtis quadrature §3.3
- Double-exponential integration (or tanh-sinh-quadrature) §3.4

For each integration method we will give a brief introduction, present error bounds and analyze algorithms that we implemented to compute the corresponding integration scheme (i.e. abscissas and weights). Although there have been comparisons of these integration methods for arbitrary precision, we will give a summary of them and add our experiences. The performance of these methods applied to the integration of differential forms will be compared in §4.7.6 for general algebraic curves and §5.7 for superelliptic curves.

Numerical integration in our context means approximating contour integrals of the form

$$\int_{\tilde{\gamma}} \omega = \int_{-1}^{1} a(\tilde{\gamma}(u))\gamma'(u)\mathrm{d}u =: \int_{-1}^{1} w(u)g(u)\mathrm{d}u. \tag{3.1}$$

Here, $a(x, y)$ is a rational function with a finite set of poles on a Riemann surface, $\gamma : [-1, 1] \to \mathbb{C}$ is an oriented smooth path in the complex plane avoiding the x-coordinates of these poles (except possibly at the end points), $\tilde{\gamma}$ is a lift of γ to the Riemann surface and $w(u)$ is a positive, integrable weight function. These objects will be explained later in more detail, but for now its important that we (mostly) deal with complex-valued functions g that are holomorphic and bounded in a neighborhood of $]-1, 1[$.

We seek to numerically approximate the integral (3.1) by a finite sum

$$I(g) := \int_{-1}^{1} w(u)g(u)\mathrm{d}u \approx \sum_{\ell=1}^{N} w_\ell g(u_\ell) =: I_N(g) \tag{3.2}$$

with abscissas $\{u_\ell\}_{1\le l\le N}$ and weights $\{w_\ell\}_{1\le \ell\le N}$ for some number $N > 0$; denoting by $E(N)$ the absolute error

$$E(N) = |I(g) - I_N(g)|\,. \tag{3.3}$$

We will analyze the complexity of the initialization of each integration scheme $\mathrm{Init}(N, D)$ depending on the number of abscissas N and some prescribed precision $D > 0$. Using suitable error bounds, we are able to introduce the minimal number of abscissas $N_{\min}(D)$ that is required to achieve some desired accuracy $D > 0$, i.e.

$$N_{\min}(D) = \min\{\, N > 0 \mid E(N) \leq e^{-D} \,\},$$

making it possible to express the initialization cost depending solely on D via

$$\mathrm{Init}(D) = \mathrm{Init}(D, N_{\min}(D)) \,.$$

Although the integration methods are (theoretically) unlimited in terms of precision, practically we are interested in

- up to ~ 2000 decimal digits of precision, i.e. $D_{10} = D/\log_{10}(D) \leq 2000$,
- which may require up to $N \approx 50000$ abscissas.

Remark 3.0.5. Note that the integral of a complex-valued function g is defined by the sum of the integrals of its real and imaginary part

$$\int g(u)\mathrm{d}u = \int \mathrm{Re}(g(u))\mathrm{d}u + i \int \mathrm{Im}(g(u))\mathrm{d}u \,.$$

If g is holomorphic on a set $U \subset \mathbb{C}$ then its real and imaginary part are harmonic functions on U and therefore analytic as functions on $\mathbb{R}^2$. Moreover, if g is bounded by $M > 0$ on U then so are $\mathrm{Re}(g)$ and $\mathrm{Im}(g)$. So, if we want the error $E(N)$ to be smaller than some $\varepsilon > 0$, we need to have errors of $\varepsilon_1, \varepsilon_2 > 0$ for the real and imaginary part such that $\varepsilon_1 + \varepsilon_2 \leq \varepsilon$, because

$$\begin{aligned} E(N) &= |I(g) - I_N(g)| \\ &= |I(\mathrm{Re}(g)) + I(\mathrm{Im}(g)) - I_N(\mathrm{Re}(g)) + I_N(\mathrm{Im}(g))| \\ &\leq |I(\mathrm{Re}(g)) - I_N(\mathrm{Re}(g))| + |I(\mathrm{Im}(g)) - I_N(\mathrm{Im}(g))| \\ &\leq \varepsilon_1 + \varepsilon_2 \leq \varepsilon. \end{aligned}$$

We will not mention this issue any further, but keep it in mind.

3.1 A versatile error bound

Integration formulas on $N > 0$ points that exactly integrate all polynomials of degree smaller than N will be called *interpolatory*. For interpolatory integration formulas with abscissas u_ℓ the quadrature weights w_ℓ are uniquely determined by integrating the polynomial of degree $N - 1$ that interpolates the N data points $(u_\ell, g(u_\ell)), l = 1, \ldots, N$. Moreover, we will call an integration formula *symmetric* if the abscissas on $[-1, 1]$ are reflected around 0 and the weights are symmetric.

Similar to [87] we will derive a bound on the eror $E(N)$ for such integration schemes, except that our weight function $w(u)$ does not need to be constant. We will denote the value of the integral of the weight function by

$$I(1) := \int_{-1}^{1} w(u)\mathrm{d}u > 0 \,.$$

Suppose we want to integrate a continuous function g on $[1, 1]$ using an interpolatory integration formula I_N. The *Chebyshev series* for g is defined as

$$g(u) = -\frac{1}{2}a_0 T_0(u) + \sum_{j=0}^{\infty} a_j T_j(u), \quad a_j = \frac{2}{\pi}\int_{-1}^{1} \frac{f(u)T_j(u)}{\sqrt{1-u^2}}\mathrm{d}u, \tag{3.4}$$

where $T_j(u)$ is the *Chebyshev polynomial* (of the first kind) of degree j (see (3.34)) and the a_j are called *Chebyshev coefficients*. On $[-1,1]$, $T_j(u)$ can also be defined trigonometrically as

$$T_j(u) = \cos(j \arccos(u)).$$

In the following we derive bounds on the error $E(N)$ for integrands that are holomorphic inside an ellipse with foci ± 1, parametrized by $r > 0$ via

$$\varepsilon_r = \{\, z \in \mathbb{C} \mid |z-1| + |z+1| = 2\cosh(r) \,\}, \tag{3.5}$$

the sum of the lengths of major and minor semiaxis being

$$\sinh(r) + \cosh(r) = e^r.$$

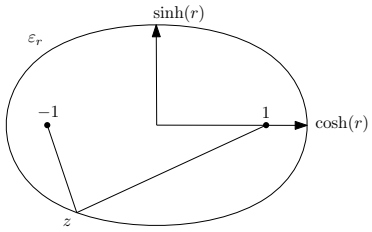


Figure 3.1: Parametrized ellipse.

Combining and generalizing several statements from [87] yields

Theorem 3.1.1. *Let $g : [-1,1] \to \mathbb{C}$ be a function that is holomorphic inside an ellipse ϵ_r (3.1) with foci ± 1 and $r > 0$. Then, for any interpolatory quadrature formula I_N on $N > 0$ points the error (3.3) satisfies*

$$E(N) \leq \frac{4MI(1)}{(1-e^{-r})e^{rN}} \tag{3.6}$$

where $M = \max\{\, |g(z)| \mid z \in \varepsilon_r \}$.

Proof. We write the integration error using the Chebyshev series for g and use that I_N is interpolatory, i.e.

$$I(g) - I_N(g) = \sum_{j=0}^{\infty} a_j (I(T_j) - I_N(T_j)) = \sum_{j=N}^{\infty} a_j (I(T_j) - I_N(T_j)). \tag{3.7}$$

Then we can estimate

$$\begin{aligned} E(N) &\leq \sum_{j=N}^{\infty} |a_j| \, |I(T_j) - I_N(T_j)| \\ &\leq \sum_{j=N}^{\infty} |a_j| \, (|I(T_j)| + |I_N(T_j)|) \\ &\leq \sum_{j=N}^{\infty} |a_j| \, (I(1) + I(1)) \end{aligned} \tag{3.8}$$

since $|T_j| \leq 1$ on $[-1,1]$. By [77, p.175] the Chebyshev coefficients decay exponentially on the ellipse ε_r; we have

$$|a_j| \leq \frac{2M}{e^{rj}} \quad \text{for } j \geq 0. \tag{3.9}$$

Plugging this into (3.8) and using the geometric series yields the claim

$$E(N) \leq 2I(1) \sum_{j=N}^{\infty} \frac{2M}{e^{rj}} = \frac{4MI(1)}{(1-e^{-r})e^{rN}}$$

□

In his paper [87] Trefethen shows that the estimates in (3.19) can be sharpened for Gauss-Legendre and Clenshaw-Curtis quadrature. Therefore, he improves the bound (3.6) in these cases by considering individual Chebyshev coefficients [87, p.77,78]. We will use the same techniques to achieve similar results for Gauss-Jacobi and Gauss-Chebyshev quadrature.

3.2 Gaussian quadratures

A numerical integration scheme is referred to as *Gaussian quadrature* on N points, if for some positive, integrable weight function $w(u)$, the approximation

$$I(g) = \int_{-1}^{1} w(u)g(u)\mathrm{d}u \approx \sum_{l=1}^{N} w_\ell g(u_\ell) = I_N(g) \tag{3.10}$$

is exact for all polynomials of degree smaller than $2N$. For every weight function $w(u)$ there is a class of orthogonal polynomials such that the abscissas u_ℓ occur as the simple roots of a polynomial of degree N of this class.

Hence Gaussian quadratures are by construction interpolatory and we obtain an even better bound on $E(N)$ than (3.6) by replacing N by $2N$ in the proof of Theorem 3.1.1.

Theorem 3.2.1. *Let Gaussian quadrature on $N > 0$ points be applied to a function $g : [-1,1] \to \mathbb{C}$ that is holomorphic inside an ellipse ϵ_r (3.5) with foci ± 1 and $r > 0$. Then, the error (3.3) satisfies*

$$E(N) \leq \frac{4MI(1)}{(1-e^{-r})e^{r2N}} \tag{3.11}$$

where $M = \max\{\, |g(z)| \mid z \in \varepsilon_r \}$.

In order to make the statement of Theorem 3.2.1 comparable to Theorem 3.4.1, our error bound for double-exponential integration, we formulate the following corollary:

Corollary 3.2.2. *For all $D > 0$, with constant r and M as in Theorem 3.2.1, if we choose N such that*

$$N \geq \frac{\log(4MI(1)) + D - \log(1-e^{-r})}{2r} \quad \textit{we have} \quad |E(N)| \leq e^{-D}\,,$$

which implies that asymptotically

$$N_{min}(D) \sim \frac{D}{2r} = O(D)\,. \tag{3.12}$$

As the asymptotic formula (3.12) for $N_{\min}(D)$ already indicates and as the comparison to other type of integration methods will confirm, Gaussian quadratures are very efficient in the sense that the error $E(N)$ converges quickly: only few integration points are required to achieve high accuracy.

As we will see later, the major drawback to (most) Gaussian quadratures is the immense initialization cost of the schemes. Another is the dependence on the parameter $r > 0$ which

can be close to zero when g has poles near $[-1,1]$. In that case $N_{\min}(D)$ will become large very quickly and Gaussian quadratures become impractical. In some cases the latter can be avoided by splitting up the integral, as explained in §4.7.5.

Integrable singularities at ± 1 can be handled by special weight functions, as we will now discuss.

3.2.1 Gauss-Jacobi quadrature

The Gaussian quadrature called Gauss-Jacobi quadrature on the interval $[-1,1]$ has weight function

$$w(u) = (1+u)^\alpha (1-u)^\beta \quad \text{with } \alpha, \beta > -1,$$

the abscissas $\{u_\ell\}_{1\le l\le N}$ are the roots of the Jacobi polynomial $p_N^{(\alpha,\beta)}$ and the weights are given in terms of the derivative $p_N'^{(\alpha,\beta)}$

$$w_\ell = \frac{\Gamma(N+\alpha+1)\Gamma(N+\beta+1)}{\Gamma(N+\alpha+\beta+1)N!}\frac{2^{\alpha+\beta+1}}{(1-u_\ell^2)p_N'^{(\alpha,\beta)}(u_\ell)^2}, \quad l=1,\dots,N. \tag{3.13}$$

Note that the Jacobi polynomials satisfy the symmetry relation

$$p_N^{(\alpha,\beta)}(-u) = (-1)^N p_N^{(\beta,\alpha)}(u). \tag{3.14}$$

In particular, Gauss-Jacobi integration with weight $\alpha = \beta$ is symmetric, i.e. the weights are symmetric by (3.13) and the roots of the Jacobi polynomial are reflected around 0, so we only need to compute $\lfloor N/2 \rfloor$ of them.

We applied Gauss-Jacobi quadrature in four different situations, namely with weight

- $\alpha = \beta = 0$ (Gauss-Legendre) for periods of general algebraic curves in Section 4.7;
- $\alpha = \beta = -\frac{1}{2}$ (Gauss-Chebyshev) for periods of hyperelliptic curves in §5.5.1;
- $\alpha = \beta \in]-1,0[$ for periods of superelliptic curves in §5.5.2;
- $\alpha \in]-1,0[, \beta = 0$ for the superelliptic Abel-Jacobi map §5.5.2.

The most important special cases for our applications, which we will analyze closely, are the Gauss-Legendre quadrature in §3.2.2 ($\alpha = \beta = 0$) and the Gauss-Chebyshev quadrature in §3.2.3 ($\alpha = \beta = -1/2$).

Error bound As Gaussian quadrature, the bound of Theorem 3.2.1 is valid for Gauss-Jacobi and can readily be used with (see Proof of [16, Theorem 5.1.])

$$I(1) = 2^{\alpha+\beta+1}\frac{\Gamma(\alpha+1)\Gamma(\beta+1)}{\Gamma(\alpha+\beta+2)} < \infty, \tag{3.15}$$

where

$$\Gamma(z) = \int_0^\infty x^{z-1}e^{-x}\mathrm{d}x \tag{3.16}$$

is the Gamma function. However, an improvement to the constant in (3.11) can be made in the case of $\alpha = \beta$ by taking a detailed look at the differences $I(T_j) - I_N(T_j)$ in equation (3.19). Through numerical experiments we found an interesting closed formula for $I(T_j)$:

Conjecture 3.2.3. For $\alpha > -1$ and even $j > 1$ we have

$$\int_{-1}^{1}(1-u^2)^{\alpha}T_j(u)\mathrm{d}u \;=\; (-1)^{j/2}2^j I(1)\prod_{k=1}^{j}\frac{(2\alpha+2k+1-j)}{(2\alpha+k+1)^2}\prod_{k=1}^{j/2}(\alpha+k)^2\,. \tag{3.17}$$

We remark that Conjecture 3.2.3 turned out to be correct, as it was proved by Alexey Chernov after he read this thesis.

Moreover, we know that $I(T_j) > I(T_{j+2})$.

For odd j we have that the Chebyshev polynomials are of odd degree and satisfy $T_j(-x) = -T_j(x)$. Since for $\alpha = \beta$ the Gauss-Jacobi integration is symmetric this implies

$$I(T_j) = 0 = I_N(T_j) \quad \text{for all odd } j.$$

Using this in the proof of Theorem 3.1.1 we obtain

Theorem 3.2.4. *Let Gauss-Jacobi quadrature with weight (α,α) on $N > 0$ points be applied to a function $g : [-1,1] \to \mathbb{C}$ that is holomorphic inside an ellipse ϵ_r (3.5) with foci ± 1 and $r > 0$. Assuming 3.2.3 is true, the error (3.3) satisfies*

$$E(N) \le \frac{2M(|I(T_{2N}|) + I(1))}{(1-e^{-2r})e^{r2N}} \tag{3.18}$$

where $M = \max\{\,|g(z)| \;|\; z \in \varepsilon_r\}$, $I(1)$ is given by (3.15) and $I(T_{2N})$ by (3.17). Replacing $I(T_{2N})$ by $I(1)$ yields an unconditional bound.

Proof. Starting in equation (3.19) of the proof of Theorem 3.1.1 we now have that

$$\begin{aligned} E(N) &\le \sum_{j=2N}^{\infty}|a_j|\,|I(T_j) - I_N(T_j)| \\ &= \sum_{j=N}^{\infty}|a_{2j}|\,(|I(T_{2j})| + |I_N(T_j)|) \\ &\le (|I(T_{2N})| + I(1))\sum_{j=N}^{\infty}|a_{2j}|\,. \end{aligned} \tag{3.19}$$

Using (3.9) and a geometric series proves the claim. If we don't want to assume Conjecture 3.2.3, we can just use $|I(T_{2j})| \le I(1)$ instead. □

Computing the scheme We implemented an algorithm that computes the Gauss-Jacobi abscissas and weights to arbitrary precision which is an adaptation of the C-routine 'gaujac' as described in [75, Chapter 4, p.155].

In the special case of Gauss-Legendre quadrature, which has more significance for our applications, this algorithm is called REC (Algorithm 3.2.5) and will be closely analyzed in §3.2.5. Nothing fundamental about the algorithm changes, except that the formulas for the recurrence relation and initial guesses become less involved (in the special case). In particular, both algorithms have the same complexities, namely

$$\mathrm{Init}(N,D) = O(N^2\mathcal{M}(D)) \quad \text{and} \quad \mathrm{Init}(D) = O(D^3\log^{1+\varepsilon}D), \tag{3.20}$$

see equations (3.26) and (3.27).

Let us mention that in fixed precision the authors of [42] give an $O(N)$ algorithm for computing the Gauss-Jacobi scheme, but, for the same reasons given in §3.2.2, their approach is useless in our multiprecision setting.

3.2.2 Gauss-Legendre quadrature

The important special case of Gauss-Jacobi quadrature with weight $(\alpha, \beta) = (0, 0)$ is called *Gauss-Legendre quadrature*, i.e. we have constant weight function

$$w(u) = 1.$$

For an N-point Gauss-Legendre integration on the interval $[-1, 1]$ the abscissas $\{u_\ell\}_{1 \le l \le N}$ are the roots of the N-th Legendre polynomial $p_N = p_N^{(0,0)}$, which can be defined via the three-term recurrence relation

$$\begin{aligned} &p_N(u) = \frac{1}{N}((2N-1)p_{N-1}(u)u - (N-1)p_{N-2}(u)), \quad N > 1, \\ &p_1(u) = u, \\ &p_0(u) = 1. \end{aligned} \tag{3.21}$$

The corresponding quadrature weights $\{w_\ell\}_{1 \le l \le N}$ (3.13) simplify to

$$w_\ell = \frac{2}{(1-u_k^2)p'_N(u_\ell)^2} \tag{3.22}$$

where the derivative of p_N at u_ℓ can be evaluated via the relation

$$p'_N(u_\ell) = \frac{N\left(p_N(u)u - p_{N-1}(u)\right)}{u^2 - 1}. \tag{3.23}$$

Error bound We are in the special case of Gauss-Jacobi quadrature with weight $(\alpha, \beta) = (0, 0)$, so we can use the error bound given by Theorem 3.2.4. In this case Conjecture 3.2.3 is not necessary, because we have the standard formula [87, p.78]

$$I(T_j) = \int_{-1}^{1} T_j(u)\mathrm{d}u = \frac{2}{1-j^2} \quad \text{for all even } j.$$

Moreover, we have that $I(1) = \int_{-1}^{1} 1\,\mathrm{d}u = 2$ and $|I(T_{2N})| + I(1) = \frac{2(2N)^2}{(2N)^2-1}$ for $j > 0$. Resulting in the bound

$$E(N) \le \frac{(4N)^2}{((2N)^2-1)(1-e^{-2r})e^{r2N}}. \tag{3.24}$$

For $N > 1$ we can now use $|I(T_{2N})| \le |I(T_4)| = 2/15$ to get exactly the bound (4.14) of [87, Theorem 4.5]

$$E(N) \le \frac{64M}{15(1-e^{-2r})e^{r2N}}, \tag{3.25}$$

which is used in practice.

Computing the scheme

There's a vast amount of literature on the computation of abscissas and weights of the Gauss-Legendre quadrature rule.

A nice overview of existing methods can be found in [42, Section 2]. Most of these methods are designed to compute the quadrature rule to machine precision 10^{-15}; some mention double precision 10^{-30}. Since we are interested in arbitrary precision, all methods that rely on precomputing certain values or polynomials are impractical for our purposes. The most basic method that yields an arbitrary precision scheme and is suitable for implementation in MAGMA is called REC algorithm (REC for recurrence). It is described and

analyzed in §3.2.2 and uses the three-term recurrence relation (3.21) to obtain the abscissas via Newton-iteration. For fixed precision the REC algorithm has complexity $O(N^2)$, although the authors of [42] find it to be $O(N^{1.7})$ in practice. Due to its (sub)quadratic behaviour, which is verified by our timings in Table 3.1, this algorithm becomes impractical for large values of N.

In recent developments several algorithms (e.g. [42], [39], [8]) that achieve an $O(N)$ complexity (for fixed precision) were published. We find that the so-called Glaser-Liu-Rohkling (GLR) algorithm [39] is best suited for generalization to arbitrary precison and hence for our purposes. A description and analysis of the (generalized) algorithm, in the case of Legendre polynomials, is given below.

Just recently, the paper [49] by Johansson and Mezzarobba appeared on arχiv. They present their arbitrary precision implementation for computing the Gauss-Legendre scheme in the C-library [47], which sets new standards in terms of speed. Moreover, they prove that if the precision is approximately the number of integration of point, i.e. $D \sim N$, we have that the complexity becomes $\tilde{O}(D^2)$, i.e. $O(D^2 \log^k D)$ for some $k \in \mathbb{Z}_{\geq 0}$. Due to the bad timing, we were not able to incorporate these new results into this thesis.

REC algorithm Our implementation of the REC algorithm is an adaptation of the C-routine 'gauleg' as described in [75, Chapter 4, p.152], which we will briefly summarize here.

Algorithm 3.2.5 (REC)**.** Computes the Gauss-Legendre abscissas and weights for $N > 0$ up to an error $\varepsilon > 0$.

(1) For $l = 1, \ldots, \lfloor N/2 \rfloor$

  (1.1) Set $\tilde{u}_l \leftarrow \cos\left(\frac{\pi(l-1/4)}{N+1/2}\right)$.

  (1.2) Repeat

    (1.2.1) Evaluate p_N and p'_N at $\tilde{u}_l$ using (3.21) and (3.23).

    (1.2.2) Newton-step: Set $u_\ell \leftarrow \tilde{u}_l$ and $\tilde{u}_l \leftarrow \tilde{u}_l - p_N(\tilde{u}_l)/p'_N(\tilde{u}_l)$.

  (1.2) until $|u_\ell - \tilde{u}_l| < \varepsilon$.

  (1.3) Compute w_ℓ with (3.22).

(2) Return $\{u_\ell, w_\ell\}$.

Remark 3.2.6. Instead of starting with the very simple approximation $\cos\left(\frac{\pi(l-1/4)}{N+1/2}\right)$, which yields about 7 correct decimal digits of u_ℓ, one could use more sophisticated asymptotic formulas that give better approximations for u_ℓ such that Newton's method converges faster. We tried the combination of the asymptotic formulas (3.4) and (3.7) of [42, §3.1], which give up to 15 correct digits, but this did not offer any advantage in our multiprecision implementation since for each l the first Newton-step is cheaper (performed with 8 digits of precision) than evaluating (3.4) and (3.7) of loc. cit. in double-precision (16 digits).

Choosing $\varepsilon < e^{-D}$ we can easily analyze the complexity in terms of real multiplications of precision D numbers. Recall from §1.3 that, for every l, only the last Newton-step needs to be performed in full precision, so that

$$\mathrm{Init}(N, D) = \lfloor N/2 \rfloor (O(1) + N\mathcal{M}(D) + \mathcal{M}(D) + \mathcal{M}(D)) = O(N^2 \mathcal{M}(D)). \tag{3.26}$$

If we asymptotically relate the minimal number of abscissas required for precision D, using $N_{\min}(D) = O(D)$ as suggested by (3.12), the complexity of the algorithm becomes

$$\mathrm{Init}(D) = O(N_{\min}(D)^2 \mathcal{M}(D)) = O(D^2 \mathcal{M}(D)) = O(D^3 \log^{1+\epsilon} D). \tag{3.27}$$

GLR algorithm Since we were not satisfied with the performance of the REC algorithm for large N, we implemented the Glaser-Liu-Rohkling (GLR) algorithm for Legendre polynomials as described in [39, §4.1] and extended it to arbitrary precision. For fixed precision, this algorithm computes the abscissas $\{u_\ell\}_{l=1,\dots,N}$ and weights $\{u_\ell\}_{l=1,\dots,N}$ in $O(N)$ time.

The approach of Glaser-Liu-Rohkling is based on the fact that the Legendre polynomials satisfy the ordinary differential equation

$$(1-u)^2 p_N''(u) - 2up_N'(u) + N(N+1)p_N(u) = 0. \tag{3.28}$$

Suppose that N is odd so that the first abscissa $u_1 = 0$ is known a priori and we only need to compute the $\frac{N-1}{2}$ positive roots of p_N. Starting from a root $u_\ell \geq 0$ of p_N, the algorithm finds the next bigger root u_{l+1} in two steps:

First, we apply a Runge-Kutta method [39, §2.3.1] on the interval $\theta \in [\pi/2, -\pi/2]$ with the initial condition $u(\pi/2) = u_\ell$ to the differential equation

$$\frac{\partial u}{\partial \theta} = -\left(\sqrt{\frac{1-u^2}{N(N+1)}} - \frac{u}{1-u^2}\frac{\sin(2\theta)}{2}\right)^{-1}, \tag{3.29}$$

as defined in [39, § 2.1]. The resulting value $u(-\pi/2)$ is an approximation $\tilde{u}_{l+1}$ to the root u_{l+1}. Crucially, this can be done in some fixed low precision, say 10 decimal digits. In the second step, we use Newton's method to refine this root up to the desired precision. In contrast to Algorithm 3.2.5 (REC), the evaluation of p_N and p_N' at $\tilde{u}_{l+1}$ is not done via the three-term recurrence relation, but by evaluating the Taylor expansions around u_ℓ

$$p_N(u_\ell + h) = \sum_{k=0}^{m} \frac{p_N^{(k)}(u_\ell)}{k!}h^k + \varepsilon_0 \text{ with } |\varepsilon_0| \leq \sup_{|u-u_\ell|\leq h}\left\{\frac{p_N^{(m+1)}(u)}{(m+1)!}h^{m+1}\right\} \tag{3.30}$$

and

$$p_N'(u_\ell + h) = \sum_{k=1}^{m} \frac{p_N^{(k)}(u_\ell)}{(k-1)!}h^{k-1} + \varepsilon_1 \text{ with } |\varepsilon_1| \leq \sup_{|u-u_\ell|\leq h}\left\{\frac{p_N^{(m+1)}(u)}{m!}h^{m}\right\} \tag{3.31}$$

at $h = \tilde{u}_{l+1} - u_k$. Evaluation of the derivatives at u_ℓ is done via the recursion

$$p_N^{(k)}(u_\ell) = \frac{2(k-1)u_\ell p_N^{(k-1)}(u_\ell) + (k^2 - 3k + 2 - N(N+1))p_N^{(k-2)}(u_\ell)}{1-u_\ell^2} \tag{3.32}$$

for $k \geq 2$ while $p_N^{(1)}(u_\ell)$ is known from the previous step and $p_N^{(0)}(u_\ell) = 0$. If we want $|\varepsilon_0|$ and $|\varepsilon_1|$ to be small enough to approximate u_{l+1} up to an error of $10^{-D_{10}}$, i.e. D_{10} decimal digits of precision, we need to evaluate the Taylor series (3.30) and (3.31) up to order $m = 2D_{10}$.

In the case where N is even the strategy is a little different because 0 is not a root, but an extremum of p_N. We can then find the first abscissa $u_1 > 0$ by using Runge-Kutta [39, § 2.3.1] on the interval $[0, -\pi/2]$ with initial condition $u(0) = 0$. The resulting value $u(-\pi/2)$ is an approximation to u_1. Afterwards, we use Newton's method as described above for refinement. For evaluating the derivatives with (3.32) we start with $p_N^{(1)}(u_\ell) = 0$ and compute $p_N^{(0)}(0)$ using (3.21).

Algorithm 3.2.7 (GLR)**.** Computes the Gauss-Legendre abscissas and weights up to an error $\varepsilon > 0$.

(1) Let $D_{10} \in \mathbb{Z}$ such that $10^{-D_{10}} < \varepsilon$.

(2) If N is odd start with initial root $u_1 = 0$, otherwise find u_1 by the procedure above.

(3) For $l = 2, \dots, \lceil N/2 \rceil$

(3.1) Evaluate the derivatives $p_N^{(k)}$ at u_{l-1} for $k = 0, \dots, 2D_{10}$ using (3.32).

(3.2) Find $\tilde{u}_l$ by solving equation (3.29) using Runge-Kutta.

(3.3) Repeat

(3.3.1) Evaluate p_N and p'_N at $\tilde{u}_l$ using (3.30) and (3.31) with $m = 2D_{10}$.

(3.3.2) Newton-step: Set $u_\ell \leftarrow \tilde{u}_l$ and $\tilde{u}_l \leftarrow \tilde{u}_l - p_N(\tilde{u}_l)/p'_N(\tilde{u}_l)$.

(3.3) until $|u_\ell - \tilde{u}_l| < \varepsilon$.

(3.4) Compute w_ℓ with (3.22).

(4) Return $\{u_\ell, w_\ell\}$.

Inductively this algorithm happily jumps from one root to the next bigger one until all roots are found. Analyzing the complexity of the GLR algorithm we notice that it achieves linear complexity in N at the cost of an extra factor D:

$$\mathrm{Init}(N, D) = \lceil N/2 \rceil (D\mathcal{M}(D) + O(1) + D\mathcal{M}(D) + \mathcal{M}(D)) = O(ND\mathcal{M}(D)).$$

Comparing this to the complexity (3.26) we notice that, if we take precision into account, the GLR algorithm is not strictly better than the REC algorithm: (GLR) scales better with N, but worse than (REC) with D.

If we go one step further and express the complexity solely in D (again using $N_{\min}(D) = O(D)$ as suggested by (3.12))) we obtain

$$\mathrm{Init}(D) = O(N_{\min}(D)D\mathcal{M}(D)) = O(D^2\mathcal{M}(D)) = O(D^3 \log^{1+\epsilon} D) \tag{3.33}$$

which is then equal to the complexity (3.27) of Algorithm 3.2.5. Surprisingly, (GLR) has no clear advantage over (REC) when put into a multiprecision setting. Hence, conducting an empirical comparison between the two algorithms seems appropriate.

Moreover, we conjecture that the algorithms given in [42] and [8], that run in $O(N)$ time for fixed precision, admit the same behaviour when being generalized to multiprecision.

Comparison between GLR and REC

As we have seen in §3.2.2 there is no clear best algorithm for the computation of the multiprecision Gauss-Legendre quadrature. Here, we want to compare our implementations of both algorithms in MAGMA for different values of D and N. The following tables show the absolute run time (in seconds); recall that D_{10} is a number of decimal digits.

D_{10}	100		500		1000		2000	
N	GLR	REC	GLR	REC	GLR	REC	GLR	REC
100	0.12	0.04	0.92	0.08	3.66	0.16	17.7	0.35
200	0.21	0.14	1.79	0.31	7.13	0.60	33.6	1.37
500	0.52	0.78	4.21	1.88	16.9	3.64	80.2	8.23
1000	1.02	3.00	8.31	7.33	33.1	14.2	157	32.1
2000	2.04	11.7	16.4	28.4	65.6	54.8	310	124
5000	5.19	74.9	41.2	175	163	321	772	712
10000	10.2	289	81.2	681	324	1297	1555	2829

Table 3.1: Timings(s) for GLR and REC.

Table 3.1 supports our complexity analysis. We conclude that depending on the ratio of N and D_{10} there is a preferable choice of one algorithm over the other. Running a few numerical tests, we approximated the values of N where both algorithms break even for several $50 \leq D_{10} \leq 1000$. In practice we can use Table 3.2 to decide which algorithm to use. For more than 1000 decimal digits we omit this comparison since neither of the algorithms remains useful.

D_{10}	50	100	200	300	500	750	1000	2000
N	250	300	500	700	1100	1900	2400	5400
N/D_{10}	5	3	2.5	2.3	2.2	2.5	2.4	2.7
GLR & REC	0.15	0.32	1.12	≈ 2.8	≈ 9.2	≈ 36.5	≈ 79	≈ 836

Table 3.2: Break even points with timings(s) of GLR and REC.

3.2.3 Gauss-Chebyshev quadrature

Gauss-Jacobi quadrature with weight $(\alpha, \beta) = (-\frac{1}{2}, -\frac{1}{2})$ is usually called *Gauss-Chebyshev quadrature*. In this case the weight function (3.10) is

$$w(u) = \frac{1}{\sqrt{1-u^2}}$$

and therefore $I(1) = \pi$. The corresponding orthogonal polynomials are the Chebyshev polynomials of the first kind $T_N(u) = p_N^{(-\frac{1}{2},-\frac{1}{2})}$ which are defined by the recurrence relation

$$\begin{aligned} &T_N(u) = 2uT_{N-1}(u) - T_{N-2}(u), \quad N > 1, \\ &T_0(u) = 1, \\ &T_1(u) = u\,. \end{aligned} \tag{3.34}$$

Since the roots of the Chebyshev polynomials are explicitly given as cosine functions (see [1, 25.4.38]) the integration formula is particularly simple: with abscissas and weights

$$\begin{cases} u_\ell = \cos\left(\frac{2l-1}{2N}\pi\right) \\ w_\ell = \frac{\pi}{N} \end{cases} \qquad \text{for } l = 1, \ldots, N,$$

the error $E(N)$ in (3.3) satisfies the estimate given by Theorem 3.2.1.

Error bound Using the same techniques as in §3.2.1 we can improve the bound (3.11) in this case as well. First note that for $\alpha = -1/2$, by orthogonality of the Chebyshev polynomials, we have

$$I(T_j) = \int_{-1}^{1} \frac{T_j(u)}{\sqrt{1-u^2}} \mathrm{d}u = \begin{cases} \pi, & j = 0 \\ 0, & j > 0. \end{cases}$$

Moreover, if u_ℓ are the abscissas of an N-point Gauss-Chebyshev integration, then it holds that for integer multiples j of $2N$

$$\left| \sum_{l=1}^{N} T_j(u_\ell) \right| = N \quad \text{for } j = 2kN,\ k \geq 0, \tag{3.35}$$

which leads to an error in integrating Chebyshev polynomials of

$$|I(T_j) - I_N(T_j)| = \begin{cases} \pi, & \text{if } j = 2kN, k > 0, \\ 0, & \text{otherwise.} \end{cases} \tag{3.36}$$

Using this error in the proof of Theorem 3.1.1 results in

Theorem 3.2.8. *Let Gaussian-Chebyshev quadrature on $N > 0$ points be applied to a function $g : [-1, 1] \to \mathbb{C}$ that is holomorphic inside an ellipse ϵ_r (3.5) with foci ± 1 and $r > 0$. Then, the error (3.3) satisfies*

$$E(N) \leq \frac{2M\pi}{e^{r2N} - 1} \tag{3.37}$$

where $M = \max\{\, |g(z)| \mid z \in \varepsilon_r \}$.

We remark that the bound (3.37) is exactly the bound of [15, Theorem 5], which they obtained by applying the residue theorem on the ellipse ε_r.

Computing the scheme Computing this integration scheme means evaluating $\lfloor N/2 \rfloor$ cosine functions to precision D and thus has complexity

$$\mathrm{Init}(N, D) = O(N\mathcal{T}(D))\,. \tag{3.38}$$

Using $N_{\min}(D) = O(D)$ the complexity becomes

$$\mathrm{Init}(D) = O(D \log D\mathcal{M}(D)) = O(D^2 \log^{2+\varepsilon} D)\,. \tag{3.39}$$

In contrast to Gauss-Legendre (or Gauss-Jacobi) the computational cost of this scheme is extremely cheap, even negligible in practice, while having the same order of convergence. Due to the specific weight function $w(u)$ we will only apply this method in the special case of integrals coming from hyperelliptic curves as explained in §5.5.1.

3.3 Clenshaw-Curtis quadrature

Here we consider another promising integration scheme, which is usually called *Clenshaw-Curtis* quadrature. For a quick introduction we follow the exposition of [95].

The Clenshaw-Curtis quadrature on $N + 1$ ($N \geq 2$) points is an interpolatory integration scheme that is exact for polynomials of degree at most N. As for Gauss-Legendre, we consider integration with constant weight functions $w(u) = 1$, so we want to approximate the definite integral of a continous function by a finite sum

$$I(g) = \int_{-1}^{1} g(u)\mathrm{d}u \approx \sum_{l=0}^{N} w_\ell g(u_\ell) = I_{N+1}(g) \tag{3.40}$$

and denote the error by

$$E(N) = |I(g) - I_N(g)|\,. \tag{3.41}$$

Here, the abscissas are defined as the extrema of the N-th Chebyshev polynomial $T_N(u)$ on the interval $]-1, 1[$ including the boundary points ± 1, i.e.

$$u_\ell = \cos(v_l) \quad \text{where} \quad v_l = l\frac{\pi}{N}\,, \quad l = 0, \ldots, N. \tag{3.42}$$

As for any interpolatory formula, the quadrature weights w_ℓ are uniquely defined by interpolation which results in

$$w_\ell = \frac{c_l}{N}\left(\sum_{j=1}^{[N/2]} \frac{b_j}{4j^2-1}\cos(2jv_l)\right), \quad l = 0,\dots,N, \tag{3.43}$$

where the coefficients b_j, c_l are defined as

$$b_j = \begin{cases} 1, & j = N/2, \\ 2, & j < N/2, \end{cases} \quad c_l = \begin{cases} 1, & l = 0 \bmod N, \\ 2, & \text{otherwise.} \end{cases} \tag{3.44}$$

Note that the Clenshaw-Curtis quadrature is a symmetric scheme, i.e. the abscissas are reflected around 0 and the weights are symmetric. In particular, combining equations (3.43) and (3.44) implies

$$w_0 = w_N = \frac{1}{N^2 - 1 + (N \bmod 2)}.$$

Error bound Since the N-point Clenshaw-Curtis quadrature is interpolatory and integrates exactly all polynomials of degree smaller than N, applying Theorem 3.1.1 with $I(1) = 2$ yields the error

$$E(N) \le \frac{8M}{(1-e^{-r})e^{rN}}. \tag{3.45}$$

Again we can improve this result using the same methods that we applied in §3.2.1. As for Gauss-Legendre quadrature,

$$I(T_j) = \int_{-1}^{1} T_j(u)\mathrm{d}u = \begin{cases} 2/(1-j^2) & \text{if } j \text{ is even}, \\ 0, & \text{if } j \text{ is odd}. \end{cases} \tag{3.46}$$

The Clenshaw-Curtis formula is interpolatory and symmetric, so we have that

$$|I(T_j) - I_N(T_j)| \begin{cases} \le I(1) + |I_N(T_j)| \le 2 + |I_N(T_N)| & \text{if } j \text{ is even and} \ge N, \\ = 0, & \text{otherwise.} \end{cases} \tag{3.47}$$

Assuming $N > 2$ we get that $|I_N(T_j)| \le |I_N(T_4)| = 2/15$ and obtain

Theorem 3.3.1. *Let Clenshaw-Curtis quadrature on $N > 2$ points be applied to a function $g : [-1,1] \to \mathbb{C}$ that is holomorphic inside an ellipse ϵ_r (3.5) with foci ± 1 and $r > 0$. Then, the error (3.41) satisfies*

$$E(N) \le \frac{64M}{15(1-e^{-2r})e^{rN}}, \tag{3.48}$$

where $M = \max\{\,|g(z)| \mid z \in \varepsilon_r\}$.

Trefethen mentions this bound for Clenshaw-Curtis quadrature (cf. [87, p.77]), but does not even state it as a separate result, because he thinks it is highly pessimistic. However, in our setting the bound is very accurate and the result useful.

Corollary 3.3.2. *For all $D > 0$, with r and M as in Theorem 3.3.1, if we choose N such that*

$$N \ge \frac{\log\left(\frac{64}{15}M\right) + D - \log(1-e^{-2r})}{r} \quad \textit{we have} \quad E(N) \le e^{-D},$$

which implies that asymptotically

$$N_{min}(D) \sim \frac{D}{r} = O(D). \tag{3.49}$$

Hence we need about twice as many points for Clenshaw-Curtis quadrature as for Gaussian quadratures in order to achieve the same accuracy. However, compared to its Gaussian counterpart (Gauss-Legendre), the Clenshaw-Curtis quadrature scheme can be computed much more efficiently.

Computing the scheme

Here we want to discuss and analyze how to compute the Clenshaw-Curtis abscissas and weights to arbitrary precision. The abscissas $u_\ell = \cos(l\frac{\pi}{N})$ are easily obtained using the cosine identity

$$\cos((l+2)x) = 2\cos(x)\cos((l+1)x) - \cos(lx). \tag{3.50}$$

Hence we can compute $\{u_\ell\}_{0\le l\le N}$ using $O(N\mathcal{M}(D))$ operations. This cost is negligible compared to the computation of the quadrature weights $\{w_\ell\}_{0\le l\le N}$ for which we are going to consider two different methods.

Classical algorithm By repeatedly applying (3.50) we can compute the weights w_ℓ essentially from the definition (3.43) using $O(N^2)$ real multiplications so that this classical algorithm (CL) has complexity in $\mathrm{Init}(N,D) = O(N^2\mathcal{M}(D))$. Using $N_{\min}(D) = O(D)$ we obtain

$$\mathrm{Init}(D) = O(N_{\min}(D)^2\mathcal{M}(D)) = O(D^3\log^{1+\epsilon} D).$$

Fortunately, there is an more elegant way to obtain the Clenshaw-Curtis weights due to their connection to the

Discrete Fourier transform The Discrete Fourier Transform (DFT) V of a vector $v = (v_0,\dots,v_N)$ of $N+1$ complex numbers is given by the formula

$$V_l = \sum_{j=0}^{N} v_l e^{2\pi i l j/N}, \quad l = 0,\dots,N \tag{3.51}$$

and, conversely, the Inverse Discrete Fourier Transform (IDFT) v of V is given by

$$v_l = \frac{1}{N}\sum_{j=0}^{N} V_l e^{-2\pi i l j/N}, \quad l = 0,\dots,N. \tag{3.52}$$

Any IDFT of a vector v can be written as combination of a DFT and complex conjugation

$$\mathrm{IDFT}(v) = \frac{1}{N}\overline{\mathrm{DFT}(\overline{v})}. \tag{3.53}$$

According to [95, Section 4], the Clenshaw-Curtis weights can be obtained as the IDFT of the real vector $v = v^{(1)} + v^{(2)} \in \mathbb{R}^N$ defined as

$$\begin{aligned} v_l^{(1)} &= \frac{2}{1-4l^2}, \quad l = 0,\dots,n-1, \\ v_{N-l}^{(1)} &= v_{l+1}^{(1)}, \quad l = 0,\dots,n-2, \\ v_l^{(1)} &= \frac{-1}{2n-1}, \quad l \in \{n+1, \lfloor (N+1)/2\rfloor + 1\}, \end{aligned} \tag{3.54}$$

where $n = \lfloor \frac{N}{2}\rfloor$ and, with $c = 1/(N^2 - 1 + (N \bmod 2))$,

$$v_l^{(2)} = \begin{cases} c(N-1), & \text{if } l \in \{1+n, 1+N-n\}. \\ -c, & \text{otherwise.} \end{cases} \tag{3.55}$$

Fast Fourier transform Computing the IDFT of v from the definition takes N^2 multiplications of complex numbers, which is worse than the classical algorithm. Fortunately, there is a famous algorithm (or rather a family of algorithms) called *Fast Fourier transform* (FFT) that computes a DFT of length N in $O(N \log N)$ time.

There exists a huge amount of literature and many different FFT algorithms and implementations that are suited for different situations. Their applicability depends mainly on the prime factor factorization of N. A nice overview of existing FFT algorithms is given in [30]. We will not go into detail about all these algorithms, but explain our multiprecision implementation of the FFT in MAGMA .

Algorithm 3.3.3 (FFT)**.** Computes the Discrete Fourier transform (3.51) of a complex vector of length $N > 1$.

(1) Factorize $N = \prod_{i=1}^{k} p_i^{s_i}$ into prime powers.

(2) Use the so-called *prime-factor algorithm* (PFA) or *Good–Thomas algorithm* [40]: The PFA expresses an FFT of size $N_1 \cdot N_2$ with N_1 FFTs of size N_2 and N_2 FFTs of size N_1 for coprime N_1, N_2. Naturally, we recursively apply it to $N_1 = p_i^{s_i}$ and $N_2 = N/(\prod_{i=1}^{k} p_i^{s_i})$, $i = 1, \ldots, k-1$ until only FFTs of prime power length are left.

(3) For each FFT of prime power length we use the classical *Cooley-Tukey algorithm* [20] with radix p, that recursively divides a FFT of length p^s into s FFT's of length p^{s-1}, until $s = 1$.

(4) The last step in each recursion is a FFT of prime length p (i.e. $s = 1$), for which we use Bluestein's algorithm (see [6]) which achieves $O(p \log p)$ complexity even for prime numbers if p is large enough, or we compute the FFT from the definition (3.51) if p is small.

In this way we can compute a DFT of length N in $O(N \log N)$ time, independently of the factorization of N. Therefore, applying (3.53) to the vector $v = v^{(1)} + v^{(2)}$ defined by (3.54) and (3.55), we can compute the Clenshaw-Curtis weights (and therefore the whole scheme) to precision D using

$$\mathrm{Init}(N, D) = O(\log N(N\mathcal{M}(D) + \mathcal{T}(D)) = O(\log N(N + \log D)\mathcal{M}(D)) \tag{3.56}$$

operations such that with $N_{\min}(D) = O(D)$ the complexity becomes

$$\mathrm{Init}(D) = O(\log D(D + \log D)\mathcal{M}(D)) = O(D^2 \log^{2+\varepsilon} D). \tag{3.57}$$

Remark 3.3.4. While (FFT) works over the complex numbers, the classical algorithm (CL) only requires real multiplications. Thus, for very high precision D and small N the classical algorithm may actually be faster than (FFT) due to better constants. This justifies the comparison between the two approaches described in §3.3.

Comparison between FFT and CL

As indicated by Remark 3.3.4 there should be combinations of D and N where the classical algorithm computes the Clenshaw-Curtis quadrature scheme faster than the approach utilizing the Fast Fourier transform. We compare the run time of our implementations of both algorithms in MAGMA for different values of D_{10} and N where $D_{10} = \frac{D}{\log(10)}$ be a number of decimal digits. Since in our setting N is rather large, ceiling the number of abscissas to a multiple of 4 gives a huge-speed up in practice. Assuming $4|N$, we can use $s = 2$ as hard-coded base of the recursion for $p = 2$.

D_{10}	100		500		1000		2000	
N	FFT	CL	FFT	CL	FFT	CL	FFT	CL
2^8	0.02	0.05	0.08	0.06	0.22	0.10	0.76	0.19
2^{10}	0.09	0.73	0.30	0.99	0.90	1.51	3.11	2.90
2^{12}	0.34	11.8	1.30	15.8	3.76	24.8	12.9	47.0
2^{14}	1.58	188	5.77	254	15.8	388	53.1	757
$2^2 \cdot (2^6 + 3)$	0.04	0.06	0.13	0.07	0.26	0.11	0.61	0.20
$2^2 \cdot (2^8 + 1)$	0.55	0.74	1.75	1.03	3.73	1.52	9.01	2.96
$2^2 \cdot (2^{10} + 7)$	3.87	12.2	15.9	16.5	45.3	24.7	155	47.7
$2^2 \cdot (2^{12} + 3)$	17.6	192	68.1	257	189	392	631	760
$2^2 \cdot 3 \cdot 5 \cdot 7$	0.03	0.12	0.08	0.17	0.16	0.25	0.46	0.48
$2^2 \cdot 5^2 \cdot 7^2$	0.36	16.6	1.03	22.4	2.30	34	6.11	67.3
$2^2 \cdot 11 \cdot 13 \cdot 17$	1.00	65.5	3.44	91.2	7.03	144	16.5	269
$2^2 \cdot 41 \cdot 103$	4.94	198	16.9	269	36.7	408	93.0	807

Table 3.3: Timings(s) for FFT and CL.

Clearly, Table 3.3 confirms our complexity analysis and our expectations: the Fast Fourier transform outperforms the classical algorithm in almost every situation, especially if N is highly composite, i.e. more than two prime factors or higher prime powers.

On the one hand our (FFT) implementation scales almost linearly with N, while (CL) suffers from quadratic behaviour. On the other hand (CL) scales better with the precision D, which is due to better constants (real versus complex multiplications).

However, in some cases (CL) is the preferable algorithm: either when the run time is negligible, or when $N = 4p$ where p is a small prime (not in range of the $O(p \log p)$ of Bluestein's algorithm). For the sake of completeness we experimentally determined the prime p for these cases where FFT and CL break even for $200 \leq D_{10} \leq 2000$.

D_{10}	200	300	500	1000	1500	2000
$p = N/4$	61	373	709	983	1667	2663
FFT & CL	0.05	∼1.84	∼7.6	∼22.3	∼97	∼318

Table 3.4: Break even points with timing(s) of FFT and CL.

3.4 Double-exponential integration

The *double-exponential integration* (DE) or *tanh-sinh-quadrature* was first introduced by Mori and Takashi in 1974 [83] and has turned out to be a useful and versatile tool in numerical integration. The scheme is based on the optimality of the trapezoidal rule when integrating along $]-\infty, \infty[$ with constant step length.

A comparison between integration schemes in a multiprecision setting, including Gauss-Legendre and double-exponential integration, has been carried out by Bailey in 2005 [3]. He pointed out that the advantages of the double-exponential integration scheme are its robustness and fast initialization. It does particularly well, compared to other schemes, for badly behaved integrands, e.g. unbounded derivatives or singularities at the end points.

For his comparison Bailey used the double-exponential integration as an adaptive scheme, without using any a priori knowledge of the integrand.

In 2010, Pascal Molin [64] gave rigorous error bounds for double-exponential integration for integrands that are holomorphic and bounded on certain domains (see 3.64). Our main goal in this section is to prove Theorem 3.4.1, which combines, improves and corrects Theorems 2.10 and 2.11 of [64]. A special case of Theorem 3.4.1 was proved in a joint paper with Molin. Here we give a slightly more detailed proof of a more general statement.

In the following we will introduce a versatile version of the double-exponential integration comparable to Gauss-Jacobi quadrature; it can be applied to the integration of differentials in the case of superelliptic curves (see §5.5.3) as well as in the case of general algebraic curves (see Section 4.7). As before, our goal is to approximate an integral (3.2), while having rigorous bounds for $E(N)$. Throughout this section, $\lambda \in [1, \frac{\pi}{2}]$ is a fixed parameter.

Let $\tilde{g} : [-1, 1] \longrightarrow \mathbb{C}$ be a complex-valued function that is holomorphic in a neighborhood of $]-1, 1[$. Assume that there exists a constant $\tilde{M}_1 > 0$ with

$$|\tilde{g}(u)| \leq \tilde{M}_1 \quad \text{for } u \in [-1, 1]$$

and that we want to numerically approximate the integral

$$\int_{-1}^{1} (1+u)^{\tilde{\alpha}} (1-u)^{\beta} \tilde{g}(u) \mathrm{d}u \quad \text{for } \tilde{\alpha}, \beta \in]-1, 0]. \tag{3.58}$$

Without loss of generality we may assume $\tilde{\alpha} = \beta$: If $\tilde{\alpha} < \beta$ we can write the integral as

$$I(g) = \int_{-1}^{1} (1+u)^{\tilde{\alpha}} (1-u)^{\tilde{\alpha}} g(u) \mathrm{d}u \tag{3.59}$$

with

$$g(u) := (1-u)^{\beta - \tilde{\alpha}} \tilde{g}(u) \quad \text{and } |g(u)| \leq M_1 := 2\tilde{M}_1 \text{ on } [-1, 1], \tag{3.60}$$

or vice versa if $\tilde{\alpha} > \beta$. Applying the double-exponential change of variable

$$u(t) = \tanh(\lambda \sinh(t)), \tag{3.61}$$

which introduces the derivative

$$u'(t) = \frac{\lambda \cosh(t)}{\cosh(\lambda \sinh(t))^2},$$

pushes the singularities at ± 1 to $\pm\infty$. Thus, using the relation

$$(1-u(t))(1+u(t)) = \frac{1}{\cosh(\lambda \sinh(t))^2}$$

the integral (3.59) becomes

$$\int_{\mathbb{R}} g(u(t)) \frac{\lambda \cosh(t)}{\cosh(\lambda \sinh(t))^{2\alpha}} \mathrm{d}t =: \int_{\mathbb{R}} g(t) \mathrm{d}t, \tag{3.62}$$

where we write $\alpha = \tilde{\alpha} + 1 > 0$ for convenience.

For $r \in]0, \frac{\pi}{2}[$ and $\lambda \sin(r) < \frac{\pi}{2}$ the zeros of cosh lie outside the strip of length $2r$, denoted

$$\Delta_r = \{z \in \mathbb{C} \mid -r < \mathrm{Im}(z) < r\}, \tag{3.63}$$

so the functions u and u' are holomorphic on Δ_r. We want to choose $r \in]0, \frac{\pi}{2}[$ as large as possible such that $g(u)$ has an analytic continuation to the domain

$$Z_r = u(\Delta_r) = \{\tanh(\lambda \sinh(z)) \mid z \in \Delta_r\}, \tag{3.64}$$

which has the shape of a burger wrapped around $]-1, 1[$ (see Figure 3.2) and there exists a constant $M_2 > 0$ with

$$|g(u)| \leq M_2 \text{ for } u \in Z_r. \tag{3.65}$$

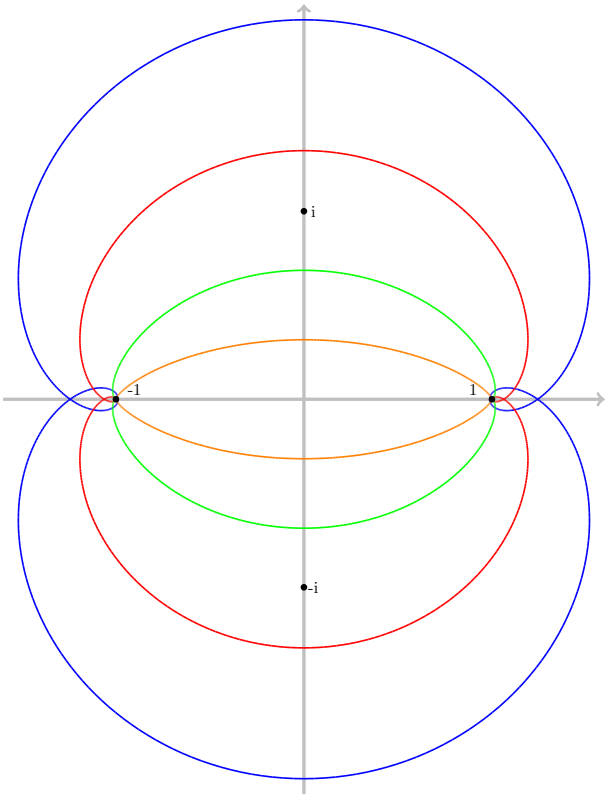


Figure 3.2: ∂Z_r for $r = \frac{\pi}{4}, \frac{\pi}{5}, \frac{\pi}{8}, \frac{\pi}{16}$.

Moreover, we introduce the following quantities

$$\begin{cases} X_r & = \cos(r)\sqrt{\frac{\pi}{2\lambda \sin r} - 1}, \\ B(r,\alpha) & = \frac{2}{\cos r}\left(\frac{X_r}{2}\left(\frac{1}{\cos(\lambda \sin r)^{2\alpha}} + \frac{1}{X_r^{2\alpha}}\right) + \frac{1}{2\alpha \sinh(X_r)^{2\alpha}}\right). \end{cases} \tag{3.66}$$

Once we have computed the two bounds M_1, M_2 and the constant $B(r, \alpha)$, we obtain a rigorous integration scheme as follows.

Theorem 3.4.1 (Double-exponential integration)**.** *For all $D > 0$, if we choose h and N such that*

$$h \leq \frac{2\pi r}{D + \log(2M_2 B(r,\alpha) + e^{-D})} \quad \textit{and} \quad Nh \geq \operatorname{asinh}\left(\frac{D + \log(\frac{2^{2\alpha+1} M_1}{\alpha})}{2\alpha\lambda}\right), \tag{3.67}$$

then

$$\left| \int_{\mathbb{R}} g(t)\mathrm{d}t - h \sum_{l=-N}^{N} w_\ell g(u_\ell) \right| \leq e^{-D},$$

where

$$u_\ell = \tanh(\lambda \sinh(lh)) \quad \textit{and} \quad w_\ell = \frac{\lambda \cosh(lh)}{\cosh(\lambda \sinh(lh))^{2\alpha}}.$$

Asymptotically we have

$$N_{min}(D) \sim \frac{D \operatorname{asinh}(D)}{2\pi r} = O(D \log D)\,. \tag{3.68}$$

The proof follows the same lines as the one in [64, Thm. 2.10]: we write the Poisson formula on $h\mathbb{Z}$ for the function $g(t)$

$$\underbrace{h \sum_{|l|>N} g(lh)}_{E_t} + h \sum_{l=-N}^{N} g(lh) = \int_{\mathbb{R}} g(t)\mathrm{d}t + \underbrace{\sum_{l \in \mathbb{Z}^*} \hat{g}\left(\frac{l}{h}\right)}_{E_d}$$

and control both error terms, the truncation error E_t by Lemma 3.4.2 and discretization error E_d by Lemma 3.4.3 below.

The actual parameters h and N in Theorem 3.4.1 follow by bounding each error by $e^{-D}/2$ (the condition of Lemma 6.4 being automatically satisfied).

Lemma 3.4.2 (Truncation error)**.** *For all N,h such that* $2\alpha \cosh(Nh) > 1$*, we have*

$$|E_t| \leq \frac{2^{2\alpha} M_1}{\alpha\lambda} \exp(-2\alpha\lambda \sinh(Nh)).$$

Proof. We bound the truncation error E_t using the integral of a (by assumption) decreasing function

$$\begin{aligned}
|E_t| \leq & \sum_{|l|>N} |hg(lh)| \\
\leq & 2M_1 \int_{Nh}^{\infty} \frac{\lambda \cosh(t)}{\cosh(\lambda \sinh(t))^{2\alpha}} \mathrm{d}t \\
= & 2M_1 \int_{\lambda \sinh(Nh)}^{\infty} \frac{1}{\cosh(t)^{2\alpha}} \mathrm{d}t \\
\leq & 2^{2\alpha+1} M_1 \int_{\lambda \sinh(Nh)}^{\infty} e^{-2\alpha t} \mathrm{d}t \\
= & \frac{2^{2\alpha} M_1}{\alpha} e^{-2\alpha\lambda \sinh(Nh)}.
\end{aligned}$$

□

Lemma 3.4.3 (Discretization error)**.**

$$|E_d| \leq \frac{2M_2 B(r,\alpha)}{e^{2\pi r/h} - 1}.$$

Proof. We start by bounding the Fourier transform by a shift of contour. For all $C > 0$ we have

$$\begin{aligned}
\hat{g}(\pm C) &= e^{-2\pi C r} \int_{\mathbb{R}} g(t \mp ir) e^{-2\pi i t C} \mathrm{d}t \\
\Rightarrow |\hat{g}(\pm C)| &\leq \frac{M_2}{e^{-2\pi C r}} \int_{\mathbb{R}} \left|u'(t \mp ir)\right| \mathrm{d}t\,.
\end{aligned}$$

Using $|u'(t+ir)| = |u'(t-ir)|$ and a geometric series we get that

$$\begin{aligned}|E_d| \leq \sum_{l\in\mathbb{Z}^*} \left|\hat{g}\left(\frac{l}{h}\right)\right| \leq 2M_2 \sum_{l>1} e^{-2\pi r l/h} \int_{\mathbb{R}} \left|u'(t+ir)\right| \mathrm{d}t \\ = \frac{2M_2}{e^{2\pi r/h}-1} \int_{\mathbb{R}} \left|\frac{\lambda\cosh(t+ir)}{\cosh(\lambda\sinh(t+ir))^{2\alpha}}\right| \mathrm{d}t.\end{aligned}$$

Now the point $\lambda\sinh(t+ir) = X(t)+iY(t)$ lies on the hyperbola

$$Y^2 = \lambda^2(\sin(r)^2 + \tan(rX^2)^2)$$

and

$$\begin{cases} |\lambda\cosh(t+ir)| & \leq \lambda\cosh(t) = \frac{X'(t)}{\cos(r)} \\ |\cosh(X+iY)|^2 & = \sinh(X)^2 + \cos(Y)^2 \end{cases}$$

so that

$$\int_{\mathbb{R}} \left|\frac{\lambda\cosh(t+ir)}{\cosh(\lambda\sinh(t+ir))^{2\alpha}}\right| \mathrm{d}t \leq \frac{2}{\cos r} \int_0^\infty \frac{\mathrm{d}X}{(\sinh(X)^2+\cos(Y)^2)^\alpha}.$$

For $X_0 = 0$, we get $Y_0 = \lambda\sin r < \frac{\pi}{2}$, and $Y_r = \frac{\pi}{2}$ for $X_r = \cos(r)\sqrt{\frac{\pi}{2Y_0}-1}$.

We split up the integral at $X = X_r$ and write

$$\int_0^{X_r} \frac{\mathrm{d}X}{(\sinh(X)^2+\cos(Y)^2)^\alpha} \leq \int_0^{X_r} \frac{\mathrm{d}X}{(X^2+\cos^2 Y)^\alpha} \tag{3.69}$$

$$\int_{X_r}^\infty \frac{\mathrm{d}X}{(\sinh(X)^2+\cos(Y)^2)^\alpha} \leq \int_{X_r}^\infty \frac{\mathrm{d}X}{(\sinh X)^{2\alpha}}. \tag{3.70}$$

We bound the integral (3.69) by convexity: since $Y(X)$ is convex and cos is concavely decreasing for $Y \leq Y_r$ we obtain by concavity of the composition that for all $X \leq X_r$

$$\cos(Y) \geq \cos(Y_0)\left(1 - \frac{X}{X_r}\right).$$

Now, $X^2 + \cos^2 Y \geq P_2(X)$ where

$$P_2(X) = \left(1 + \frac{\cos^2(Y_0)}{X_r^2}\right)X^2 - 2\frac{\cos^2(Y_0)}{X_r}X + \cos^2(Y_0)$$

is a convex quadratic function, so $X \mapsto P_2(X)^{-\alpha}$ is still convex and the integral (3.69) is bounded by a trapezoid:

$$\int_0^{X_r} \frac{\mathrm{d}X}{P_2(X)^\alpha} \leq \frac{X_r}{2}(P_2(0)^{-\alpha} + P_2(X_r)^{-\alpha}) = \frac{X_r}{2}\left(\frac{1}{\cos^{2\alpha}(Y_0)} + \frac{1}{X_r^{2\alpha}}\right). \tag{3.71}$$

For the integral (3.70) we use $\sinh(X) \geq \sinh(X_r)e^{X-X_r}$ to obtain

$$\int_{X_r}^\infty \frac{\mathrm{d}X}{\sinh(X)^{2\alpha}} \leq \frac{1}{2\alpha\sinh(X_r)^{2\alpha}}. \tag{3.72}$$

Combining (3.71) and (3.72) yields

$$\int_{\mathbb{R}} \left|\frac{\lambda\cosh(t+ir)}{\cosh(\lambda\sinh(t+ir))^{2\alpha}}\right| \mathrm{d}t \leq B(r,\alpha).$$

□

Remark 3.4.4 (Applications of (DE) integration).

- For our applications the value $\lambda = \frac{\pi}{2}$ is always a good choice, so we did not investigate this issue any further.
- In the case of general algebraic curves we will apply Theorem 3.4.1 with $\alpha = 1$.
- In the case of superelliptic curves we will apply Theorem 3.4.1 with $\alpha = 1 - j/m$, $1 \leq j < m$.

3.4.1 Adaptive double-exponential integration

As already mentioned in the beginning of this section, due to double-exponential decay of the integration weights, (DE) integration performs extraordinarily well for quite unruly integrands. In [3], Bailey et al. used (DE) integration, among other schemes, for high precision integration of several integrands with infinite derivative or blow-up singularities at one or both end points. The results were heavily in favor of (DE) integration. Following this, Bailey presents [2] his version of the double-exponential integration (he refers to it as tanh-sinh-quadrature) as an adaptive, non-rigorous scheme, that requires no a priori knowledge of the integrand (except that the definite integral is finite and infinitely differentiable). We will briefly summarize his tactic here and put in our context. Let $g :]-1, 1[\longrightarrow \mathbb{C}$ be a complex-valued function such that the definite integral on $[-1, 1]$ is well-defined and $\lambda = \frac{\pi}{2}$.

For given step length $h > 0$ and precision D, we approximate

$$\int_{-1}^{1} g(u)\mathrm{d}u = \int_{\mathbb{R}} g(u(t))u'(t)\mathrm{d}t \approx h \sum_{l=-N}^{N} w_j g(u_\ell) \tag{3.73}$$

where

$$u_\ell = \tanh(\lambda \sinh(lh)) \quad \text{and} \quad w_\ell = \frac{\lambda \cosh(lh)}{\cosh(\lambda \sinh(lh))^2}.$$

and N is chosen such that

$$|w_\ell g(u_\ell)| < e^{-D} \quad \text{for all} \quad l > N. \tag{3.74}$$

In practice, we can start with a step length of $h = 2^{-k}$ for some $k \geq 1$. We then compute pairs of abscissas and weights (u_ℓ, w_ℓ) until (3.74) is true and evaluate the sum (3.73). On the next level, say $h = 2^{-m}$ for $m \geq k$, we repeat this step. On level $k + 1$, we only need to evaluate the integrand at half the abscissas, since the abscissas at level k occur as subset of the abscissas of each higher level. Integration schemes with this property are called *nested*, they are particularly applicable for adaptive integration methods. A simple heuristic error estimation scheme can be found in [2, §4].

Later on (for the Abel-Jacobi map 4.9 associated to compact Riemann surfaces), we will deal with integrals

$$\int_{-1}^{1} a(\gamma(u), \tilde{\gamma}(u))\gamma'(u)\mathrm{d}u$$

that are, by theory, well defined, but the integrand $a(\gamma(u), \tilde{\gamma}(u))$ can not be evaluated at $u = 1$ since

$$|a(\gamma(u), \tilde{\gamma}(u))| \to \infty \quad \text{as} \quad u \to 1\,.$$

These singularities are not integrable, but polynomial in the sense that the denominator of $a(x, y)$ is a polynomial, $a(u)$ is an algebraic function. The weights w_ℓ decay double-exponentially as $lh \to \infty$, i.e. much faster than $|a(u)|$ approaches infinity for $u \to 1$, such that the values $|w_\ell g(u_\ell)|$ are well-defined as long as $u_\ell < 1$. We have to be careful though to avoid cancellation errors by working with sufficiently high precision $\tilde{D} > D$.

Computing the scheme Once we know N and h from Theorem 3.4.1 we can easily compute the double-exponential integration scheme. The hyperbolic functions $\sinh, \cosh$ and $\tanh$ are closely related (they're all defined in terms of the exponential function), so we only need to evaluate one exponential and a few real multiplications for each $l = 1, \ldots, N$.

More precisely, we require a total of $N + 1$ exponentials, $2N + 2$ divisions, $8N + 1$ multiplications to compute the scheme, which means it has complexity

$$\mathrm{Init}(D, N) = O(N\mathcal{T}(D)) \,. \tag{3.75}$$

Thus, with $N_{\min}(D) = O(D \log D)$ as suggested by Theorem 3.4.1, computing the abscissas u_ℓ and weights w_ℓ has complexity

$$\mathrm{Init}(D) = O(D^2 \log^{3+\varepsilon} D) \,. \tag{3.76}$$

Due to the small number of arithmetic operations, computing the scheme is exceptionally fast in practice, even for thousands of digits. Table 3.5 confirms our analysis: the run time is linear in N (the actual number of abscissas is $2N + 1$) and quasi-linear in the precision D.

N \ D_{10}	200	500	1000	2000
2000	0.03	0.06	0.17	0.53
5000	0.07	0.15	0.40	1.30
10000	0.13	0.29	0.79	2.59
20000	0.25	0.56	1.57	5.18

Table 3.5: Running times for DE scheme.

3.5 A priori comparison

We give a brief informal overview of the integration methods in form of Table 3.6. Note that the content of the Table only reflects the properties of our implementations of these methods as presented in this chapter.

Type	Interpolatory		Transformatory
Method	Gaussian	Clenshaw-Curtis	Double-exponential
$N_{\min}(D)$	$O(D)$	$O(D)$	$O(D \log D)$
Init(D)	$O(D^2 \mathcal{M}(D))$	$O(D \log D \mathcal{M}(D))$	$O(D \log^2 D \mathcal{M}(D))$
Robustness?	weak	weak	strong
Efficient splitting?	iff. $(\alpha, \beta) = (0, 0)$	yes	rarely
Adaptive integration?	no	yes	yes
Integrable singularities?	yes (Gauss-Jacobi)	no	yes
(α, β)	Initialization costs (absolute)		
> -1	very high	-	low
$\alpha = \beta$	high	-	low
$(0, 0)$	high	medium	low
$(-1/2, -1/2)$	very low	-	low

Table 3.6: Properties of integration methods.

3.6 Outlook

In this section we summarize the most impactful possible future improvements (speeding up the computation of the integration schemes).

Optimization Although we put quite some effort in implementing and optimizing the methods presented in this chapter, the timings are still far from optimal. Since none of these methods requires more functionality than basic multi-precision arithmetic of real/complex numbers, we could gain tremendous speed-ups by implementing everything in MAGMA's core which is written in C-language. Another option would be to compute outsource this task using another librarby and then import the integration schemes.

Discrete cosine transform We are aware that the Clenshaw-Curtis weights can also be computed using a *discrete cosine transform* (DCT) (see, for example, [86]). Similar to the (FFT) there are fast cosine transform algorithms that run in $O(N \log N)$ time and work over the real numbers. Having such an an algorithm at our disposal would further decrease the total run time that is needed to compute the integration scheme. However, we did not implement such an algorithm. Therefore, Clenshaw-Curtis integration has a slight disadvantage in our comparison of integration methods, see §4.7.6 and §4.8.3, due to sub-optimal initialization.

Computing the Gauss-Legendre scheme As already mentioned in the beginning of this chapter, Johansson and Mezzarobba in their paper [49] manage to compute the Gauss-Legendre scheme rigorously and incredibly fast in the C-library ARB [47], combining several formulas and using advances techniques for their evaluation. Having this algorithm at our disposal would certainly be optimal and give a big advantage to (GL) integration over the two other schemes in our comparisons §4.7.6 and §4.8.3, but this is definitely beyond the scope of this thesis.

Chapter 4

Computing period matrices & the Abel-Jacobi map: general case

The primary concerns of this chapter are the computation of period matrices and the Abel-Jacobi map of compact Riemann surfaces as defined in Chapter 2. In particular, we consider Riemann surfaces associated to irreducible affine plane curves, as defined in §2.5.3, that are defined over number fields. The ingredients that are required for this and how we compute them will be explained along the way.

This chapter explains in detail the algorithms that we implemented in the computer algebra system MAGMA for these purposes. Throughout, we point out the strengths and weaknesses of all the different approaches that we experimented with. We analyze the complexity of each algorithm that deals with inexact computations in terms of precision, while actual timings are given in several tables. Combining this information we conduct comparisons between different integration methods and approaches to period matrix computations. Whenever we rely on the heuristic assumptions made in §1.3, this will be indicated by the label 'heuristic'.

Structure of this chapter We start by explaining our setup and notation in Section 4.1, followed by the holomorphic differentials in Section 4.2. Our construction of the fundamental group of the punctured projective line is covered in Section 4.3. We talk about different root approximation methods in Section 4.4 that are used for analytic continuation along paths in the plane. In Section 4.5 we present an algorithm for analytic continuation and explain how to obtain a monodromy representation. Having these tools available, we are able to compute a canonical homology basis in Section 4.6. We explain in detail how to apply numerical integration methods, as introduced in the previous chapter, to efficiently integrate differentials forms in Section 4.7. Afterwards, it is straightforward to compute period matrices in Section 4.8. In Section 4.9 we present several ways of computing the Abel-Jacobi map. The issue of precision loss is touched in Section 4.10. Finally, in Section 4.11 we discuss the potential of symbolic integration and sum up our experiments.

4.1 The Riemann surface

Let K be a number field and $\iota : K \hookrightarrow \mathbb{C}$ be an embedding. Throughout this chapter C/K will be a smooth projective curve of genus $g > 0$. Moreover, C_f will be a (possibly singular) affine model for C

$$C_f : f = 0$$

where $f \in K[x,y]$ is geometrically irreducible. We denote by $d = \deg(f)$ the total degree of f and the degrees in one variable by $d_x = \deg_x(f)$ and $d_y = \deg_y(f)$. In practice, we require such an affine model as input and the smooth projective curve C/K is the non-singular projective model of the affine curve C_f (see Definition 2.2.3). If we rely on exact computations, we work with the K-rational function field of the curve

$$K(C) = K(C_f) = \mathrm{Quot}(K[x,y]/(f)),$$

rather than with the curve itself. In fact, we will be working almost exclusively with the associated compact Riemann surface (see §2.4.1)

$$X = C(\mathbb{C}),$$

also denoted by $X : f = 0$, if it is defined by $f \in K[x,y]$ (embedded into $\mathbb{C}[x,y]$ via ι).

Remark 4.1.1. (Affine plane models) Although an algebraic curve may also be given by more than one affine (or projective) equation, we restrict our algorithms to that case. By [70, Corollary 2.5.26] there always exists an affine plane model and it can be computed, for example, by computing a primitive element of the corresponding function field or by using a suitable projection to a plane (e.g. using the SAGE function 'plane_projection').

4.1.1 Holomorphic map to the projective line

Closely connected to compact Riemann surfaces (see §2.5) are holomorphic ramified covering maps to the projective line (see §2.5.1). For our approach to compact Riemann surfaces, choosing such a map is a necessity in order to do explicit computations.

As before, let C/K be a smooth projective curve defined over a number field K and let $K(C)$ the corresponding K-rational function field. Then for any rational function $s \in K(C)$ we can define a finite morphism to $\mathbb{P}^1$ via

$$\varphi_s : C(\overline{K}) \to \mathbb{P}^1(\overline{K}),\ P \mapsto \begin{cases} [s(P) : 1], & \text{if } P \text{ is not a pole of } s, \\ [1 : 0], & \text{otherwise.} \end{cases} \tag{4.1}$$

Of course, such a morphism becomes a holomorphic ramified covering map

$$\varphi_{\iota(s)} : X \to \mathbb{P}^1(\mathbb{C})$$

by extending the embedding $\iota : K \hookrightarrow \mathbb{C}$ to an embedding $K(C) \hookrightarrow \mathbb{C}(C) = \mathbb{C}(X)$. For simplicity, we will write $\mathbb{P}^1 := \mathbb{P}^1(\mathbb{C})$ as in §2.4.1.

For example, s could be the rational function $x \in K(C)$. Although there are many valid choices here, we will use the projection onto either the x or the y-coordinate, i.e. $s = x$ or $s = y$. Unless stated otherwise, for the rest of this chapter we assume that we project onto the x-coordinate, i.e. we choose the morphism φ_x of degree $m := \deg(\varphi_x) = d_y$. More details on that choice are given in §4.2.

4.1.2 Exceptional points

There are several points on $X : f = 0$ that have to be treated in a special way. Choosing the morphism $\varphi_x : X \to \mathbb{P}^1$ we consider f as a univariate polynomial in $(K[x])[y]$ of degree $m = d_y$ and may write it as

$$f = \sum_{k=0}^{m} a_k(x) y^{m-k}.$$

Let $\mathrm{disc}_y(f) \in K[x]$ be the discriminant of f with respect to y and $a_0(x) \in K[x]$ be the leading coefficient. We define the *exceptional values* of φ_{x} (or, equivalently, of f with respect to y) as the set

$$\mathcal{L} := \mathcal{L}(f,y) = \mathcal{L}(\varphi_x) = \{\, z \in \mathbb{C} \mid \mathrm{disc}_y(f)(z) = 0 \ \text{ or } \ a_0(z) = 0 \,\} \subset \mathbb{C} \tag{4.2}$$

and the *extended exceptional values* as

$$\hat{\mathcal{L}} = \mathcal{L} \cup \{\infty\} \subset \mathbb{P}^1 \,,$$

Note that the set of extended exceptional values contains the branch points of φ_{x}, i.e. $\mathcal{B} \subset \hat{\mathcal{L}}$, and therefore $\mathcal{R} \subset {\varphi_{\mathrm{x}}}^{-1}(\hat{\mathcal{L}})$.

If $a_0(x_P) \neq 0$, then any ramification point $P = (x_P, y_P) \in C_f(\mathbb{C})$ is a *critical point* of C_f, i.e. it satisfies $\partial_y f(x_P, y_P) = 0$. In that case, x_P is called a *critical value*. If both partial derivatives vanish at P, i.e. $\partial_y f(x_P, y_P) = \partial_x f(x_P, y_P) = 0$, we call P a *singular point* of C_f. If $x_P \in \mathbb{C}$ is a root of the leading coefficient $a_0(x) \in K[x]$, then there is at least one infinite point in the fiber $\#\varphi_x^{-1}([x_P : 1]) \subset X$. We will call such points *y-infinite points*. All y-infinite points correspond to the point $[0:1:0] \in \mathbb{P}^2$ in the projective closure $\overline{C}_f$.

Points at infinity With our morphism to the projective line φ_x chosen as explained above, we will call the poles of the meromorphic function x *points at infinity*, i.e. the points in the fiber ${\varphi_{\mathrm{x}}}^{-1}(\infty)$ above $\infty := [1:0] \in \mathbb{P}^1$. Although, ∞ may also be a branch point of φ_{x}, it is not needed for the period matrix computation. In MAGMA we can obtain the corresponding function field places by calling 'Poles(x)'. The places in the function field that correspond to y-infinite points can be either found via 'Zeros($a_0(x)$)' or 'Poles(y)'. Points at infinity and y-infinite points become important for the computation of the Abel-Jacobi map in Section 4.9.

Singular points One could also further analyze the (finite and infinite) singularities of the projective closure $\overline{C}_f$ by computing their multiplicities and delta invariants, as is done for example by [27] or [35]. This is necessary if one wants to obtain a desingularization of $\overline{C}_f$ over the complex numbers; in our approach this is done by MAGMA's function field functionality. Similar to the points at infinity, singular points can be ignored for the period matrix computation, but we have to deal with them if we want compute the Abel-Jacobi map for points lying over such singularities in Section 4.9.

Upper bound for the number of exceptional values For the complexity analysis of our period matrix algorithm it is crucial to have an upper bound for the number of exceptional values $\mathcal{L} \subset \mathbb{C}$. By [94, Theorem 6.2.2.], the degree of the discriminant is bounded by

$$\deg_x(\mathrm{disc}_y(f)) \leq (d_x + d_y)d$$

and since $\deg_x(a_0(x)) \leq d_x$ we have that

$$\#\mathcal{L} \leq (d_x + d_y)d + d_x \leq 2d^2 + d = O(d^2) \,.$$

4.1.3 Ordering of the sheets

After choosing a base point $x_0 \in \mathbb{C} \setminus \mathcal{L}$ (see §4.3.1) we can locally number the sheets of X (from 1 to m) by ordering the fiber above x_0

$${\varphi_{\mathrm{x}}}^{-1}(x_0) = \{P_1, \ldots, P_m\} \subset \mathbb{C} \,.$$

In this chapter and in practice, we choose to sort by increasing real part first and by increasing imaginary part second:

$$z_1 >_X z_2 \quad :\Leftrightarrow \quad \mathrm{Re}(z_1) < \mathrm{Re}(z_2) \;\text{ or }\; \mathrm{Re}(z_1) = \mathrm{Re}(z_2) \text{ and } \mathrm{Im}(z_1) < \mathrm{Im}(z_2)\,. \tag{4.3}$$

For every fiber above a *regular value* $z \in \mathbb{C} \setminus \mathcal{L}$ (i.e. non-exceptional value) the notion of sheet (see §2.4.3) is then well-defined with respect to the base point.

4.2 Holomorphic differentials

For an irreducible polynomial $f \in K[x, y]$, where K is a number field, we can compute a basis $\omega_1, \ldots, w_g$ of holomorphic differentials of the function field

$$K(C) = \mathrm{Quot}(K[x, y]/(f))$$

using the function field functionality of MAGMA which is due to Florian Hess [45]. The function 'BasisOfDifferentialsFirstKind' returns a basis of the form

$$\omega_i = a_i(x, y)\mathrm{d}u \quad \text{where} \quad a_i \in K(C) \tag{4.4}$$

and

$$u = \begin{cases} x, & \text{if } d_x < d_y, \\ y, & \text{if } d_x \geq d_y. \end{cases} \tag{4.5}$$

The algorithm's complexity is polynomial in d_x and d_y and it is incredibly fast in practice. It can become slow when the number field K is of large degree or the coefficients of f have huge height.

However, this is an exact calculation and therefore independent of precision. As such, we will not include this computation in our period matrix Algorithm 4.8.1, but assume such a basis of differentials to be given.

Embedding the coefficients of $\bar{\omega}_K = (\omega_1, \ldots, \omega_g)$ into $\mathbb{C}$ using the complex embedding $\iota : K \hookrightarrow \mathbb{C}$, we obtain a basis $\iota(\omega_1), \ldots, \iota(\omega_g)$ of holomorphic differentials of

$$K(C) \otimes_K \mathbb{C} \cong \mathbb{C}(C) = \mathbb{C}(X)$$

which is, due to category equivalence, a basis of $\Omega^1(X)$ (see §2.3.3). In the following we will assume that

$$\bar{\omega} = (\omega_1, \ldots, \omega_g) \tag{4.6}$$

is a vector of basis differentials for $\Omega^1(X)$ obtained in this way.

The choice of the local parameter (4.5) strongly suggests choosing our morphism §4.1.1 to the projective line accordingly. While choosing the morphism of smaller degree is a reasonable choice, it is not always the best choice for our purposes, see Remark 4.2.1 below. If we are not satisfied by the choice of 'BasisOfDifferentialsFirstKind', we can simply multiply all differentials ω_i by $\mathrm{d}y/\mathrm{d}x$ or $\mathrm{d}x/\mathrm{d}y$ respectively.

One could also try and compute the gonality of X (especially for low genus curves) in order to find a morphism $\varphi : X \to \mathbb{P}^1$ of minimal degree, but this could potentially result in a complicated defining equation which may ruin that advantage again.

As already indicated in §4.1.1 we will from now on assume that we project onto the x-coordinate, i.e. use the morphism φ_x (4.1) defined by $x \in K(C)$. In practice, we might just as well decide to use φ_y, but in this case we simply swap the variables and use the affine equation

$$f(y, x) = 0\,,$$

bringing us back to the former case and thus, justifying our assumption.

Remark 4.2.1. The number of exceptional values $n = \#\mathcal{L}$ is a huge factor in the total running time of the period matrix computation (see §4.8). By our analysis in §4.1.2, this number depends heavily on the choice of φ. A good example to illustrate this is given by the affine equation

$$y^6 = x^5 - 7x^4 - 3x^3 + 3x^2 + 13x + 20 \in \mathbb{Q}[x, y]\,.$$

We have $5 = \#\mathcal{L}(\varphi_{\mathrm{x}}) < \#\mathcal{L}(\varphi_{\mathrm{y}}) = 24$, while the degrees only differ by $d_y - d_x = 1$, so we would gain a big disadvantage if we chose φ_y here (as suggested by (4.5)). In this case it is clear that we really want to project onto the x-coordinate. Generally, finding the optimal choice here is not an easy task.

During actual computations we will not evaluate the differentials (4.4) directly, but use the MAGMA function 'ProductRepresentation' which decomposes the $a_i(x, y) \in K(C)$ into products of irreducible factors that are bivariate polynomials, i.e.

$$a_i(x, y) = \prod b_{i,j}(x, y)^{p_{j,i}} \quad \text{with} \quad b_{i,j} \in K[x, y],\, p_{j,i} \in \mathbb{Z}\,.$$

As can be seen in Example 4.2.2 below, the a_i usually share many common irreducible factors, so we save a lot of polynomial evaluations by evaluating these factors instead. Note that going against the choice of (4.5) also introduces two extra factors that have to be evaluated, namely the numerator and denominator of the rational function $\mathrm{d}y/\mathrm{d}x$ (or $\mathrm{d}x/\mathrm{d}y$ respectively).

Example 4.2.2. A running example for this chapter will be the curve defined by the affine equation $f_1 = 0$ where

$$f_1 = x^9 + 2x^6y^2 + x^2 + y^6 + 2y^4 \in \mathbb{Q}[x, y]\,.$$

The corresponding compact Riemann surface X (or, equivalently, the function field $\mathbb{C}(X)$) has genus $g = 17$. Employing our strategy we can compute a basis $\omega_1, \ldots, \omega_{17}$ of $\Omega^1(X)$ using MAGMA and represent these differentials as products of only 4 irreducible factors in $\mathbb{Q}[x, y]$, namely

$$\begin{aligned} b_1(x, y) &= x, \\ b_2(x, y) &= y^2 + 2, \\ b_3(x, y) &= y, \\ b_4(x, y) &= 2/3x^6 + y^4 + 4/3y^2 \end{aligned}$$

together with the 4×17 matrix of exponents

$$(p_{j,i})_{j,i} = \begin{pmatrix} 0 & 1 & 0 & 3 & 2 & 1 & 0 & 4 & 3 & 2 & 1 & 6 & 5 & 4 & 3 & 2 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 2 & 2 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \end{pmatrix}$$

such that the basis differentials are given by the products

$$\omega_i = \prod_{j=1}^{4} b_j(x, y)^{p_{j,i}} \mathrm{d}x, \quad i = 1, \ldots, 17. \tag{4.7}$$

Standard basis For compact Riemann surfaces there is also a standard basis of $\Omega^1(X)$ (see Theorem 2.6.2) of the form

$$\omega_i = \frac{b_i(x,y)}{\partial_y f(x,y)} \mathrm{d}x\,, \tag{4.8}$$

where the bivariate polynomials $b_i(x,y) \in \mathbb{C}[x,y]$ are of degree at most $d-3$. The b_i are called *adjoint polynomials* of degree $d-3$. There are several methods to compute these polynomials and they require a desingularization of the projective closure $\overline{C}_f$, see for example [27, Section 5] or [35, Section 4], but we did not implement any of these. Our focus on applications within number theory justifies limiting our algorithm to exact defining polynomials, i.e. polynomials which are defined over number fields, so that we can use MAGMA's function fields for the differentials. Note that the function field algorithm also computes a normalization.

Implementing an algorithm for computing this standard basis of differentials would be a big step towards a removal of the assumption on the coefficients of f being exact. The important part of our period matrix algorithm does not rely on exactness of the input, so having these differentials at our disposal would extend our algorithm to complex polynomials $f \in \mathbb{C}[x,y]$. However, allowing inexact polynomials would automatically restrict the attainable accuracy and introduce new challenges in identifying exceptional values. The authors of [35] naturally face these problems in their MATLAB implementation, as their approach is limited to double-precision floating point numbers.

Complexity Let us now consider the complexity of evaluating a vector of differentials $\bar{\omega} = (\omega_1, \ldots, \omega_g)$, that is a standard basis (4.8) for $\Omega^1(X)$, at $z \in \mathbb{C} \setminus \mathcal{L}$ and $y(z) = (y_1(z), \ldots, y_m(z))$. If we evaluate the numerators b_i and the denominator $\partial_y f(x,y)$ we see that

- evaluation of $\bar{\omega}$ at z costs up to $(g(d-3)+d_x)\mathcal{M}(D)$,
- evaluation of $\bar{\omega}$ at $y(z)$ costs up to $m(g(d-3)+m-1)\mathcal{M}(D)$
- and divisions cost $mg\mathcal{M}(D)$,

which results in the complexity

$$O(g(d\mathcal{M}(D) + md\mathcal{M}(D) + m\mathcal{M}(D))) \;=\; O(gmd\mathcal{M}(D)) \;=\; O(d^4\mathcal{M}(D))\,. \tag{4.9}$$

Evaluating the differentials using the product representation, as explained above, gives a nice speed-up in practice (we need to evaluate less polynomials of lower degrees), but for the complexity we have to be a bit careful. The number of irreducible factors is still in $O(g)$ and since $\deg(a_i) = \sum_j p_{j,i} \deg(b_j)$ we can evaluate the values $b_j(x,y)^{p_{j,i}}$ at $(z, y(z))$ using $2gmd\mathcal{M}(D)$ operations, if we assume that the degrees in x and y of the numerators and denominators of the $a_i(x,y)$ are bounded by d and m respectively. Afterwards we need $O(g^2\mathcal{M}(D))$ operations to compute the ω_i as in (4.7) and obtain a total of $O(d^4\mathcal{M}(D))$ operations, exactly as for the naive approach.

4.3 Fundamental group

Later we will describe a homology basis for X in terms of lifts of closed paths that, starting from a base point x_0, encircle exactly one exceptional value once in counterclockwise direction. These paths form a generating set of $\pi_1(\mathbb{P}^1 \setminus \hat{\mathcal{L}}, x_0)$, where $\hat{\mathcal{L}} = \mathcal{L} \cup \{\infty\}$ and $\mathcal{L} = \mathcal{L}(\varphi_{\mathrm{x}}) \subset \mathbb{C}$ is the locus of exceptional values that we want to avoid; we denote by n the

cardinality of $\mathcal{L}$. Here, we want to give an algorithm that computes such a generating set based on the approach of [74]. The ground work for our implementation of this algorithm was laid by Stefan Hellbusch.

Algorithm 4.3.1 (Generating set)**.** Computes a set of closed paths $\{\gamma_i \mid i = 1, \dots, n\}$ starting at a base point $x_0 \in \mathbb{C} \setminus \mathcal{L}$ such that γ_i encircles the exceptional value $x_i \in \mathcal{L}$ exactly once and no other exceptional values. The homotopy classes of the γ_i then generate $\pi_1(\mathbb{P}^1 \setminus \hat{\mathcal{L}}, x_0)$ as a free group.

(1) Compute a minimal (or maximal) spanning tree consisting of line segments between the points in $\mathcal{L}$ with respect to a weight function (§4.3.1).

(2) Choose a suitable base point $x_0 \in \mathbb{C} \setminus \mathcal{L}$ and connect it to the spanning tree via a line segment (§4.3.1).

(3) For each point $x_i \in \mathcal{L}$ choose a radius $r_i > 0$ and construct l_i arcs of radius r_i around x_i such that their concatenation is a full circle. Here, l_i is the number of edges in the spanning tree that are incident to x_i. The start- and endpoints of the arcs are determined by the angles of the corresponding line segments.

(4) For each $i = 1, \dots, n$ use a depth-first search with root x_0 to find the paths γ_i and express them as sequences of line segments and arcs.

(5) Order the exceptional values x_i in a way such that the closed path around infinity is given by the relation $\gamma_1 \dots \gamma_n \gamma_\infty = 1$.

One advantage of this approach is that many of the line segments and arcs can be reused for several γ_i. Constructing the paths $\gamma_1, \dots, \gamma_n$ that generate $\pi_1(\mathbb{C}\setminus\mathcal{L}, x_0) = \pi_1(\mathbb{P}^1\setminus\hat{\mathcal{L}}, x_0)$ requires

- $n - 1$ line segments for the spanning tree.
- $2(n - 1)$ arcs around the exceptional values by the degree-sum-formula,
- 1 line segment to connect the base point x_0,

which results in a total of $3n - 2 = O(n)$ paths.

Complexity We analyze the complexity of Algorithm 4.3.1 in terms of the number of exceptional values n and the precision D. Note that the spanning tree computation can be done in fixed low precision, so that depending on the weight function (see §4.3.1) it has complexity $O(n^2)$ or $O(n^3)$. Except for when n is very large (comparable to the precision D), this cost is negligible. This issue is also discussed in §5.6.4. However, the starting points of the individual paths have to be computed in full precision since these values define the parametrizations (§4.3.3) which we need to evaluate for numerical integration. For the radii (see §4.3.1) we need to know all the $n(n - 1)/2$ distances between the exceptional values. Once the radii are known, each of the connection points requires just one evaluation of a line segment. The rest of the algorithm is just sorting exceptional values and ordering the paths correctly. Hence, if the weight of an edge can be computed in $O(1)$, then Algorithm 4.3.1 has complexity

$$O(n^2 \mathcal{M}(D)) = O(n^2 D \log^{1+\varepsilon} D)\,. \tag{4.10}$$

4.3.1 Choices

There are several choices to be made in Algorithmn 4.3.1 that we are going to discuss now.

Weight function Reasonable choices for the weight function in the construction of the spanning tree include minimizing euclidean distances between exceptional values (i.e. the length of line segments) and maximizing the integration parameter r for line segments as explained in §4.7. As discussed §5.6.4, the euclidean distance is much easier to compute and gives satisfying results.

Base point There are many reasonable choices for the base point. If one wants to compute the Abel-Jacobi map of points at infinity (§4.9) using numerical integration (see §4.9.5) it can be useful to choose a real base point that lies 'left' of all exceptional values as suggested by [27, p.9], i.e. taking $x_0 \in \mathbb{R}$ such that $x_0 < \mathrm{Re}(x_i)$ for all $x_i \in \mathcal{L}$. This has the advantage that there is an obvious choice for a path to infinity (4.12) avoiding exceptional values, namely $\gamma_{\inf} : [-1, 1] \to \mathbb{P}^1$, $t \mapsto \frac{-2\,\mathrm{sgn}(x_0)x_0}{(1-t)}$. If one uses Puiseux-series for that purpose one might want to choose x_0 such that $|x_0| > |x_i|$ for all $x_i \in \mathcal{L}$. For the approach to the Abel-Jacobi map that we present in §4.9.4 it is even necessary to choose $x_0 \in K$ (without restriction we may simply choose $x_0 \in \mathbb{Z}$).

If one is only interested in period matrices, the above choices offer no advantage at all. Instead, one can just take the worst edge (think of the line segment of maximal euclidean length) in step (1) of Algorithm 4.3.1, say $e = (a, b)$, and split it in half by setting $x_0 = \frac{a+b}{2}$. If no Abel-Jacobi map is required, this is our preferred choice.

Another option would be to choose the base point after constructing arcs and circles and taking it as the start point of a full circle that corresponds to a leaf of the spanning tree. This was suggested by [37] and has the advantage of reducing the total number of paths by one. However, in some examples splitting up a line segment (§4.7.5) can be more useful than saving one integration if we measure by the total number of integration points.

Radii In step 3 of Algorithm 4.3.1 one has to choose a radius $r_i > 0$ for the arcs around x_i. In accordance with [27, p.9], we found that

$$r_i = \min \begin{cases} c \cdot \mathrm{dist}(x_i, \mathcal{L} \setminus \{x_i\}) \text{ with } c = 2/5, \\ 1/4 \end{cases} \tag{4.11}$$

is generally a good choice. Any value $c \in]0, 1/2[$ is valid in theory, but this choice has a lot of influence on the integration process (as we will see in Section 4.7). For c close to zero, arcs will require very few integration points, but analytic continuation along them will cause numerical instability and the line segments will require more points. Conversely, arcs will be harder to integrate for c close to $1/2$, while line segments become easier. Therefore, choosing a balanced value, say $c \in [1/5, 2/5]$, is recommended. Moreover, limiting the radius to $1/4$ prevents arcs and circles from having very large radius. In the case where there is an exceptional value far away from the others we rather integrate a long line segment than a circle with huge radius.

4.3.2 Alternatives

Voronoi cells One alternative, that was first used by Paul van Wamelen in his MAGMA implementation for hyperelliptic curves [93], is to integrate along line segments that are given by the edges of a Voronoi cell decomposition of the set $\mathcal{L}$ plus 6 auxiliary points, so that all exceptional values are enclosed by finite edges. This approach has also been used by Nils Bruin and Alexandre Zotine for compact Riemann surfaces given by an affine equation. One advantage here is that one only needs line segments, which are easier to evaluate than arcs. From graph theory we know that the graph corresponding to the Voronoi cells (without the infinite edges) has at least $\frac{3}{2}n$ edges. Moreover, the average

number of edges per cell in a Voronoi diagram is 6 which means that, asymptotically we have $\approx 3n$ integrals. We did not implement this approach, but we expect that there are no big differences in terms of efficiency between our method and this one.

Spanning tree Arguably the most efficient and elegant way is to directly integrate between exceptional values: it only requires $n-1$ line segments and numerical stability is greatly improved. This approach requires desingularization of the holomorphic differentials above the exceptional values. For general algebraic curves this can, to our knowledge, only be done using Puiseux series expansions above exceptional values which is something we want to avoid in this chapter, see Section 4.11. In the case of superelliptic curves (see Chapter 5), due to the special form of the differentials presented in Proposition 5.1.8, the corresponding integrands can be desingularized symbolically, making this approach the indispensable choice.

4.3.3 Types of paths & parametrizations

Numerical integration is naturally performed on the interval $[-1,1]$, so we use the following parametrizations $\gamma : [-1,1] \to \mathbb{C}$ for our paths. For complex line segments $[a,b]$ we use the parametrization

$$\gamma_{\text{ls}} : u \mapsto \frac{b-a}{2}\left(u + \frac{b+a}{b-a}\right) \quad \text{with derivative} \quad \gamma'_{\text{ls}} : u \mapsto \frac{b-a}{2}.$$

For a circle of radius $c > 0$ around x with orientation $o \in \{\pm 1\}$ whose starting point is determined by $\varphi_0 \in (-\pi, \pi]$, we use the parametrization

$$\gamma_{\text{fc}} : u \mapsto c\exp(i(\pi o(u+1) + \varphi_0)) + x$$

with derivative

$$\gamma'_{\text{fc}} : u \mapsto ci\pi o \exp(i(\pi o(u+1) + \varphi_0)).$$

For arcs of radius $c > 0$ around x whose start and end point are determined by arguments $\varphi_2, \varphi_1 \in (-\pi, \pi]$, respectively, we use the parametrization

$$\gamma_{\text{arc}} : u \mapsto c\exp\left(i\left(\frac{\varphi_2 - \varphi_1}{2}\left(u + \frac{\varphi_2 + \varphi_1}{\varphi_2 - \varphi_1}\right)\right)\right) + x$$

with derivative

$$\gamma'_{\text{arc}} : u \mapsto ci\left(\frac{\varphi_2 - \varphi_1}{2}\right)\exp\left(i\left(\frac{\varphi_2 - \varphi_1}{2}\left(u + \frac{\varphi_2 + \varphi_1}{\varphi_2 - \varphi_1}\right)\right)\right)$$

Finally, we also introduce infinite lines that start at $x \in \mathbb{C} \setminus \{0\}$ and end at $\infty \in \mathbb{P}^1$ in the direction of x via

$$\gamma_{\text{inf}} : u \mapsto \frac{2x}{(1-u)} \quad \text{with derivative} \quad \gamma'_{\text{inf}} : u \mapsto \frac{2x}{(1-u)^2}. \tag{4.12}$$

Evaluation We always evaluate paths and their derivative simultaneously (both values are needed for integration) in order to save arithmetic operations. It is easy to see that evaluating line segments costs $\mathcal{M}(D)$, while circles and arcs require evaluating an exponential and thus cost $\mathcal{T}(D)$.

Example 4.3.2. Figure 4.1 shows the paths produced by Algorithm 4.3.1 applied to the $n = 31$ exceptional values $\mathcal{L} = \mathcal{L}(f_2, y)$ of the genus 14 Riemann surface defined by

$$f_2 = 15x^5y^5 + 44x^4y + 24x^3y^3 + 15xy^4 - 49 \in \mathbb{Q}[x,y].$$

We highlighted in blue the closed path γ_{26} encircling the exceptional value x_{26}.

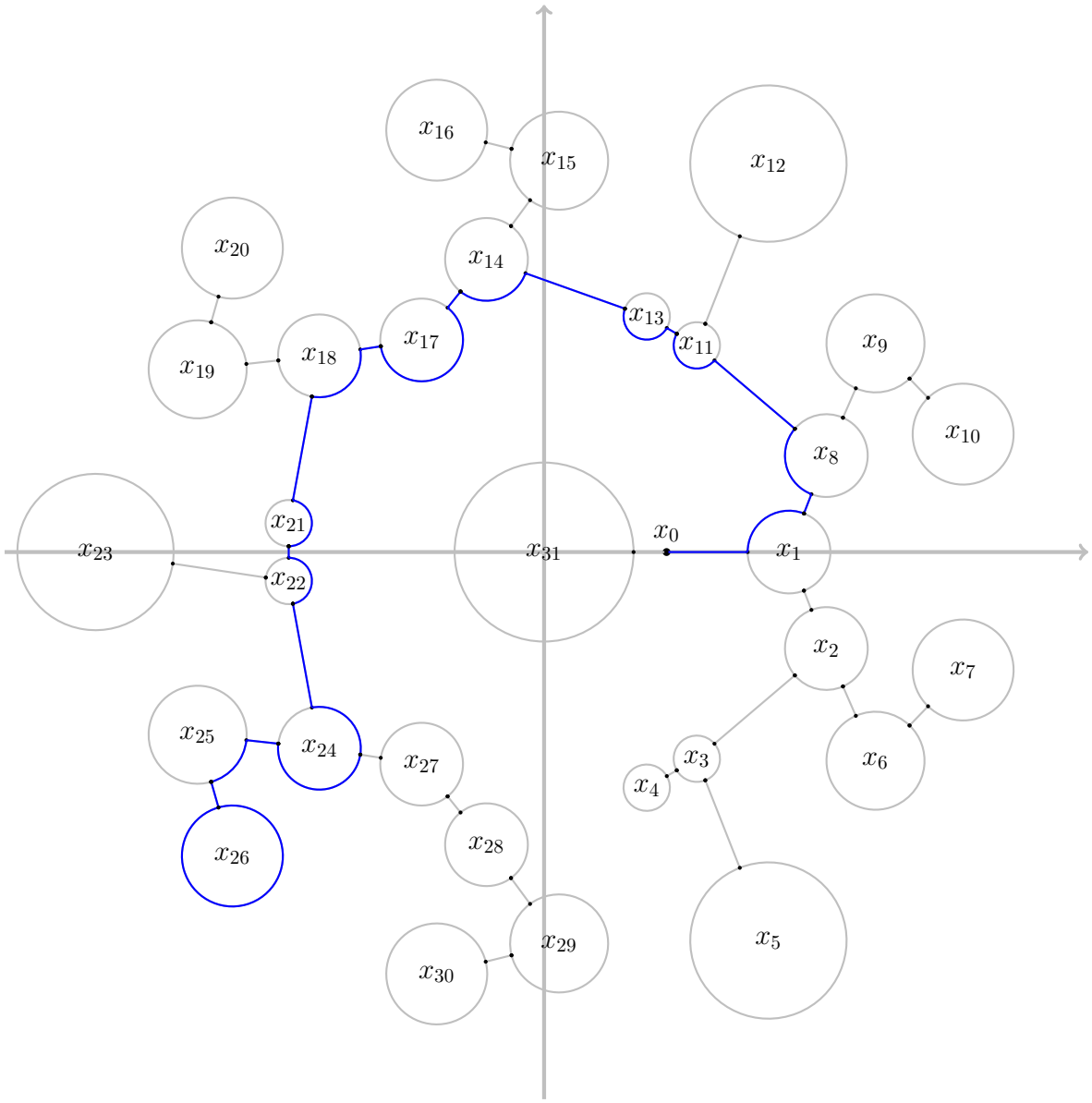


Figure 4.1: Paths for the fundamental group $\pi_1(\mathbb{C} \setminus \mathcal{L}(f_2, y), x_0)$.

4.4 Root approximation methods

In order to perform analytic continuation of algebraic functions (see §4.5) we implemented several methods that are used for simultaneous approximation of all zeros of a separable complex polynomial (given by precision D coefficients). Here, we want to briefly present and analyze three of them: *Durand-Kerner's method*, *Börsch-Supan's method* and *Börsch-Supan's method with Weierstrass corrections*. While these methods are similar to and essentially based on Newton's method, as described in §1.3, they are much more efficient if one is interested in determining all polynomial zeros. Their best properties are guaranteed conditions for convergence, rigorous error bounds and derivative-free iterative formulas. A detailed description and proofs can be found in [72, Chapter 3].

Let us start by introducing some notation. In the following $p \in \mathbb{C}[z]$ will be a monic

polynomial of degree $m \geq 3$ with simple roots $\eta_1, \ldots, \eta_m$ and respective approximations $z_1, \ldots, z_m$. Denote the index set $I_m = \{1, \ldots, m\}$ and

$$d = \min_{i,j \in I_m,\, i \neq j} |z_i - z_j|$$

the minimal distance between all approximations. For each $i \in I_m$ we define a term W_i, called *Weierstrass correction*, by

$$W_i = W_i(z_i) = p(z_i) / \prod_{j \neq i} (z_i - z_j) \quad (i \in I_m)$$

and the corresponding maximal absolute value of all Weierstrass corrections

$$w = \max_{i \in I_m} |W_i| \ .$$

Note that the methods described below solely rely on these quantities (and therefore the data $p, z_1, \ldots, z_m$) as input to approximate $\eta_1, \ldots, \eta_m$ to arbitrary precision. We start by formulating a result (see [72, Corollary 3.1]) that is central to our applications.

Theorem 4.4.1 (Corollary 3.1 in [72])**.** *Let $p \in \mathbb{C}[z]$ be a separable polynomial of degree $m \geq 3$. Under the condition*

$$w < c_m d, \tag{4.13}$$

where $c_m \leq 1/(2m)$ is a constant that depends only on m, each of the disks D_i defined by

$$D_i = \{z_i; \rho_i\} := \{\, z \in \mathbb{C} \mid |z - z_i| < \rho_i \,\} \quad \textit{with} \quad \rho_i = \frac{|W_i(z_i)|}{1 - m c_m} \quad (i \in I_m)$$

contains exactly one zero of p.

Iteratively calculating the centers of these disks $z_i^{(k)}$ by using a suitable iterative formula produces a sequence of disks

$$D_i^{(k)} = \{z_i^{(k)}; \rho_i^{(k)}\} \quad (k \geq 0)$$

whose radii $\rho_i^{(k)} = \left| W_i(z_i^{(k)}) \right| / (1 - m c_m)$ converge to 0 under some convergence conditions, thus approximating the zeros of p.

A posteriori error bound methods Assume that condition (4.13) holds for $w^{(0)}$ and $d^{(0)}$, which have been calculated from initial approximations $z_i^{(0)}$ $(i \in I_m)$. Then, an *A posteriori error bound methods* (PEB method) is defined by the sequences of disks $\{D_i^{(k)}\}$ given by

$$D_i^{(k)} = \{z_i^{(k)}; \rho_i^{(k)}\}, \quad \rho_i = \frac{|W_i(z_i^{(k)})|}{1 - m c_m}, \quad z_i^{(k+1)} = \Phi(z_i^{(k)}), \quad (i \in I_m, k \geq 0)$$

where $\Phi(z_i^{(k)})$ is an iterative formula given by either (4.14), (4.15) or (4.16). We found that these iterative formulas possess the highest computational efficiency in the context of our applications.

The Durand-Kerner method (DK)

$$\Phi(z_i^{(k)}) = z_i^{(k)} - W_i^{(k)} \quad (i \in I_m, k \geq 0) \tag{4.14}$$

The Börsch-Supan method (BS)

$$\Phi(z_i^{(k)}) = z_i^{(k)} - \frac{W_i^{(k)}}{1 + \sum_{j \neq i} \frac{W_j^{(k)}}{z_i^{(k)} - z_j^{(k)}}} \quad (i \in I_m, k \geq 0) \tag{4.15}$$

The Börsch-Supan method with Weierstrass' corrections (BSW)

$$\Phi(z_i^{(k)}) = z_i^{(k)} - \frac{W_i^{(k)}}{1 + \sum_{j \neq i} \frac{W_j^{(k)}}{z_i^{(k)} - W_i^{(k)} - z_j^{(k)}}} \quad (i \in I_m, k \geq 0) \tag{4.16}$$

Theorem 4.4.2 (Convergence of PEB methods, Theorems 3.13, 3.14, 3.15 in [72])**.**

(a) The PEB method based on the Durand-Kerner method (DK) converges quadratically if the initial condition (4.13) *holds, where* $c_m = 1/(2m)$*.*

(b) The PEB method based on the Börsch-Supan method (BS) converges cubically if the initial condition (4.13) *holds, where* $c_m = 1/(2m)$*.*

(c) The PEB method based on the Börsch-Supan method with Weierstrass' corrections (BSW) converges with order 4 *if the initial condition* (4.13) *holds, where* $c_m = 1/(2m+1)$*.*

Remark 4.4.3.

- The Durand-Kerner method also works for $m = 2$ and converges globally in that case.
- In practice, we can choose the factor c_m much larger ($c_m = 1/2$ is a good choice) if we want to allow worse initial approximations at the cost of worse convergence.

Hence, we obtain the following algorithm:

Algorithm 4.4.4 (PEB method)**.** Given a polynomial $p \in \mathbb{C}[z]$ with m simple roots and approximations $z_1^{(0)}, \ldots, z_m^{(0)}$ that satisfy the initial convergence condition (4.13), this algorithm (heuristically) approximates the roots of p to precision D.

(1) Set $k \leftarrow 0$.

(2) Repeat

(2.1) Compute Weierstrass' corrections $W_i^{(k)}$ at $z_i^{(k)}$ $(i \in I_m)$.

(2.2) Compute the radii $\rho_i^{(k)} = \left| W_i(z_i^{(k)}) \right| / (1 - mc_m)$ $(i \in I_m)$.

(2.3) Compute $\Phi(z_i^{(k)})$ using (4.14),(4.15) or (4.16) $(i \in I_m)$.

(2.4) Set $k \leftarrow k + 1$.

(2) until $\max_{i \in I_m} \rho_i^{(k)} < e^{-D}$.

(3) Return $z_1^{(k)}, \ldots, z_m^{(k)}$.

Complexity In order to analyze the complexity of Algorithm 4.4.4, suppose we want to approximate the simple zeros of a degree m polynomial up to precision D and that the convergence condition from Theorem 4.4.2 is satisfied for either of the presented PEB methods.

Similar to Newton's method, as described in 1.3, Algorithm 4.4.4 needs $O(\log D)$ iterations, but again we can successively increase the precision in each step according to the order of convergence of our chosen method (doubling for (DK), tripling for (BS), quadrupling for (BSW)). Therefore, we only need to analyze the last iterative step that has to be performed at precision D.

In [72, Table 3.5] the author states the exact numbers of operations that are required in each iterative step (as real arithmetic operations): additions & subtractions (A+S), multiplications (M) and divisions (DIV). We exhibit the data that is interesting to us:

	(A+S)	(M)	(DIV)
(DK)	$8m^2+m$	$8m^2+2m$	$2m$
(BS)	$15m^2-6m$	$14m^2+2m$	$2m^2+2m$
(BSW)	$15m^2-4m$	$14m^2+2m$	$2m^2+2m$

Table 4.1: Number of arithmetic operations for PEB methods.

We really want to minimize (DIV) first, (M) second and ignore (A+S) here. Going more into the details of Table 4.1 we find that, due to an equal number of divisions and multiplications while having a lower order of convergence, (BS) is inferior to (BSW) in all cases. So we are left with a comparison between (DK) and (BSW). We use the same estimates as in (1.3), but take into account the different orders of convergence and therefore the number of required iterations. The cost $\mathcal{F}(D)$ of one iteration is now given by Table 4.1. Assuming (DIV)$\sim$ 2(M) we estimate the costs of Algorithm 4.4.4 using each method as

$$\begin{aligned}(\text{DK}) &\leq 2\mathcal{F}(D) = 2((8m^2+2m)(\text{M})+2m(\text{DIV})) \sim (16m^2+12m)(\text{M})\\ (\text{BSW}) &\leq \frac{4}{3}\mathcal{F}(D) = \frac{4}{3}((14m^2+2m)(\text{M})+(2m^2+2m)(\text{DIV})) \sim (24m^2+6m)(\text{M})\end{aligned}$$

We conclude: Durand-Kerner (DK) is the superior root approximation method for our applications. Independently of the chosen method, Algorithm 4.4.4 has complexity in

$$O(m^2\mathcal{M}(D)) = O(m^2 D\log^{1+\varepsilon} D)\,.$$

4.5 Analytic continuation & local monodromy

In this section we denote by $y(x) = (y_1(x),\dots,y_m(x))$ the vector of algebraic functions given by

$$f(x,y_j(x)) = 0 \quad \text{for } j = 1,\dots,m = \deg_y(f)\,.$$

We already explained the connection between analytic continuation of algebraic functions and affine plane curves in §2.4.4. In order to obtain a monodromy representation of the holomorphic map $\varphi_x : X \to \mathbb{P}^1$ and for the numerical integration of differential forms it is crucial to compute $y(x) = (y_1(x),\dots,y_m(x))$ as a vector of m analytic functions, while following a path $\gamma : [-1,1] \to \mathbb{C}\setminus\mathcal{L}$ in the x-plane. By the path-lifting property, we can lift the path γ from $\mathbb{C}$ to m different paths $\tilde{\gamma} = (\tilde{\gamma_1},\dots,\tilde{\gamma_m})$ on the Riemann surface X

$$\tilde{\gamma_j} : [-1,1] \to X,\ t \mapsto (\gamma(t), y_j(\gamma(t))) \quad j = 1,\dots,m\,. \tag{4.17}$$

Generally, this can not be done symbolically (it can be done symbolically for superelliptic curves, see §5.1.2). We represent the functions in $y(x)$ by approximation on a finite set of points lying on the path γ instead. In practice, this set will be the abscissas

$$\{u_\ell\}_{\ell=1,\dots,N} \subset [-1,1]$$

that are required for numerical integration along γ as explained in Section 4.7. Thus, we define the *discrete lifts* of γ as the finite set

$$\tilde{\gamma_j}(N) := \{\, \tilde{\gamma_j}(u_\ell) \mid \ell = 1,\dots,N \,\} \subset \tilde{\gamma_j}\,. \tag{4.18}$$

Suppose we are given two points $x_1, x_2 \in \gamma([-1,1])$ and the correct fiber $y(x_1) = (y_1(x_1),\dots,y_m(x_1)) \in \mathbb{C}^m$. Correct here means that the values correspond to analytic continuations of $y(x)$ along γ. We move from $y(x_1)$ to the correct $y(x_2)$ using the root approximation methods presented in Section 4.4. The key here is to use the correct fiber $y(x_1)$ as approximation to the m simple roots of the univariate polynomial $f(x_2, y)$. By Theorem 4.4.1 we can successively define analytic functions $y_j(x)$ this way: we apply it to the polynomial $f(x_2, y)$ with approximations $z_j = y_j(x_1)$ for $j \in I_m$. Suppose condition (4.13) is satisfied, then we have that each value $y_j(x_1)$ ($j \in I_m$) converges to a unique root of $f(x_2, y)$. For each $j \in I_m$, defining $y_j(x_2)$ as that unqiue root, we obtain

$$|y_j(x_1) - y_j(x_2)| \overset{4.4.1}{<} \frac{|W_i(z_i)|}{1 - mc_m} \le \frac{w}{1 - mc_m} < \frac{c_m d}{1 - mc_m} \le \frac{d}{m} \le \frac{d}{2} \tag{4.19}$$

which, by [53, Remark 2.2], is a sufficient criterion for analytic continuation. Therefore, as long as the condition (4.13) is satisfied, we can uniquely continue our functions $y_j(x)$ from x_1 to x_2 for all $j \in I_m$ simultaneously.

Algorithm 4.5.1 (Analytic continuation)**.** Given an affine plane curve $C_f : f = 0$ to precision $\tilde{D} > D$ (see §4.10.1), a path $\gamma : [-1,1] \to \mathbb{C}\backslash\mathcal{L}$ in the x-plane avoiding exceptional values and a set of abscissas $\{u_l\}_{1 \le l \le N} \subset [-1,1]$ where $N > 2$, this algorithm (heuristcally) computes the discrete lifts $\tilde{\gamma_j}(N)$ ($j \in I_m$) of γ, as defined in (4.18), to precision $D > 0$.

(1) Set $u_0 \leftarrow -1$, $u_{N+1} \leftarrow 1$.

(2) Set $x_0 \leftarrow \gamma(u_0)$. Compute the ordered fiber $y(x_0)_{>_X} = (y_1(x_0),\dots,y_m(x_0))$ by solving $f(x_0, y) = 0$ in fixed low precision and order the roots with respect to $>_X$ (see §4.1.3).

(3) For $l = 1,\dots,N+1$ do

- **(3.1)** Set $x_l \leftarrow \gamma(u_l)$.
- **(3.2)** If (4.13) is satisfied for $p(y) = f(x_l, y)$ and $z_j = y_j(x_{l-1})$ ($j \in I_m$), then use Algorithm 4.4.4 to compute approximations $y_j(x_l)$ ($j \in I_m$) of the roots of $f(x_l, y)$ with $\varepsilon < e^{-D}$.
- **(3.3)** Otherwise, introduce an intermediate step by setting $u_l' = \frac{u_l + u_{l-1}}{2}$ and go to step (3.1) with $l = l'$.

(4) Return $\tilde{\gamma}(N) = \{\, (x_l, y(x_l)) \mid l = 1,\dots,N \,\}$.

Complexity For the complexity analysis of Algorithm 4.5.1 we assume that $\tilde{D} - D = O(1)$. The cost $\mathcal{R}(m, D')$ for finding the roots of $f(x_0, y)$ can be neglected as it can be done in fixed low precision $D' < D$. We count the following significant operations, each of which has to be executed $N + N_r$ times where N_r is the, a priori unknown, number of necessary refinements, see step (3.3). Note that the refinement steps are also independent of D since they can be performed in fixed low precision D'. We have that

- evalutation of γ at u_l has complexity $\mathcal{M}(D)$ for line segments and $\mathcal{T}(D)$ for arcs and circles,
- evaluation of $f(x_l, y)$ at $x_l = \gamma(u_l)$ takes $O(m\mathcal{M}(D))$ operations,
- while approximating the roots of $f(x_l, y)$ to precision D, using Algorithm 4.4.4, has complexity $O(m^2\mathcal{M}(D))$.

Hence, we obtain the following

Theorem 4.5.2. *Computing the discrete lifts* (4.18) *of γ to the Riemann surface X to precision D using Algorithm 4.5.1, has complexity*

$$O(N(\log D + m^2)\mathcal{M}(D))\,. \tag{4.20}$$

Proof. The number of operations that cannot be performed in fixed precision is given by $N(\log D\mathcal{M}(D) + m\mathcal{M}(D) + m^2\mathcal{M}(D))$. □

The complexity (4.20) is reduced to $O(Nm^2\mathcal{M}(D))$ when γ is a line segment. Explicit timings for Algorithm 4.5.1 can be found in the Tables 4.4 & 4.5.

Remark 4.5.3. We clearly see from the above analysis that the cost of evaluating arcs and circles increases the computational cost of Algorithm 4.5.1. In practice, this is barely noticeable and the running time is strictly dominated by Algorithm 4.4.4. Note that while evaluating $\gamma(u_l)$, we also obtain $\gamma'(u_l)$ at almost no additional cost (see §4.3.3) and these values are required for numerical integration (4.24).

Remark 4.5.4. Usually, the number of connection points N is governed by the integration scheme that we want to apply and it is much bigger than the minimal number of connection points that are required for sucessful analytic continuation. In [53], Stefan Kranich gives an algorithm that performs analytic continuation while choosing steps of maximal size, thus using the minimal number of connection points. Adrien Poteaux gives two asymptotic statements about the number of connection points in [73].

Example 4.5.5 (Limitations)**.** Algorithm 4.5.1 works well for a wide range of examples. Yet, it is not that hard to find (admittedly extreme) examples where the algorithm gets lost in recursions, i.e. N_r becomes extremely large. This is due to our condition (4.13) for certified analytic continuation being satisfied only for tiny step lengths. Even if we weaken (4.13) by using $c_m = 1/2$, the algorithm does not terminate within reasnoable time and if the condition is taken away entirely, the algorithm outputs false data and analytic continuation failed. Take for example the genus 54 Riemann surface $X : f_3 = 0$ with

$$\begin{aligned} f_3 = {} & x^9y^2 + 7x^9y + 6x^8y^6 - 6x^8y^4 - 7x^7y - x^6y^5 - 9x^6y^4 \\ & - 7x^6y^3 - 4x^6y^2 + 6x^5y^6 + x^5y^4 + 3x^5 + 5x^4y^8 - 5x^4y^6 \\ & + x^4y^5 + 8x^4y^4 + 5x^4y^3 + 5x^4y^2 - 6x^3y^6 + 3x^3y^4 - 4x^3y^3 \\ & + 7x^3 + x^2y^9 + 3x^2y^6 - 6x^2y^4 + 7x^2y - 7xy^9 - 7xy^7 \\ & + 8xy^2 - 2xy + 3y^5 - 9y^3\,. \end{aligned}$$

There is an exceptional value at -173554.58, the next closest one being at -1.61. Our fundamental group algorithm places the base point in the middle and constructs a line segment $[-173554.33, -86778.09]$ which Algorithm 4.5.1 cannot handle. In such cases one could try to find a different equation for X, one that admits a simpler configuration of exceptional values.

In contrast to Example 4.5.5 above, the convergence condition (4.13) can in fact be left out for more moderate examples (e.g. all other examples considered in this work) if one is not interested in obtaining a certified analytic continuation. In particular, in §4.9.5 we will make use of this to obtain the Abel-Jacobi map above exceptional values using adaptive double-exponential integration.

4.5.1 Monodromy representation

As before, let $\gamma : [-1, 1] \to \mathbb{C} \setminus \mathcal{L}$ be a path that avoids exceptional values. After applying Algorithm 4.5.1 we may assign a permutation $\sigma_\gamma \in \mathrm{Sym(m)}$ to γ by identifying the continued fiber above the endpoint $y(\gamma(1))$ with the ordered fiber $y(\gamma(1))_{>_X}$. For the identification we compare the numerical values up to precision D. These permutations behave multiplicatively for concatenation of compatible paths, i.e.

$$\gamma = \gamma_1 \cdot \dots \cdot \gamma_s \;\Rightarrow\; \sigma_\gamma = \prod_{i=1,\dots,s} \sigma_{\gamma_i} .$$

Definition 4.5.6 (Local monodromy)**.** If γ is a closed path based at $x_0 \in \mathbb{C}$, encircling one and only one exceptional value x (see Section 4.3), then $\sigma_x := \sigma_\gamma$ is the called *local monodromy* at x. The local monodromy σ_x depends on the the ordering $>_X$ and the base point x_0.

After ordering the branch points correctly, the local monodromy at infinity is given by

$$\sigma_\infty = \left(\prod_{x \in \mathcal{L}} \sigma_x \right)^{-1} .$$

Now the *branch points* $\mathcal{B}$ of φ_{x} can be characterized as the points $x \in \mathbb{P}^1$ with non-trivial local monodromy, i.e.

$$\mathcal{B} = \{\, x \in \hat{\mathcal{L}} \mid \sigma_x \neq \mathrm{id} \,\} .$$

The ramification index $e_P(\varphi_{\mathrm{x}})$ of a point P lying over x can be read off the local monodromy: it is the length of the corresponding cycle in the cycle decomposition of σ_x. Once we have computed a generating set $\{\gamma_1, \dots, \gamma_n\}$ for $\pi_1(\mathbb{P}^1 \setminus \hat{\mathcal{L}}, x_0) = \pi_1(\mathbb{C} \setminus \mathcal{L}, x_0)$ using Algorithm 4.3.1, we can analytically continue every closed path in γ_i (by applying Algorithm 4.5.1) to obtain a *monodromy representation* of the covering map $\varphi_{\mathrm{x}} : X \to \mathbb{P}^1$, namely

$$\mathrm{Mon}(\varphi_{\mathrm{x}}) = \{\, \sigma_x \mid x \in \mathcal{B} \,\} .$$

This representation depends on the choice of ordering $>_X$ and on the base point x_0, but it is unique up to simultaneous conjugation. The permutation group generated by $\mathrm{Mon}(\varphi_{\mathrm{x}})$ is called the *monodromy group*.

Testing Suppose we already know the genus of the Riemann surface. After computing a monodromy representation $\mathrm{Mon}(\varphi_{\mathrm{x}})$ we also know the ramification indices $e_P(\varphi_{\mathrm{x}})$ for all $P \in X$ such that $\varphi_{\mathrm{x}}(P) \in \mathcal{B}$. A strong test for the correctness of our computation, which is done in practice, is checking the Riemann-Hurwitz formula (2.3).

Example 4.5.7. We consider an example that admits an interesting monodromy representation of φ_x, namely the Riemann surface $X : f_4 = 0$ defined by the polynomial

$$f_4 = x^6y^6 + x^3 + y^2 + 1 \in \mathbb{Q}[x, y]. \tag{4.21}$$

The genus of X is 11, there are $n = 16$ exceptional values and $m = 6$ sheets. We compute the corresponding fundamental group, as shown by Figure 4.2 Afterwards, we apply Algorithm 4.5.1 for analytic continuation to obtain a monodromy representation for φ_{x} (as described in §4.5.1), which is displayed in Table 4.2.

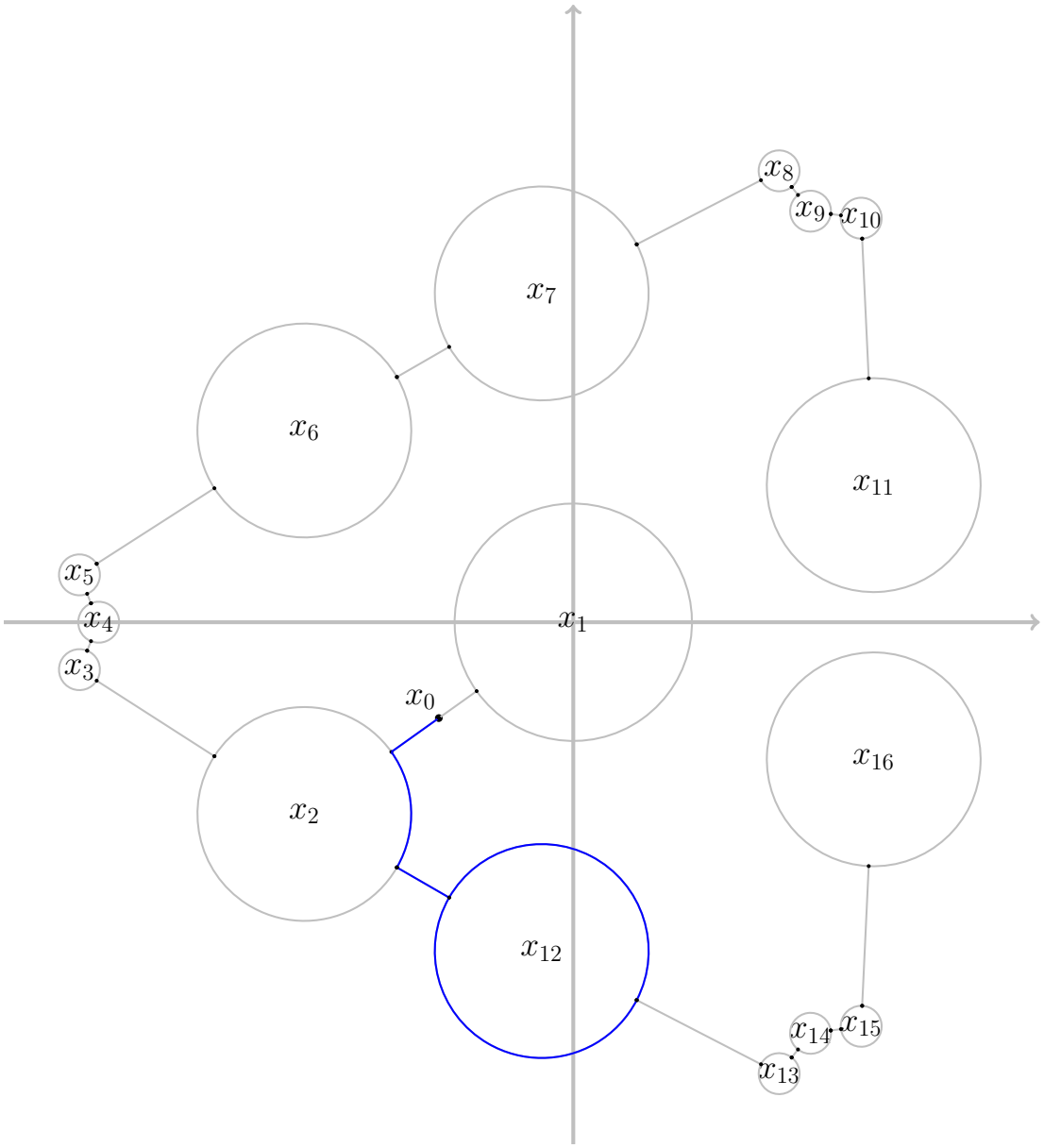


Figure 4.2: Paths for the fundamental group $\pi_1(\mathbb{P}^1 \setminus \mathcal{B}, x_0)$ (see Table 4.2).

Example 4.5.8. (cont.) When it comes to analytic continuation and monodromies, it can be quite enlightening to work with pictures. In Figure 4.3 we visualize the $m = 6$ lifts of γ_{12} to the Riemann surface where different colors indicate the sheets of X, namely

$$I_m = \{ 1, 2, 3, 4, 5, 6 \}$$

and γ_{12} itself is displayed in grey. The local monodromy $\sigma_{x_{12}} = (1,4)(3,6)$ at $x_{12} = -0.06646105747 - 0.6927249016 \cdot I$ is perfectly illustrated by our picture: while the lifts on the 2nd and the 5th sheet are closed paths on X that are homotopic to zero in $\pi_1(X)$, the lifts running on the sheets 1,4 (and 3,6 respectively) overlap because these sheets are glued together around x_{12}.

Figure 4.4 shows the lifts of the closed path γ_7. Since this path consists of many different pieces, the situation is more involved (compared to γ_{12}) and overlapping colours keep us from seeing things as clearly as in the previous picture. Nonetheless, we can get a better idea of how the vector of analytic continuations $y(x)$ behaves in this example.

x_0	$-0.2833434170 - 0.2019597075 \cdot I$
$\mathcal{B}$	σ
$x_1 = 0.0000000000$	$(1,6)(2,5)$
$x_2 = -0.5666868339 - 0.4039194149 \cdot I$	$(2,3)(4,5)$
$x_3 = -1.040501261 - 0.09982723868 \cdot I$	$(1,2)(5,6)$
$x_4 = -1.000000000$	$(1,6)$
$x_5 = -1.040501261 + 0.09982723868 \cdot I$	$(1,4)(3,6)$
$x_6 = -0.5666868339 + 0.4039194149 \cdot I$	$(2,3)(4,5)$
$x_7 = -0.06646105747 + 0.6927249016 \cdot I$	$(1,2)(5,6)$
$x_8 = 0.4337977057 + 0.9510141438 \cdot I$	$(1,4)(3,6)$
$x_9 = 0.5000000000 + 0.8660254038 \cdot I$	$(3,4)$
$x_{10} = 0.6067035550 + 0.8511869051 \cdot I$	$(2,3)(4,5)$
$x_{11} = 0.6331478914 + 0.2888054867 \cdot I$	$(1,2)(5,6)$
$x_{12} = -0.06646105747 - 0.6927249016 \cdot I$	$(1,4)(3,6)$
$x_{13} = 0.4337977057 - 0.9510141438 \cdot I$	$(2,4)(3,5)$
$x_{14} = 0.5000000000 - 0.8660254038 \cdot I$	$(3,4)$
$x_{15} = 0.6067035550 - 0.8511869051 \cdot I$	$(1,4)(3,6)$
$x_{16} = 0.6331478914 - 0.2888054867 \cdot I$	$(2,4)(3,5)$
∞	$(1,6)(2,5)(3,4)$

Table 4.2: Monodromy representation of φ_{x} induced by $f_4 = 0$ (4.21).

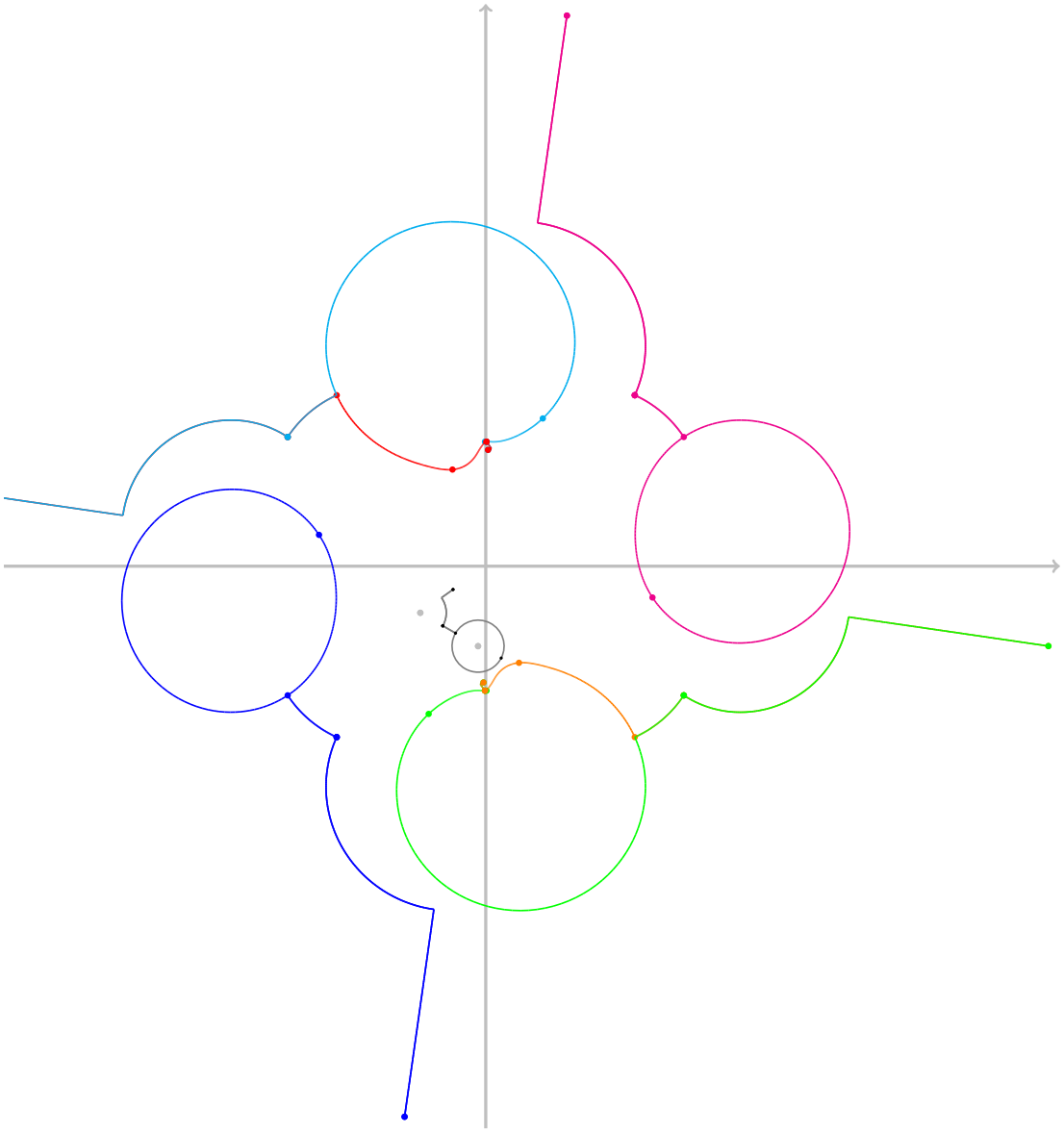


Figure 4.3: Visualization of the lifts of γ_{12} (the closed path around x_{12}) via analytic continuation.

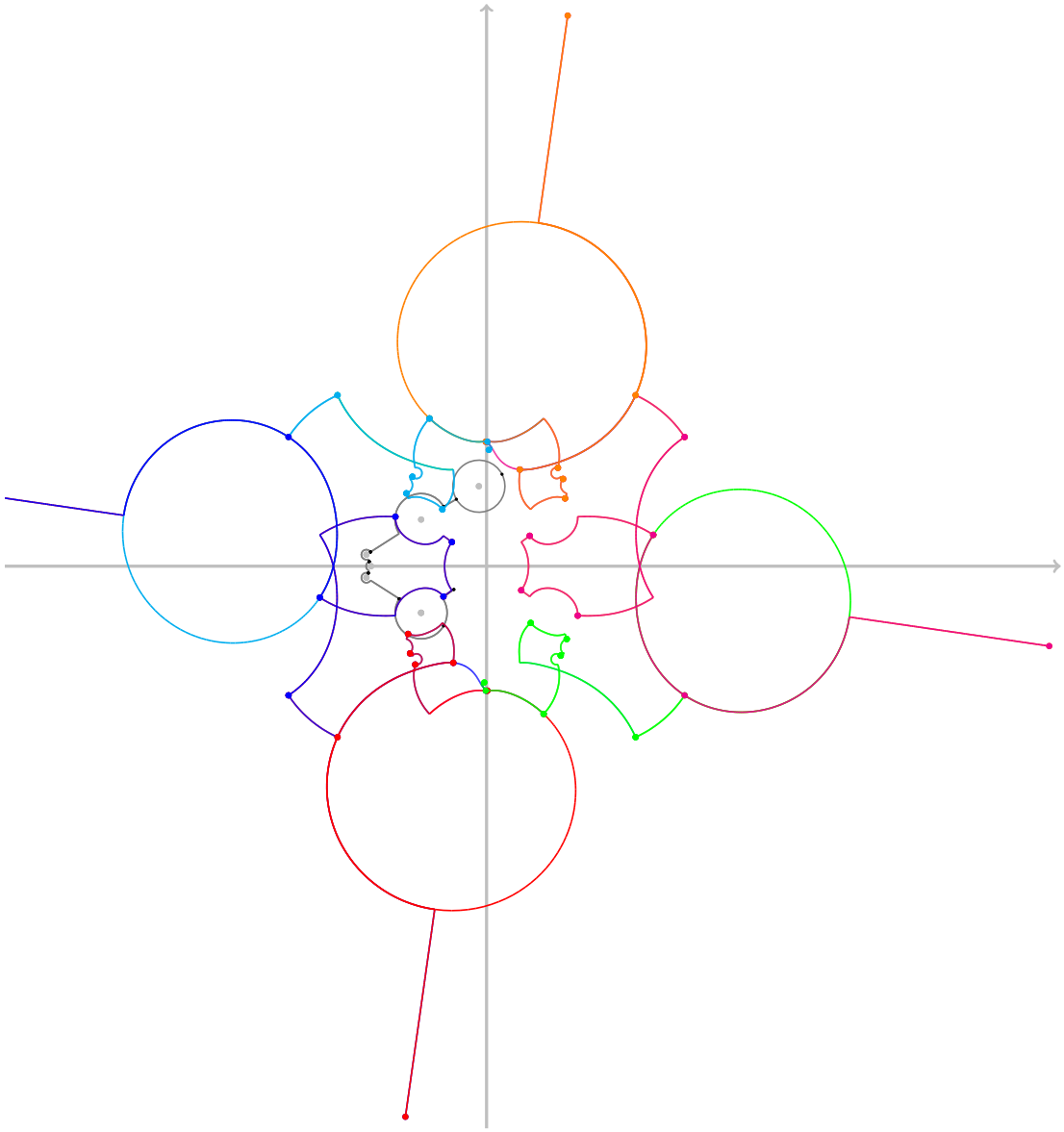


Figure 4.4: Visualization of the lifts of γ_7 (the closed path around x_7) via analytic continuation.

4.6 Computing a homology basis

By its definition, a big period matrix requires integration along a canonical basis of the homology group. From Definition 2.7.3 we recall that such a basis consists of two families of independent cycles

$$\{\alpha_i, \beta_i\}_{1 \le i \le g}$$

that generate $H_1(X, \mathbb{Z})$ and such that their intersection numbers satisfy

$$\alpha_i \circ \alpha_j = \beta_i \circ \beta_j = 0, \quad \alpha_i \circ \beta_j = \delta_{ij}.$$

A canonical homology basis for a genus 3 Riemann surface was already shown in Figure 1.1. The task of computing such a basis is split up into two parts. First we use our implementation of the Tretkoff algorithm to compute generators of the homology group and the corresponding intersection matrix, then we compute a symplectic base change matrix.

4.6.1 The Tretkoff algorithm

In [88] Tretkoff & Tretkoff give an ingenious algorithm that computes a set of generators Γ for the first integral homology group $H_1(X, \mathbb{Z})$ of a compact Riemann surface X, as well as the corresponding intersection matrix K_Γ.

The input of this algorithm is a monodromy representation

$$\mathrm{Mon}(\varphi_{\mathrm{x}}) = \{\, \sigma_x \mid x \in \mathcal{B} \,\}$$

of the holomorphic ramified covering $\varphi_{\mathrm{x}} : X \mapsto \mathbb{P}^1$, see §4.5.1. After specifying a base point $P_0 \in X$ lying on sheet $s \in \{1, \ldots, m\}$ (naturally we choose the first sheet $s = 1$ with respect the ordering $>_X$) it computes the stabilizer of s as a subgroup of the free group generated by $\mathrm{Mon}(\varphi_{\mathrm{x}})$ (see §2.5.2).

The algorithm iteratively creates a planar graph G, a tree in fact, level by level, starting with the vertex labeled $s = 1$ on level 1. The vertices on the even levels correspond to sheets $s \in \{1, \ldots, m\}$ and the odd levels correspond to ramification points, say P lying over $x \in \mathcal{B}$. They are represented by the corresponding element, say $\tau_P \in \mathrm{Sym(m)}$, in the cycle decomposition of the local monodromy σ_x.

On the odd levels the algorithm creates an edge connecting sheet s to a ramification point P precisely when $s \in \tau_P$ and P did not occur in this branch of the graph before. On the even levels an edge connecting P to a sheet s is created if $s \in \tau_P$ and s did not already occur in this branch of the graph. By branch of the graph we mean a sequence of edges that traces a sheet s or a ramification point P back to the root of the graph (which is $s = 1$).

After each level, the branches whose current end sheet (resp. ramification point) occurs more than once in the tree are labeled terminated and will not be considered for the next level. This process is repeated until all branches are terminated. After each level and after all branches are terminated the edges have to be sorted in a particular manner.

The graph constructed in this way has exactly $4g+2m-2$ branches. Every branch can be identified with exactly one other branch, creating $r := 2g+m-1$ cycles on X that start on sheet 1 and return to sheet 1 while encircling at least two different ramification points. The intersection pairing between these cycles can be read off the constructed graph by transversing through the final edge of each branch in clockwise direction.

The author finds that the details of the algorithm, especially the sorting, cannot be explained very well in text. Consequently, we included the code of our MAGMA implementation of the Tretkoff algorithm in the appendix, see Section A.2.

Our implementation of the Tretkoff algorithm follows the outline of Frauendiener and Klein [35, Section 7] who give a very hands-on explanation of the algorithm. Another helpful description of the algorithm can be found in [27, Section 4].

The first output is a set of cycles $\Gamma = \{\Gamma_1, \ldots, \Gamma_r\}$. These cycles are encoded as sequences of integers $s_{i_k} \in \{1, \ldots, m\}$ and branch points $x_{i_k} \in \mathcal{B}$

$$\Gamma_i = [\, s = s_{i_1}, x_{i_1}, s_{i_2}, x_{i_2}, s_{i_3}, \ldots, s_{i_{k-1}}, x_{i_k}, s_{i_k} = s \,].$$

If we interpret Γ_i as the path on X that is obtained by moving from sheet s_{i_j} to sheet $s_{i_{j+1}}$ by encircling the ramification point lying over x_{i_k} the correct number of times for each $j = 1, \ldots, k$, then Γ_i defines a non-trivial cycle in $H_1(X, \mathbb{Z})$.

In particular, the algorithm produces exactly $r = 2g+m-1$ of these cycles and Tretkoff & Tretkoff [88] show that they generate the fundamental group $\pi_1(X, P_0)$ and therefore their classes generate the homology group $H_1(X, \mathbb{Z})$. Since the homology group has rank $2g$, there must be exactly $m - 1$ dependent cycles in Γ. While these dependent cycles do not contribute to the period matrix computation, they we can be used to verify the quality of our numerical computations (see §4.8.1).

The second output of the algorithm is an intersection matrix

$$K_\Gamma = (\Gamma_i \circ \Gamma_j) \in \mathbb{Z}^{r \times r}$$

with entries in $\{-1, 0, 1\}$ that is skew-symmetric, i.e. satisfies $K_\Gamma + K_\Gamma^T = 0$, and has rank $2g$.

Complexity The Tretkoff algorithm is purely combinatorial and extremely fast in practice. Its complexity does not depend on the precision D and the only numbers appearing are very small integers of size at most $r+m = O(g+m)$. The algorithm mainly consists of checking for equality of such integers, which can be done using $O(\log(g+m))$ operations. Consequently, we will analyze the complexity of the algorithm in terms of m and g, which are themselves related to the total degree d.

Let us now analyze the complexity of our implementation in terms of equality checks. For this we assume that searching a list of length r can be done using $O(r)$ equality checks using *linear search*, while we can sort a list of length r using $O(r^2)$ equality checks with *quick sort*.

- The number of branches of the tree, which is also the maximal number of edges per level, is exactly $2r = 4g + 2m - 2 = O(g + m)$.
- The number of levels is bounded by $2m = O(m)$.
- On even levels: for each edge ($\leq 2r$) and each length of a cycle ($\leq m$) we have to search a list of length at most $2m$, resulting in a total of $O(rm^2) = O((g + m)m^2)$.
- On odd levels: for each edge ($\leq 2r$) and each ramification point ($\leq r$) we have to search two lists of size at most $2m$, resulting in a total of $O(r^2m) = O((g + m)^2m)$.
- After each level ($\leq 2m$) we sort the list of terminated edges of length at most $2r$, resulting in a total of $O(mr^2) = O((g + m)^2m)$.
- For the intersection matrix K_Γ we first have to order the terminated edges by searching r lists of length r and then compute the correct coefficients which requires checking equality at most r^2 times, resulting in a total of $O(r^2) = O((g + m)^2)$.

Theorem 4.6.1. *Given a monodromy representation* $\mathrm{Mon}(\varphi_x)$*, our implementation of the Tretkoff algorithm (given in Section A.2), computes a homology basis* Γ *with corresponding intersection matrix* K_Γ *using*

$$O((g+m)^2 m \log(g+m))$$

operations. If $\mathrm{Mon}(\varphi_x)$ *is induced (as described in §4.5.1) by an irreducible polynomial* $f(x,y) \in \mathbb{C}[x,y]$ *of total degree* d *with* $m = \deg_y(f)$*, then the complexity in* d *is*

$$O(d^5 \log d)\,.$$

Proof. The first complexity is obtained from adding together all the aforementioned complexities. We require a total of $O((g+m)^2 m)$ equality checks that require $O(\log(g+m))$ operations each. For the second complexity we use $m = O(d)$ and $g = O(d^2)$. □

Example 4.6.2. We consider the simple example of a genus 3 superelliptic Riemann surface $X : f_5 = 0$ given by the affine equation

$$f_5 = y^3 - (x-2)(x-1)(x-(1+I))(x-(1-I))\,. \tag{4.22}$$

Applying the algorithms of the previous sections we obtain the monodromy representation:

x_0	$-I/2$
$\mathcal{B}$	σ
$x_1 = -1$	$(1,3,2)$
$x_2 = -2I$	$(1,3,2)$
$x_3 = -1-I$	$(1,3,2)$
$x_4 = 1+I$	$(1,3,2)$
∞	$(1,2,3)$

Table 4.3: Monodromy representation of φ_x induced by $f_5 = 0$.

Feeding the monodromy representation $\{\sigma_1, \ldots, \sigma_5\}$ to our implementation of the Tretkoff algorithm results in an output of $r = 8$ cycles

$$\begin{aligned} \Gamma_1 &= [1,2,3,1,1], \quad \Gamma_2 = [1,2,2,1,1], \\ \Gamma_3 &= [1,3,3,1,1], \quad \Gamma_4 = [1,3,2,1,1], \\ \Gamma_5 &= [1,4,3,1,1], \quad \Gamma_6 = [1,4,2,1,1], \\ \Gamma_7 &= [1,5,2,1,1], \quad \Gamma_8 = [1,5,3,1,1], \end{aligned}$$

such that $H_1(X,\mathbb{Z}) = \langle\, \Gamma_1, \ldots \Gamma_8 \,\rangle$, as well as the intersection matrix

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

In Chapter 5 we will approach the homology of superelliptic curves more explicitly. In particular, a generating set is given by Theorem 5.1.6 and the corresponding intersection matrix by Theorem 5.3.1. The approach presented here is completely general and will work for any monodromy representation of a holomorphic ramified covering $\varphi : X \to \mathbb{P}^1$ where X is a compact Riemann surfaces.

4.6.2 Symplectic reduction

The Tretkoff algorithm provides us with a generating set Γ for the homology and the corresponding intersection matrix K_Γ, but there is no reason why it should be a canonical basis in the sense of Definition 2.7.3. Since the intersection pairing on Γ is a non-degenerate skew-symmetric bilinear form, we can apply our implementation of an algorithm, due to Frobenius [38, Section 7], that computes a symplectic basis for K_Γ over $\mathbb{Z}$, i.e. it returns a unimodular base change matrix $S \in \mathbb{Z}^{r\times r}$ such that

$$S^T K_\Gamma S = J, \quad \text{where} \quad J = \begin{pmatrix} 0 & I_g & 0 \\ -I_g & 0 & 0 \\ 0 & 0 & 0_{m-1} \end{pmatrix}.$$

The linear combinations given by the first $2g$ columns of $\Omega_\Gamma \cdot S$ then correspond to a canonical homology basis. On the side of period matrices this translates to

$$(\Omega_A, \Omega_B, 0_{m-1}) = \Omega_\Gamma \cdot S$$

where $(\Omega_A, \Omega_B) \in \mathbb{C}^{g\times 2g}$ is a big period matrix and the $m-1$ zero columns correspond to the dependent cycles in Γ and contribute nothing. In practice however, we can verify the quality of the results from numerical integration by asserting that the absolute values of these entries differ from zero at most by e^{-D}.

Example 4.6.3. (cont.) Applying the algorithm for symplectic reduction to the homology basis of Example 4.6.2 yields the following base change matrix

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

The approach of obtaining a canonical basis from symplectic reduction of the intersection matrix is fairly standard and has already been used by [27] and [35]. Except for the case of hyperelliptic curves where one can always use the Mumford basis, we are not aware of other possibilities to solve this problem.

Complexity The algorithm for symplectic reduction is a classical matrix algorithm and runs in $O(g^3)$ time. Knowing that our input, the intersection matrix K_Γ, has entries in $\{0, \pm 1\}$, we obtain a nice speed-up by restricting our implementation to such matrices (instead of allowing general skew-symmetric matrices with integer coefficients). Moreover, due to the special form of K_Γ, all the coefficients appearing in the reduction algorithm are tiny integers in practice. Unfortunately, we have no proof of this fact. In theory the reduction algorithm could produce a dense base change matrix S with entries of size $O(g)$ which may cause precision issues. This is also discussed in §4.8.1.

4.7 Numerical integration

The integrals we are interested in are of the form

$$\int_{\tilde{\gamma}} \omega = \int_{\tilde{\gamma}} a(x,y)\mathrm{d}x = \int_{-1}^{1} a(\tilde{\gamma}_j(u))\gamma'(u)\mathrm{d}u \tag{4.23}$$

where $\omega \in \Omega^1(X)$, $a(x,y) \in \mathbb{C}(X)$, $\gamma : [-1,1] \to \mathbb{C} \setminus \mathcal{L}$ is a parametrizable path in the x-plane (avoiding exceptional values) and

$$\tilde{\gamma}_j(u) = (\gamma(u), y_j(\gamma(u))) \quad \text{with} \quad j \in \{1, \dots, m\}$$

is a lift of γ to the Riemann surface X.

Since ω is a holomorphic differential, the poles of the meromorphic function $a(x,y)$ are exactly the critical points of φ_{x} (see §4.1.2) whose x-coordinates, the critical values, occur as a subset of the exceptional values $\mathcal{L}$. This can be used to determine the area of holomorphicity ε_r (resp. Z_r) such that we can apply the theorems from Chapter 3 here.

Note that the value of r, as computed in §4.7.2 and §4.7.3, depends on the path γ and the configuration of exceptional values $\mathcal{L}$. As the statements in Chapter 3 show, r scales the number of required abscissas linearly and has a huge influence on the integration process. The value of M from §4.7.2 (resp. M_1,M_2 from §4.7.3) does not only depend on r, but on the basis of differentials $\bar{\omega} = (\omega_1, \dots, \omega_g)$.

For the complexity analysis in terms of the required precision D in §4.7.4, we will ignore these dependencies and treat r and M (resp M_1, M_2) as constants.

When we talk about integration of differential forms in this section, we mean: given a path γ and a basis of holomorphic differentials $\bar{\omega}$ we seek to simultaneously approximate a $(g \times m)$-matrix of integrals by a matrix consisting of finite sums

$$\begin{aligned} \int_\gamma \bar{\omega} &:= \left(\int_{-1}^{1} a_i(\tilde{\gamma}_j(u))\gamma'(u)\mathrm{d}u \right)_{\substack{1 \le i \le g \\ 1 \le j \le m}} \\ &\approx \left(\sum_{l=1}^{N} w_\ell a_i(\tilde{\gamma}_j(u_\ell))\gamma'(u_\ell) \right)_{\substack{1 \le i \le g \\ 1 \le j \le m}} \end{aligned} \tag{4.24}$$

where the weights w_ℓ and abscissas u_ℓ are given by a suitable integration scheme as presented in Chapter 3. Suitable here means either Gauss-Legendre (GL), Clenshaw-Curtis (CC) or double-exponential (DE) (with $\alpha = 1$) integration.

We define the approximation error $E(\gamma, \bar{\omega}, N)$ as the maximum of all approximation errors that occur for the individual integrals, i.e. for the entries of $\int_\gamma \bar{\omega}$.

4.7.1 Concatenation

It is absolutely crucial that once we have computed the integral matrix (4.24) for several individual paths, we also know the integrals for every path that is obtained through concatenation. Assume that $\gamma_1, \dots, \gamma_k$ are compatible paths in the plane and that we have computed the integral matrix (4.24) for each of them. Since this computation requires analytic continuation (§4.5), we also know the corresponding permutations $\sigma_{\gamma_1}, \dots, \sigma_{\gamma_k}$ (§4.5.1). Moreover, contour integration on Riemann surfaces respects concatenation, so we have

$$\int_{\gamma_1 \cdots \gamma_k} \bar{\omega} = \int_{\gamma_1} \bar{\omega} + \sum_{j=2}^{k} \left(\prod_{i=2}^{j} \sigma_{\gamma_{i-1}} \right) \int_{\gamma_j} \bar{\omega} \tag{4.25}$$

where $\sigma \in \mathrm{Sym(m)}$ acts on the matrix $\int_\gamma \bar{\omega}$ by permuting its columns. Similarly, if we denote by $-\gamma$ the path obtained from reversing orientation, letting the inverse permutation act on the columns yields the corresponding integral matrix

$$\int_{-\gamma} \bar{\omega} = -\sigma_\gamma^{-1} \int_\gamma \bar{\omega}\,. \tag{4.26}$$

4.7.2 Gauss-Legendre & Clenshaw-Curtis quadrature

We want to compute all integrals in the matrix (4.24) in a unified setting such that, once initialized, we can reuse our integration scheme as often as possible. By the Theorem's 3.2.4 and 3.3.1 we achieve an error of

$$|E(\gamma, \bar{\omega}, N)| \leq e^{-D}$$

for

$$N \geq \frac{\log(64/15M) + D - \log(1 - e^{-2r})}{c\, r}, \tag{4.27}$$

where $c = 2$ for Gauss-Legendre quadrature and $c = 1$ for Clenshaw-Curtis quadrature, the value

$$r = r(\gamma, \mathcal{L}) > 0$$

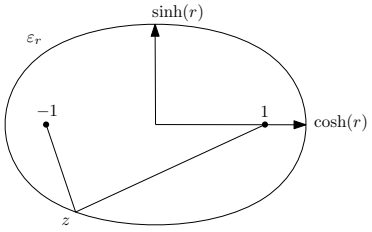
parametrizes the boundary ε_r of a region of holomorphicity and

$$M = M(\gamma, \omega, r) := \max_{z \in \varepsilon_r} \{ a_i(\tilde{\gamma}_j(z))\gamma'(z) \mid i = 1, \ldots, g, j = 1, \ldots, m \} > 0$$

bounds the absolute value of the integrands along the boundary of that region. From (4.27) we can see that the value of r has strong influence on the size of N while the contribution of M is merely logarithmic. So, in order to minimize N the priority is to maximize r such that M is still decent. The parameters r and M for (GL) and (CC) are identical, so we can treat both methods simultaneously.

Computation of r Recall that the value of $r = r_\varepsilon > 0$ parametrizes the ellipse

$$\varepsilon_r = \{\, z \in \mathbb{C} \mid |z - 1| + |z + 1| = 2\cosh(r) \,\}$$



with foci ± 1 and sum of lengths of the semi-axes

$$\exp(r) = \cosh(r) + \sinh(r)\,.$$

We want to choose $r = r(\gamma, \mathcal{L})$ as large as possible such that $a(\tilde{\gamma}(u))\gamma'(u)$ is still holomorphic inside ε_r. Since γ' is entire and $a(x, y(x))$ is holomorphic for all $x \in \mathbb{C} \setminus \mathcal{L}$, $a(\tilde{\gamma}(u))$ is holomorphic for all $u \in \gamma^{-1}(\mathcal{L})$. Hence, we have to choose

$$r < r_0 := \min\{\, r_u \mid u \in \gamma^{-1}(\mathcal{L}) \,\} \tag{4.28}$$

where

$$\cosh(r_u) = (|u - 1| + |u + 1|)/2\,.$$

In practice we will choose r heuristically slightly smaller than r_0. In particular, taking

$$\cosh(r) = \begin{cases} (\cosh(r_0) + 1)/2, & \text{if } r_0 \leq 1/50, \\ \cosh(r_0) - 1/100, & \text{otherwise.} \end{cases}$$

has been working out fine.

Computation of M For general affine algebraic curves computing this value is really problematic. For the integrals appearing in (4.24) it is possible to compute upper bounds for the numerator of $a(\tilde{\gamma}(u))\gamma'(u)$: we find bounds for the values $|\gamma(z)|$ and $|\gamma'(z)|$ for $z \in \varepsilon_r$ from the parametrization of γ, see §4.3.3. This can be used to find an upper bound on the $|y_j(x)|$, as described in §4.10.1, from the defining equation $f(x, y) = 0$. Using the triangle inequality then yields a bound on the numerator.

The hard part is to bound the denominator of $a(x, y) \in \mathbb{C}(X)$ from below. Generally, the denominator is a bivariate polynomial (mostly $\partial_y f(x, y)$) and there is no hope of computing a lower bound analytically. In the superelliptic case (see Section 5.5) this is possible and M can be computed quite easily.

A pragmatic workaround is to obtain an approximation of M by sampling the integrands on points of ε_r that are close to singularities. Since

$$|a(\tilde{\gamma}(z))| \to \infty \quad \text{as} \quad z \to u \in \gamma^{-1}(\mathcal{L})$$

it is very likely that this approximation is sufficiently close to the true value of M. Since the size of M contributes logarithmically to N, this does not lead to precision issues in practice. An algorithm that computes the distance to an ellipse can be found in the appendix (§A.1).

Rigor The downside of this workaround is that our numerical integration scheme is no longer rigorous. This can be avoided if one uses ball arithmetic (for example the C-library ARB [47]). One simply evaluates the integrand at finitely many arithmetic balls whose centers lie on the ellipse where the the radii are chosen such that the union of these balls contains the boundary ε_r as a subset. Naturally, in ball arithmetic, one can always obtain an upper bound for the value of a function evaluated at an arithmetic ball. Until these upper bounds are finite for all balls in the covering, we recursively introduce more points by subdividing the radii of the balls that yield infinite values. By our choice of $r < r_0$ this process is finite and leads to rigorous error bounds.

4.7.3 Double-exponential integration

Now we want to employ the double-exponential integration scheme, using the estimates presented in Section 3.4, to approximate the matrix of integrals (4.24). More precisely, we apply Theorem 3.4.1 with $\alpha = 1$ and $\lambda = \frac{\pi}{2}$ to the functions

$$a_i(\tilde{\gamma}_j(u))\gamma'(u)\,. \tag{4.29}$$

Recall from Section 3.4 that in the case of (DE) integration the area of holomorphicity is the burger-shaped area (see also Figure 3.2)

$$Z_r = \{\, \tanh(\lambda \sinh(z)) \mid |\mathrm{Im}(z)| < r \,\}$$

which is parametrized by a value

$$r = r(\gamma, \mathcal{L}) \in]0, \pi/2[\,.$$

chosen such that the integrands (4.29) are holomorphic inside Z_r. In particular, this means that we achieve an error of

$$|E(\gamma, \bar{\omega}, N)| \leq e^{-D}$$

if we choose

$$h \leq \frac{2\pi r}{D + \log(2M_2 B(r, 1) + e^{-D})} \quad \text{and} \quad Nh \geq \operatorname{asinh}\left(\frac{D + \log(8M_1)}{2\lambda}\right),$$

where the constants M_1 and M_2 are given by

$$\begin{aligned} M_1 &:= M_1(\gamma,\bar{\omega}) := \max_{z\in[-1,1]} \{\, a_i(\tilde{\gamma}_j(z))\gamma'(z) \mid i=1,\dots,g, j=1,\dots,m \,\}, \\ M_2 &:= M_2(\gamma,\bar{\omega},r) := \max_{z\in Z_r} \{\, a_i(\tilde{\gamma}_j(z))\gamma'(z) \mid i=1,\dots,g, j=1,\dots,m \,\} \end{aligned} \tag{4.30}$$

and the quantiy $B(r,1)$ is given by (3.66).

Computation of r Similar to (GL)/(CC) integration, we choose the parameter $r = r(\gamma,\mathcal{L})$ as large as possible such that

$$r < r_0 := r_{u_0} = \min\{\, r_u \mid u \in \gamma^{-1}(\mathcal{L}) \,\}$$

where r_u is given by

$$u = \tanh(\lambda \sinh(t_u + i r_u)) \,.$$

Hence we ensured that $a(\tilde{\gamma}_j(u))$ is holomorphic on the set Z_r and that the constants M_1 and M_2 are finite. For actual numerical integration we obtain r from scaling down r_0 heuristically; the value

$$r = (19/20) r_0$$

has been producing satisfying results in practice.

Computation of M_1 and M_2 Here, we run exactly in the same problems as for the computation of M in §4.7.2: while the numerator can be bound from above in exactly the same way than for M, we are unable to find lower bounds for the denominator. As before, we have to sacrifice rigor of our numerical integration here and instead sporadically evaluate the integrands $a_i(\tilde{\gamma}_j(u))\gamma'(u)$ on the boundary of Z_r. In particular, in the neighborhood of close-by singularities as well as on the end points ± 1. Finally, choosing M_2 as the maximum of these sampled values and $M_1 := M_2$ is working out in practice. The distance to the boundary ∂Z_r can be computed using Newton's method, see §5.5.3. As for the computation of M, rigor (§4.7.2) can be retrieved by using ball arithmetic to compute the bounds M_1 and M_2.

4.7.4 Integration algorithm

Finally, we formulate our algorithm for numerical integration of differential forms. Let X be a compact Riemann surface given by an affine equation $f(x,y) = 0$ as discussed in Section 4.1 and let $\bar{\omega}$ a basis of $\Omega^1(X)$. Moreover, let $\mathcal{L} = \mathcal{L}(\varphi_x)$ denote the exceptional values of the corresponding holomorphic ramified covering $\varphi_x : X \to \mathbb{P}^1$ (see §4.1.1).

Algorithm 4.7.1 (Integration)**.** For a path $\gamma : [-1,1] \to \mathbb{C}\setminus\mathcal{L}$ and a basis of differentials $\bar{\omega}$, this algorithm (heuristically) computes the matrix of integrals (4.24) to precision D, i.e. such that $E(\gamma,\bar{\omega},N) \le e^{-D}$, using a suitable integration scheme.

(1) Compute the number of abscissas N for a given integration method (GL), (CC), (DE) with respect to $r(\gamma,\mathcal{L})$, $M(\gamma,\bar{\omega},r)$ (resp. M_1, M_2) and D.

(2) Compute the integration scheme, i.e. abscissas and weights $\{u_\ell, w_\ell\}$, $l = 1,\dots,N$.

(3) Compute the discrete lifts $\tilde{\gamma}(N)$ (4.18) of γ using Algorithm 4.5.1 (including the values of γ').

(4) For each abscissa u_ℓ, evaluate the vector of differentials $\bar{\omega}$ on all sheets and multiply the resulting matrix with w_ℓ and $\gamma'(u_\ell)$.

(5) Summing up these matrices results in the approximation (4.24).

Correctness From the results of this chapter it is clear that Algorithm 4.7.1 is correct up to computational issues. Firstly, the computation of the bound M (resp. the bounds M_1, M_2) is quite problematic, as explained previously in this Section (see §4.7.2 resp. §4.7.3). Once we computed suitable integration parameters, the Theorems of Chapter 3 provide rigorous error bounds. Secondly, one has to prevent precision loss; this is discussed in §4.10.

Complexity Incorporating our prior complexity analyses we are set to analyze Algorithm 4.7.1, which is central to this thesis.

Recall that D is the desired precision (a number of digits), $m = \deg(f, y)$ is the degree of the covering and $d = \deg(f)$ is the total degree.

Step (1) does not contribute, as the parameters can be computed in fixed low precision (say ≈ 20 digits). The cost of initializing an integration scheme for precision D, denoted $\mathrm{Init}(N, D)$ with $N = N_{\min}(D)$, has been analyzed in Chapter 3 and depends on the chosen integration method. Computing the discrete lifts costs $O(N(\log D + m^2)\mathcal{M}(D))$ via analytic continuation, see (4.20). Evaluating the differentials at u_ℓ takes $O(gmd\mathcal{M}(D)) = O(d^4\mathcal{M}(D))$ operations, as discussed in §4.2, while multiplication with w_ℓ and $\gamma'(u_\ell)$ costs $gm\mathcal{M}(D)$. Hence, step (4) requires a total of $O(N(d^4 + gm)\mathcal{M}(D)) = O(Nd^4\mathcal{M}(D))$ operations. Finally, step (5) requires $O(NgmD) = O(Nd^3\mathcal{M}(D))$ operations. Thus, we obtain the following

Theorem 4.7.2. *Algorithm 4.7.1 has computational complexity in*

$$\begin{cases} O(D(D + d^4)\mathcal{M}(D)), & \text{using Gauss-Legendre integration,} \\ O(D(\log D + d^4)\mathcal{M}(D)), & \text{using Clenshaw-Curtis integration,} \\ O(D \log D(\log D + d^4)\mathcal{M}(D)), & \text{using double-exponential integration.} \end{cases}$$

Proof. The complexities of our prior discussion add up to

$$O(N_{\min}(D)(\log D + d^4)\mathcal{M}(D) + \mathrm{Init}(D)) .$$

Combining this with the results from Chapter 3, in particular with the following complexities from Table 3.6

- $\mathrm{Init}(D) = O(D^2\mathcal{M}(D))$ and $N_{\min}(D) = O(D)$ for Gauss-Legendre,
- $\mathrm{Init}(D) = O(D \log D\mathcal{M}(D))$ and $N_{\min}(D) = O(D)$ for Clenshaw-Curtis and
- $\mathrm{Init}(D) = O(D \log^2 D\mathcal{M}(D))$ and $N_{\min}(D) = O(D \log D)$ for double-exponential

yields the claim. □

4.7.5 Improving integration paths

As already mentioned, it might happen that numerical integration along some path γ becomes costly, due to nearby singularities causing a small value of $r = r(\gamma, \mathcal{L})$. In such situations splitting up the integral matrix

$$\int_\gamma \bar{\omega} = \int_{\gamma_1} \bar{\omega} + \int_{\gamma_2} \bar{\omega}$$

where $\gamma = \gamma_1 \circ \gamma_2$, can lead to massive improvements as shown in Example 4.7.4 below.

Splitting line segments Splitting up the integration path is particularly efficient in the case of line segments when using an ellipse as boundary for the area of holomorphicity, like we do for (GL) or (CC) integration. For the burger-shaped area (see Figure 3.2), which is used for (DE) integration, splitting line segments rarely improves things. An example for this is discussed in §5.6.4.

We lay down our strategy for splitting line segments in the case of (GL) and (CC) integration using (4.27) to compute the number of integration points.

Algorithm 4.7.3 (Splitting line segments)**.** Let $\gamma : [-1,1] \to [a,b]$ be a line segment. Splitting(γ) recursively subdivides into sub-paths as long as the splitting reduces the total number of integration points.

(1) Let $u_0 \in \gamma^{-1}(\mathcal{L})$ correspond to r_0 (4.28).

(2) Set $t \leftarrow \operatorname{sgn}(\operatorname{Re}(u_0)) \cdot \min(|\operatorname{Re}(u_0)|, 3/4)$.

(3) Set $x_0 \leftarrow \gamma(t)$.

(4) Split γ into $\gamma_1 \leftarrow [a, x_0]$ and $\gamma_2 \leftarrow [x_0, b]$.

(5) Compute N, N_1, N_2 using (4.27).

(6) If $N_1 + N_2 < N$, then Paths $\leftarrow$ Splitting(γ_1) $\cup$ Splitting(γ_2), otherwise Paths $\leftarrow [\gamma]$.

(7) Return Paths.

Example 4.7.4. In order to see how the splitting works in practice, we continue with the genus 17 Riemann surface $X : f_1 = 0$ from Example 4.2.2 where

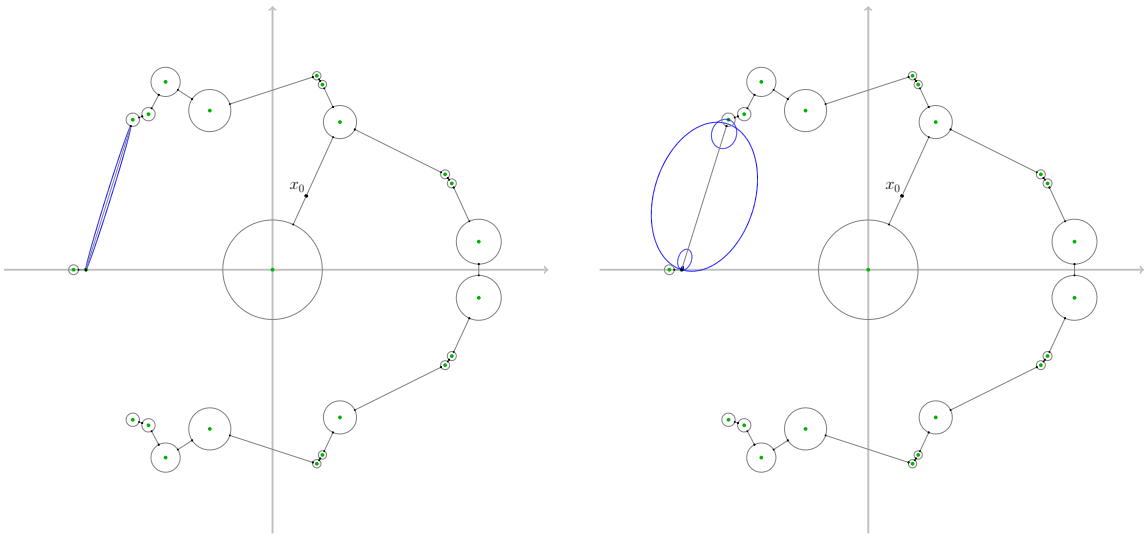
$$f_1 = x^9 + 2x^6y^2 + x^2 + y^6 + 2y^4 = 0\,.$$

Figure 4.5 shows the fundamental group obtained from applying Algorithm 4.3.1. Highlighted in (a) is the image of ε_r under the line segment

$$\gamma = [-0.71247 + 0.72197 \cdot I, -0.93815 + 0.00039 \cdot I].$$

There are two discriminant points very close to γ with respective distances of 0.033 to the start point and 0.00017 to the end point that are responsible for the small value of $r = 0.03$. For $D_{10} = 100$ digits of precision, we would need $N = 3942$ abscissas using (GL) integration.

Algorithm 4.7.3 applied to γ yields 5 paths with respective values 1.81, 1.32, 1.29, 1.32, 1.24 of r_ε resulting in a total of $100 + 152 + 161 + 154 + 176 = 743$ integration points instead of 3942. Figure 4.5 (b) shows the corresponding images of the new ellipses. For the same integral we achieve a value of $r_Z = 0.34$ using (DE) integration and we would need $N = 1125$ integration points to achieve precision $D_{10} = 100$. Immediately it becomes clear that without splitting (DE) would be the better choice here while with splitting it is not.



(a) Ellipse $\gamma(\varepsilon_r)$ before splitting.

(b) Multiple ellipses after splitting γ.

Figure 4.5: Paths for the fundamental group $\pi_1(\mathbb{C} \setminus \mathcal{L}(f_1, y), x_0)$.

Improving circles and arcs We could also apply Algorithm 4.7.3 to circles and arcs in order to try to improve the value of $r(\varepsilon)$. Unfortunately, in almost all cases this improvement is not significant enough to justify splitting the path. More influential on their values of r, as already mentioned in §4.3.1, is their radius $c > 0$. Let

$$\gamma_{\text{fc}} : u \mapsto c \exp(i(\pi o(u+1) + \varphi_0)) + x$$

be the parametrization of a full circle (§4.3.3) where $x \in \mathcal{L}$ is an exceptional value. Then, in the cases of (GL) and (CC) integration, for all $\tilde{u} \in \mathcal{L}$ we have

$$\gamma_{\text{fc}}^{-1}(\tilde{u}) = \frac{-oi}{\pi} \log\left(\frac{\tilde{u} - x}{c \exp(i\varphi_0)}\right),$$

and from (4.28) we see that

$$r_{\tilde{u}} \to \infty \quad \text{and} \quad M(r) \to \infty \quad \text{as} \quad c \to 0\,.$$

Letting the radius tend towards zero gives us larger values for r_ε, but the bound $M = M(r)$ also tends to infinity (analogously for arcs). In theory, choosing the radii quite small should make integration along such paths easier. The same is true for (DE) integration, but there we have that

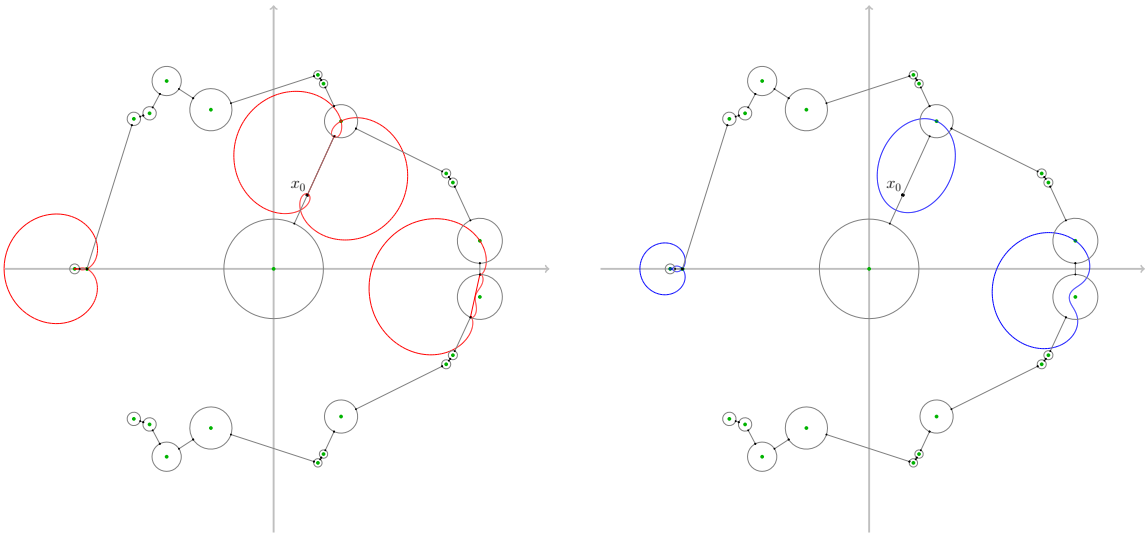
$$r_{\tilde{u}} \to \pi/2 \quad \text{and} \quad M_2(r) \to \infty \quad \text{as} \quad c \to 0\,.$$

Keep in mind that one has to be careful to choose the correct branch of the complex logarithm when computing $\gamma_{\text{fc}}^{-1}(\tilde{u})$ and $\gamma_{\text{arc}}^{-1}(\tilde{u})$.

In practice, however, choosing small radii has several drawbacks: performing (certified) analytic continuation very close to a exceptional value leads to a much bigger number of refinement steps (see Section 4.5) and may also cause numerical instability. Moreover, having circles and arcs with smaller radii also decreases the respective values of r of the line segments connecting them, due to their end points being closer to exceptional values. Intensive testing has shown that (even with splitting bad line segments) keeping a certain distance (see §4.3.1) from exceptional values is the preferable choice.

4.7.6 Comparison of integration methods (I)

Conducting comparisons between the integration methods presented in Chapter 3 applied to the integration of differential forms on Riemann surfaces is one of the main topics of this thesis. Here, we begin by comparing the performance of Gauss-Legendre (GL), Clenshaw-Curtis (CC) and double-exponential (DE) integration along individual paths $\gamma : [-1, 1] \to \mathbb{C} \setminus \mathcal{L}$, i.e. application of Algorithm 4.7.1. Here we are particularly interested in how the computational cost is actually distributed.



(a) Zone $\gamma(Z_r)$ for paths $\gamma_{\text{fc}}, \gamma_{\text{ls}}, \gamma_{\text{arc}}$.

(b) Ellipse $\gamma(\varepsilon_r)$ for paths $\gamma_{\text{fc}}, \gamma_{\text{ls}}, \gamma_{\text{arc}}$.

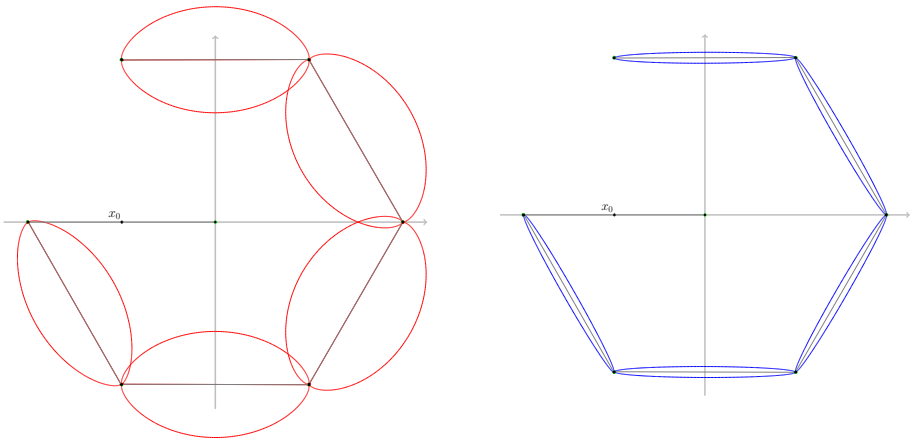
Figure 4.6: Paths for the fundamental group $\pi_1(\mathbb{C} \setminus \mathcal{L}(f_1, y), x_0)$.

Example 4.7.5 (cont.)**.** We continue with Example 4.7.4 by considering three paths that were constructed for the fundamental group: the full circle γ_{fc}, the line segment γ_{ls} and the arc γ_{arc}, as can be seen in Figure 4.6 (from left to right). These paths are chosen in a way such that they represent the 'average' integral from a numerical integration perspective. Figure 4.6 (a) shows (in red) the boundaries of the zones $\gamma(Z_r)$ that are used for (DE) integration, while (b) highlights the boundaries of the ellipses $\gamma(\varepsilon_r)$, used for (GL)/(CC) integration, in blue.

Timings for Algorithm 4.7.1 applied to these paths are given in Table 4.4. Analytic continuation is done via Algorithm 4.5.1 using Durand-Kerner's method. A description for a basis of differentials $\bar{\omega}$ as products of irreducible factors was already given in Example 4.2.2. This product representation is used for evaluation of the differentials. Although we only analyze three different integrals here, they give us a pretty good idea of the performance of our integration methods. Table 4.4 indicates many interesting facts that can be readily generalized. Keep in mind that we consider the initialization cost for integration along a single path here.

Path	γ_{fc}			γ_{ls}			γ_{arc}		
Integration method	GL	CC	DE	GL	CC	DE	GL	CC	DE
Parameter r_ε/r_Z	0.55		0.46	0.97		0.89	0.86		0.59
D_{10} = Digits	100								
N = #Abscissas	229	461	889	127	253	467	144	289	705
Initialization	0.19	0.04	0.01	0.06	0.02	0.00	0.08	0.02	0.01
Eval. of γ, γ'	0.00	0.01	0.02	0.00	0.00	0.00	0.00	0.01	0.01
Anal. cont. of γ	0.20	0.40	0.64	0.12	0.25	0.35	0.13	0.25	0.51
Eval. of $\bar{\omega}$	0.23	0.45	0.89	0.12	0.47	0.46	0.15	0.29	0.70
Total time	0.62	0.90	1.56	0.30	0.49	0.81	0.36	0.57	1.22
D_{10} = Digits	500								
N = #Abscissas	1094	2188	5557	604	1209	2897	685	1373	4399
Initialization	8.82	4.73	0.16	2.71	1.28	0.08	3.56	0.24	0.12
Eval. of γ, γ'	0.09	0.18	0.94	0.00	0.01	0.02	0.05	0.11	0.35
Anal. cont. of γ	2.39	4.71	9.45	1.30	2.57	4.95	1.49	2.94	7.50
Eval. of $\bar{\omega}$	2.48	4.98	12.3	1.35	2.72	6.46	1.56	3.11	9.66
Total time	13.8	14.6	22.8	5.36	6.58	11.51	6.66	6.40	17.6

Table 4.4: Timings for numerical integration (in seconds).



(a) Zone $\gamma(Z_r)$ for different γ. (b) Ellipse $\gamma(\varepsilon_r)$ for different γ.

Figure 4.7: Paths for the fundamental group $\pi_1(\mathbb{C} \setminus \mathcal{L}(f_6, y), x_0)$.

Example 4.7.6. We consider yet another example, namely, the genus 14 Riemann surface defined by

$$f_6 = 23x^6y + xy^5 - 100y^2 - 10000y + 1\,.$$

There are $n = 32$ exceptional values and some of them are as close together as $1.0546 \cdot 10^{-5}$ which leads to 5 line segments being so close to exceptional values that even splitting them

does not improve the situation for (GL)/(CC) integration. As you can see in Figure 4.7, the burger-shaped zones (a) $\gamma(Z_r)$ are quite spacious while the ellipses (b) $\gamma(\varepsilon_r)$ really suffer from nearby singularities. We compare the timings for Algorithm 4.7.1 applied to the line segments

$$\gamma_1 : [-1.367 - 2.379 \cdot I, 1.3669 - 2.3861 \cdot I]$$
$$\gamma_2 : [2.7475 + 0.008 \cdot I, 1.3788 + 2.3742 \cdot I]$$

Considerings the timings given by Table 4.5, This example highlights perfectly that (DE) integration can be extremely useful.

Path	γ_1			γ_2		
Integration method	GL	CC	DE	GL	CC	DE
Parameter r_ε/r_Z	0.0606		0.28	0.0613		0.38
D_{10} = Digits	100					
N = #Abscissas	2867	5733	1445	2832	5665	467
Initialization	3.05	4.24	0.01	3.00	1.05	0.00
Eval. of γ, γ'	0.01	0.02	0.01	0.02	0.02	0.00
Anal. cont. of γ	2.11	4.24	0.93	2.06	3.94	0.35
Eval. of $\bar{\omega}$	1.69	3.49	0.86	1.68	3.47	0.46
Total time	6.86	11.8	1.81	6.75	8.48	0.81
D_{10} = Digits	500					
N = #Abscissas	13610	27221	9045	13445	26893	6737
Initialization	119	82.6	2.71	118	18.8	0.19
Eval. of γ, γ'	0.06	0.12	0.04	0.06	0.12	0.03
Anal. cont. of γ	24.2	46.9	13.6	24.0	46.3	10.2
Eval. of $\bar{\omega}$	19.0	39.2	12.2	18.7	38.9	9.20
Total time	162	169	26.2	161	104	19.6

Table 4.5: Timings for numerical integration (in seconds).

Conclusions

Initialization Computing the integration scheme is a huge factor for the numerical integration of our differentials and depends on the prescribed precision D and the parameter $r = r(\gamma, \mathcal{L})$ (we ignore the constants M, M_1, M_2 here). For this we applied the algorithms described in Chapter 3. The obvious big winner here is (DE) as its initialization cost is really negligible in practice (see also §3.4.1). Definitely slower than (DE), but still reasonable fast we compute the (CC) scheme using the (FFT), as described in §3.3, in most instances while the classical algorithm (CL) was used for γ_{fc} and $D_{10} = 500$. As we have already seen in §3.2.2, computing the (GL) scheme is simply slow (although our implementation of of (GLR) provides a nice speed-up for γ_1 and γ_2).

Number of abscissas The number of integration points N scales the amount of operations that are required (beyond initialization) linearly and the initialization itself at least linearly (depending on the algorithm that is being used). Since our cost per abscissa is quite high, we really want to this quantity to be minimal. This is where (GL) integration shines: for the 'average' integrals it outclasses the other methods due to the minimal value of N, making (GL) the big winner in this category (and therefore overall).

Unsurprisingly, and independently of our examples, (CC) requires about twice as many integration points as (GL), which is a major drawback of (CC) in our examples.

Another important lesson here is that there exist integrals (e.g. along γ_1, γ_2 from Example 4.7.6) where (DE) outclasses (GL) and (CC) not only in initialization, but also requires much less integration points. For compact Riemann surfaces such integrals might occur during period matrix computations (or for the Abel-Jacobi map) when the exceptional values cluster. The timings for γ_1 and γ_2 strongly suggest to use (DE) whenever we encounter such integrals. In Example 4.7.6 this means using (DE) integration for 5 integrals and (GL)/(CC) for the other 94 integrals occurring in the period matrix computation (see Table 4.5).

Most integrals are much easier to integrate though, comparable to $\gamma_{\text{fc}}, \gamma_{\text{ls}}, \gamma_{\text{arc}}$ or even easier. So generically, (GL) and (CC) will require much fewer integration points than (DE) and will thus be preferred over (DE) in almost all cases.

Evaluating the parametrization The costs for evaluating γ and γ' are independent of m and g and only depend on the precision D. It is negligible compared to the other steps, even for circles and arcs (whose evaluation involves computing an exponential). This is, of course, partly due to our examples having high genera (17 and 14) and degrees (6 and 5), but even when g and m are small and D is large, we can ignore this cost.

Analytic continuation For any path γ that we integrate along, analytic continuation is one of the most time-consuming tasks (due to the large number of abscissas) and depends heavily on m and D. Again, this is completely independent of the examples considered above. We put quite some effort and time into optimization here (using different root approximation methods, see §4.4, and an efficient implementation of Algorithm 4.5.1), but in the end this remains costly. Note that analytic continuation was not the dominant factor in the complexity analysis of Algorithm 4.7.1.

Evaluation of differentials Evaluating the vector of differentials requires roughly as much time as analytic continuation in our examples. This cost depends heavily on the genus g, the total degree d of the affine equation, how many irreducible factors we need to evaluate and on the required precision D. In our example the genera are quite high, so the cost here really explodes and evaluation of differentials takes half of the time after

initialization of the integration scheme. For Riemann surfaces of lower genus we can (generally) expect this step to cost less time than analytic continuation.

Recycling integration schemes So far we ignored the issue of reusing integration schemes in our comparison, because we only considered integration along single paths. During period matrix computations we usually integrate a large number of paths, $(3n-2)$ with $n = \#\mathcal{L}(\varphi_x)$ to be exact. Hence it is advantageous to classify paths according to the value of r (see §4.8.2), compute an integration scheme for the 'worst' integral within the class and then use it to evaluate all integrals belonging to the class.

Since we can easily decide whether to use (DE) or not, this mainly influences our choice between (GL) and (CC). The trade-off is between fewer integration points (GL) and faster initialization (CC). The more integrals can be computed with the same scheme, the more efficient (GL) becomes and the less impact does its costly initialization have. Conversely, we should prefer (CC) if we only need to evaluate a few integrals. There are many other factors that either favor one or the other method, which makes it hard to find an optimal answer to that question. For more details, see part (II) of our comparison in §4.8.3.

Which method is best in practice? We briefly sum up the main points of our comparison so far.

Features of Gauss-Legendre (GL) integration:

- minimal number of abscissas N for well-behaved integrands, i.e. for average to high values of r (++)
- lengthy initialization, as abscissas and weights have to be iterated which becomes slow for high precision D (−)
- large value of N for integrands with nearby singularities, i.e. r close to 0 (−)

Features of Clenshaw-Curtis (CC) integration:

- N twice as big as for (GL), but still reasonable (+)
- fast initialization using the fast Fourier transform (+)
- similar problems as (GL) for very small r (−)

Features of Double-exponential (DE) integration:

- extremely fast initialization for any precision D (+)
- robustness: (DE) shines for badly-behaved integrands with nearby singularities (++)
- significantly higher number of abscissas (−−)

Conclusion: Use Gauss-Legendre as standard integration method and double-exponential sporadically (for dreadful integrals).

4.8 Strategy for the period matrix

Finally, we lay out our period matrix algorithm. Structurally, it does not differ too much from the algorithms employed by [27] and [35]. Nonetheless, it is worth writing down the steps in order to give an analysis of its complexity.

Algorithm 4.8.1 (Period matrix)**.** Given a standard basis (4.8) of holomorphic differentials $\bar{\omega}$ of $\Omega^1(X)$ and a holomorphic ramified covering $\varphi_x : X \to \mathbb{P}^1$, this algorithm (heuristically) computes a basis of the period lattice Λ for the compact Riemann surface X to precision $D > 0$.

(1) Compute the locus $\mathcal{L}(\varphi_x)$ (§4.1.2) of exceptional values; with $n \leftarrow \#\mathcal{L}$.

(2) Compute generators $\{\gamma_1, \ldots, \gamma_n\}$ of $\pi_1(\mathbb{C} \setminus \mathcal{L}, x_0)$ using Algorithm 4.3.1.

(3) Apply Algorithm 4.7.1 to compute the matrix of integrals (4.24) to precision D for each of the $3n-2$ paths that make up the fundamental group.

(4) Obtain a monodromy representation $\mathrm{Mon}(\varphi_x)$ (as described in §4.5.1) and the integral matrices along the γ_i by concatenations (using the $3n-2$ path pieces).

(5) Apply the Tretkoff algorithm §4.6.1 to $\mathrm{Mon}(\varphi_x)$ and obtain a generating set Γ of $H_1(X, \mathbb{Z})$ with intersection matrix K_Γ.

(6) Piece together a period matrix $\Omega_\Gamma = (c_j)_{j=1,\ldots,2g+m-1}$ with

$$c_j = \sum_{\tilde{\gamma} \in \Gamma_j} \int_{\tilde{\gamma}} \bar{\omega} \in \mathbb{C}^{g \times 1}$$

using the rules for concatenation §4.7.1.

Correctness By the theory of compact Riemann surfaces, that has been established in Section 2.4, the above algorithm is correct and does compute a period matrix. The problematic part is to guarantee that the result is indeed correct to precision D. This is precisely the same issue that has been described in §4.7.4.

Complexity We are mainly interested in how Algorithm 4.8.1 behaves in terms of the precision D. Therefore, the steps (3), (4) and (6) are the most important ones, but we analyze the other steps as well.

(1) Computing the exceptional values was covered in §4.1.2. On the exact side we have to calculate and factorize the discriminant and the leading coefficient over K, but this is independent of the precision. The numerical part here consists of computing the roots of the resulting factors over $\mathbb{C}$, which takes $\mathcal{R}(k, D)$ operations (see §1.3) for each irreducible factor of degree $k \in \mathbb{Z}_{>0}$. However, this has to be done only once, independently of the core of the algorithm.

(2) We compute generators for the fundamental group using $O(n^2\mathcal{M}(D)) = O(d^4\mathcal{M}(D))$ operations (4.10).

(3) Algorithm 4.7.1 whose complexity, depending on the integration method, is given by Theorem 4.7.2, is applied $3n-2 = O(n) = O(d^2)$ times.

(4) The local monodromies are obtained through analytic continuation, which is part of step (3). For each closed path γ_i ($i = 1, \ldots, n$) we add together $O(n)$ integral matrices of dimension $g \times m$, which takes $O(n^2 mgD) = O(n^2 d^3 D) = O(d^7 D)$ operations.

(5) As stated by Theorem 4.6.1, the Tretkoff algorithm is independent of D and has complexity $O(d^5 \log d)$.

(6) There are $2g+m-1$ cycles of length $O(m)$. For each change of sheet we require up to m values for each differential, so obtaining the integrals c_j along the cycles Γ_j (adding together the correct values of the integral matrices from step (4)), takes up to $O(2g+m-1)m^2gD) = O(d^6D)$ operations.

Adding up all the costs of the individual steps we see that the complexity in terms of D is heavily dominated by Algorithm 4.7.1, which includes computing the integration schemes, analytic continuation and numerical integration. The costs of steps (1) and (2) are absorbed by step (3) in the big-O notation. The costs of steps (5) and (6) are swallowed by the $O(d^7D)$ of step (4). All in all, we obtain the following result:

Theorem 4.8.2. *Algorithm 4.8.1 has computational complexity in*

$$\begin{cases} O(d^2D^2(D+d^4)\log^{1+\varepsilon} D + d^7D), & \text{using Gauss-Legendre integration,} \\ O(d^2D^2(\log D+d^4)\log^{1+\varepsilon} D + d^7D), & \text{using Clenshaw-Curtis integration,} \\ O(d^2D^2\log D(\log D+d^4)\log^{1+\varepsilon} D + d^7D), & \text{using double-exponential integration.} \end{cases}$$

Proof. Combine the complexities of our analysis with the complexities of Theorem 4.7.2 and plug in $\mathcal{M}(D) = O(D\log^{1+\varepsilon} D)$. □

Corollary 4.8.3. *Let X be a compact Riemann surface of genus $g>0$ defined by an affine equation $f(x,y)=0$, $f\in\mathbb{C}[x,y]$ irreducible of total degree d, and $(\omega_1,\dots,\omega_g)$ a standard basis* (4.8) *$(\omega_1,\dots,\omega_g)$ a basis of the space of holomorphic differentials $\Omega^1(X)$. There is an algorithm that (heuristically) computes a basis of the period lattice Λ to precision $D>0$ using*

$$O(c(f)d^2D^2(\log D+d^4)\log^{1+\varepsilon} D + d^7D) \text{ operations,}$$

where $c(f) = 1/r$, for some $r>0$ (see §4.7.2), depends on the configuration of the exceptional values (4.2)*. In particular, r depends on their absolute values and absolute and relation distances between them.*

Proof. We obtain the claimed complexity (except for the factor $c(f)$) as an consequence of Theorem 4.8.2 using Clenshaw-Curtis integration, where the holomorphic ramified covering $\varphi_x : X\to\mathbb{P}^1$ is defined by $f(x,y)=0$. The linear factor $c(f)$ comes from the minimal value of $r = r(\gamma,\mathcal{L})$ (see §4.7.2) appearing in the $3n-2$ numerical integrations along a path γ that are performed in step (3) of Algorithm 4.8.1. Once we stop viewing r as a constant, the claimed dependence on the configuration of exceptional values becomes clear from its definition. □

4.8.1 Big and small period matrices

We can easily extend Algorithm 4.8.1 from computing Ω_Γ (a basis for the period lattice Λ) to computing big and small period matrices by adding a few steps.

Algorithm 4.8.4 (Big/small period matrix)**.** Given a period matrix Ω_Γ to precision D with intersection matrix K_Γ, this algorithm (heuristically) computes a big period matrix (Ω_A,Ω_B) and a small period matrix τ to precision D.

(1) Compute a symplectic base change matrix $S\in\mathrm{GL}(\mathbb{Z},2g+m-1)$ for K_Γ.

(2) The first $2g$ columns of $\Omega\cdot S$ form a big period matrix $(\Omega_A,\Omega_B)\in\mathbb{C}^{g\times 2g}$.

(3) A small period matrix is then obtained by solving $\Omega_A\cdot\tau=\Omega_B$.

Complexity The complexity of Algorithm 4.8.4 is dominated by matrix algorithms and thus not very exciting. Computing the matrix S is an exact computation and takes negligible time since the entries of K_Γ only consist of 0 and ± 1. Moreover, as you can see in Example 4.6.3, the base change S is usually a sparse matrix with tiny integer coefficients (less than m), so that the change of basis is a matrix multiplication of size $O(g)$ with precision D coefficient and is performed performed using $O(g^\omega \mathcal{M}(D))$ operations. However, we have no proof of this fact and in general the symplectic reduction could produce a dense base change matrix with coefficients of size $O(g)$, so that we state the following far from optimal result:

Theorem 4.8.5. *Algorithm 4.8.4 (heuristically) computes a big/small period matrix to precision $D > 0$ using $O(g^\eta \mathcal{M}(D+g))$ operations.*

Proof. The complexity for the big period matrix follows from the discussion above. Step (3) requires one matrix inversion and one matrix multiplication of $g \times g$ matrices of precision $D + O(g)$ numbers, which can be done using $O(g^\eta \mathcal{M}(D+g))$ operations. □

Numerical verification There are several tests that can (and should) be executed during the period matrix algorithm to confirm the numerical quality of our computations. Suppose we want our results to be correct up to precision D, then

(i) the columns $2g+1, \ldots, 2g+m-1$ of $\Omega \cdot S$ in step (2) correspond to linearly dependent cycles in Γ and have to be zero up to an error of e^{-D};

(ii) for each of the $3n-2$ individual integrations along a path γ, the absolute value of the sum over a column of the approximated integral matrix (4.24) has to be zero (because on the Riemann surface the concatenation of all lifts of γ is homotopic to zero), i.e. we can check whether

$$\left| \sum_{j=1}^{m} c_{i,j} \right| < e^{-D} \quad \text{for every } i = 1, \ldots, g, \text{ where } \quad \int_\gamma \bar{\omega} = (c_{i,j})\,.$$

(iii) Riemann's bilinear relations (§2.9.1) show that $\tau \in \mathfrak{H}_g$ is symmetric, so we can check whether

$$\max \left| \tau - \tau^T \right| < e^{-D}\,.$$

Moreover, τ has to have positive definite imaginary part, which can also be checked using the MAGMA function 'IsPositiveDefinite'.

4.8.2 Classifying integration paths

In Algorithm 4.8.1 and for its complexity analysis the assumption was that we compute an integration scheme for each of the $3n-2$ paths that we integrate along. In practice, this is completely inefficient. Instead, we group paths with similar values of r together and then compute the 'worst case' integration parameters r and M for this class. Depending on the initialization cost of the integration method that is being used and the number of total paths, we adjust the number of different classes, as can be seen in the Tables of §4.8.3. Although this approach leads to a huge gain in absolute run time, it is not a justification to view the number of integration scheme as constant in the complexity analysis §4.8.

4.8.3 Comparison of integration methods (II)

We expand our comparison of integration methods, that we started in §4.7.6, from considering individual paths to computing big period matrices.

Our favorite example f_1, see Table 4.6 below, plays out as already indicated by the timings given previously in Table 4.4: Gauss-Legendre integration (with splitting line segments) requires by far the minimal number of integration points, which is the deciding factors here. The initialization cost of the integration schemes has almost no influence here (the genus of this example is 17).

Integration method	GL	CC	DE
#Integration schemes	5	8	19
D_{10} = Digits		50	
Total #Abscissas	7384	13423	22512
Total Time	13.3	22.7	35.5
D_{10} = Digits		100	
Total #Abscissas	13905	24812	47772
Total Time	28.0	47.6	84.0
D_{10} = Digits		200	
Total #Abscissas	26917	47361	103828
Total Time	67.7	113	227

Table 4.6: Timings for the big period matrix of $X : f_1 = 0$ (in seconds).

In Table 4.5 we showed that there are integrals that can be handled easily by (DE) integration, while (GL) and (CC) struggle. As we already predicted in §4.7.6, it is optimal to handle very difficult integrals by double-exponential integration. It can be seen from Table 4.7 that this approach can drastically reduce the total number of integration points, as well as the total run time. Let us mention that efficiently combining these integration methods is not hard to implement.

Integration method	GL	CC	DE	GL/DE	CC/DE
#Integration schemes	5	8	20	5+2	8+2
D_{10} = Digits			50		
Total #Abscissas	16813	31921	31066	11482	19144
Total Time	21.1	37.2	33.8	14.5	22.5
D_{10} = Digits			100		
Total #Abscissas	31928	59135	65628	22756	36503
Total Time	45.9	76.7	80.4	31.4	47.8
D_{10} = Digits			200		
Total #Abscissas	62145	113587	142482	46043	71889
Total Time	123	192	224	81.4	123

Table 4.7: Timings for the big period matrix of $X : f_6 = 0$ (in seconds).

Finally, we consider the Riemann surface from Example 4.6.2 that is defined by the superelliptic equation

$$f_5 : y^3 = (x-2)(x-1)(x-(1+I))(x-(1-I))\,.$$

We only have $n = 4$ critical values and only 10 (easy) integrations are required to obtain a period matrix. As we can see in Table 4.8 and as we should expect from the complexity analysis, for high precision and a small number of integrals Clenshaw-Curtis beats Gauss-Legendre integration even though we need to evaluate the differentials at roughly twice the number of integration points. As already mentioned in Remark 3.6, Clenshaw-Curtis integration would do probably be even better here if we implemented a fast algorithm for the discrete cosine transform (DCT).

Integration method	GL	CC	DE
#Integration schemes	2	3	6
D_{10} = Digits		500	
Total #Abscissas	7074	13538	37562
Total Time	16.1	14.0	33.0
D_{10} = Digits		1000	
Total #Abscissas	14059	26842	82516
Total Time	87.4	79	130
D_{10} = Digits		2000	
Total #Abscissas	28009	53434	180202
Total Time	643	420	660

Table 4.8: Timings for the big period matrix of $X : f_5 = 0$ (in seconds).

Conclusion We finish our comparison of the integration methods (GL), (CC) and (DE) applied to the problem of computing big period matrices by concluding: with the MAGMA implementations of all previous described algorithms, using a combination of Gauss-Legendre integration as standard integration (with recursive splitting) and optional double-exponential integral (for very hard integrals) is the best choice for all precisions up to ≈ 500 decimal digits, (almost) independently of the Riemann surface. For even higher precisions or simple defining equations Gauss-Legendre may be replaced by Clenshaw-Curtis integration. In all following applications of our algorithm it is implied that we use the aforementioned (GL/DE) combination.

4.8.4 Comparison with other implementations

We test our implementation of Algorithm 4.8.1 in MAGMA, denoted (M), for the computation of big period matrices against the multi-precision implementations in MAPLE 17, due to Deconinck and van Hoeij [27], and in SAGE 8.0 due to Nils Bruin. We work with moderate precisions $D_{10} \in \{50, 100, 200\}$ (decimal digits) for our comparison. Both implementations cannot handle such extreme cases as posed by the polynomials f_1 or f_6 above, so we are going to look (Table 4.9) at the rather well-behaved (but still sufficiently random) Riemann surfaces defined by the family of polynomials

$$x^k y^2 + (x+y)^{k-1} + 1 \in \mathbb{Q}[x,y] \quad \text{for } k = 2, \ldots, 10\,. \tag{4.31}$$

Although there are no really challenging polynomials (from a numerical integration perspective) among these, the genus becomes large rather quickly for this family. In contemporary applications one often encounters Riemann surfaces of smaller genus.

Consequently, for the second part of our comparison (see Table 4.10), we consider several smooth plane quartics (of small discriminant) that are going to appear in the genus 3 section of the [58, LMFDB]. For the computation of the endomorphism rings of the corresponding Jacobians (see Section 6.2, in particular §6.2.1) big period matrices were computed to 100 decimal digits of precision.

We will briefly summarize the characteristics of these implementations:

- The (optimal) version of our period matrix algorithm in MAGMA uses a spanning tree method to construct integration paths (line segments, arcs and circles) (see Algorithm 4.3.1), a combination of Gauss-Legendre integration (with splitting) and double-exponential integration, using the error bounds of Chapter 3 as explained in Section 4.7, while analytic continuation is done using Algorithm 4.5.1 with Durand-Kerner's method for simultaneous root approximation.

- From personal communication with Nils Bruin we know that SAGE integrates along line segments obtained from a Voronoi cell decomposition (as already mentioned in §4.3.2); it uses adaptive Gauss-Legendre integration and Newton's method for analytic continuation.

- The MAPLE implementation is explained in detail in [27]: for the fundamental group they use a combination of line segments and semi-circles; they compute the roots above every integration point and use some minimal distance criterion for analytic continuation; numerical integration is done with the internal MAPLE routines.

k	2	3	4	5	6	7	8	9	10
Genus	1	2	6	10	14	21	28	35	45
D_{10}					50				
MAPLE	5.72	14.7	123	481	1532	3054	-	-	-
SAGE	0.95	2.70	30.7	24.3	289	110	582	1153	1818
(M)	0.34	0.64	1.97	5.37	11.6	24.2	53.5	102	211
D_{10}					100				
MAPLE	28.7	40.5	425	1545	err	4856	-	-	-
SAGE	1.65	8.78	17.5	78.7	83.2	187	742	1577	2450
(M)	0.89	1.49	4.07	11.0	24.1	48.7	98.1	188	308
D_{10}					200				
MAPLE	123	313	1315	5773	-	-	-	-	-
SAGE	6.71	49.3	411	108	661	416	1146	2867	5544
(M)	3.38	4.40	10.5	27.3	56.0	114	221	393	809

Table 4.9: Timings(s) for big period matrices associated to (4.31).

As announced in the beginning of this subsection, we extend our comparison by looking at the 9 smooth plane quartics of smallest given discriminant with respect to the ordering that is going to be used by the LMFDB . All three period matrix algorithms require as input an affine model for the Riemann surface, so we use the following equations, which were obtained by setting $z = 1$ in the respective projective equations.

$$
\begin{aligned}
q_1 &= x^3y + x^3 + x^2y^2 + 3x^2y + x^2 - 4xy^3 - 3xy^2 - 3xy - 4x + 2y^4 + 3y^2 + 2,\\
q_2 &= x^3 + x^2 + xy^3 - xy^2 + y^2 - y,\\
q_3 &= x^4 + 2x^3y + 2x^3 - 4x^2y^2 + 2x^2y - 4x^2 - xy^3 - x + 2y^4 - 3y^3 + 5y^2 - 3y + 2,\\
q_4 &= x^3 + x^2y^2 + x^2y + xy^3 + xy^2 + xy + x + y^3 + y^2,\\
q_5 &= x^3 + x^2y^2 + xy^3 - xy^2 - 2x - y^2 - 1,\\
q_6 &= x^3 + x^2y + x^2 - xy^3 + xy^2 + x - y^2 + y,\\
q_7 &= x^3 + x^2y + x^2 + xy^3 - 3xy^2 - 4x - y^4 + 2y^3 + 2,\\
q_8 &= x^3 + x^2y^2 + x^2 + xy^3 + xy + y^3 + y,\\
q_9 &= x^3 + x^2y + x^2 + xy^3 + xy + y^3 + y^2 + y\,.
\end{aligned}
$$

Of course, homogenizing the q_i and setting $x = 1$ or $y = 1$ would be fine too, as the algorithms would yield isomorphic period matrices. Again, we compare the timings for big period matrices for precisions $D_{10} \in \{50, 100, 200\}$ (decimal digits).

	q_1	q_2	q_3	q_4	q_5	q_6	q_7	q_8	q_9
D_{10}					50				
MAPLE	137	58.7	177	80.6	80.9	69	401	52.3	50.3
SAGE	27.6	13.2	11.8	4.00	2.89	9.80	41.7	3.40	3.47
(M)	1.56	0.98	2.28	0.94	1.00	0.96	1.56	0.87	0.96
D_{10}					100				
MAPLE	338	212	6513	184	200	178	-	145	165
SAGE	5663	19.8	30.0	8.45	6.04	6.24	5578	7.27	8.40
(M)	3.55	2.27	4.86	2.13	2.27	2.15	3.66	2.05	2.20
D_{10}					200				
MAPLE	2132	946	-	855	1014	875	-	578	612
SAGE	-	39.0	92.7	23.2	15.8	15.2	-	91.3	29.5
(M)	9.67	6.30	12.9	5.82	6.15	5.63	10.2	5.77	6.08

Table 4.10: Timings(s) for big period matrices associated to $q_1, \ldots, q_9$.

Conclusion

- We see that in every example shown in the Tables 4.9 & 4.10 our algorithm (M) is several times faster than the other options and finishes the computation within reasonable time, the timings scale consistently with the genus and the required accuracy.
- The SAGE implementation does reasonably well for most examples and precisions considered here. From the tables we can exhibit quite a few inconsistencies of the timings with respect to the precision. Furthermore, this implementation does not seem to be sophisticated enough to handle inputs where integration becomes 'tough' (see q_1 or q_7).
- The MAPLE implementation is just outdated: it scales badly with the genus and the accuracy, fails in particular examples, and is generally not suitable for high precision computations.

4.9 Computing the Abel-Jacobi map

Here we are concerned with explicitly computing the Abel-Jacobi map of divisors on X. For this section we assume that we have already computed a big period matrix and all related data as explained in the preceding sections of this chapter.

Let $D = \sum_{P \in X} v_P P \in \mathrm{Div}(X)$ be a divisor on X and let $P_0 \in X$ be a base point. By linearity the computation of the Abel-Jacobi map reduces to

$$\mathcal{A}(D, P_0) \equiv \sum_{P \in X} v_P \int_{P_0}^{P} \bar{\omega} \mod \Lambda.$$

It is very convenient to choose as base point the point $P_0 = (x_0, y_1(x_0)_{>_X}) \in X$ where x_0 may chosen as the 'left' base point (§4.3.1) for the fundamental group. This choice is

advantageous when we use adaptive (DE) integration for points at infinity (§4.9.5). Recall that $y_1(x_0)$ denotes the first value of the fiber over x_0 with respect to the ordering $>_X$. The choice of the sheet should be made in accordance with the Tretkoff algorithm (in §4.6.1 we chose $s = 1$).

We now split up the computation of $\mathcal{A}$ into different cases. Namely, computing to precision D the integral vector

$$\int_{P_0}^{P} \bar{\omega} \mod \Lambda \quad \text{where} \tag{4.32}$$

- $P = (x_0, y_j(x_0))$ for some $j \in I_m$ (see §4.9.1);
- $P = (x_P, y_P)$ is a non-critical point (see §4.9.2);
- $P = (x_k, y_j(x_k))$ is a non-singular, critical point (see §4.9.3);
- $P \in X$ is a singular point, point at infinity or y-infinite point (see §4.9.4);
- Alternatively, we heuristically compute $\mathcal{A}(P, P_0)$ for any $P \in X$ that is non-regular (with respect to the affine model) in §4.9.5.

After computing all the different individual vector integrals, the linear combination that corresponds to the divisor D is reduced modulo the period lattice Λ as explained in §4.9.6. We choose to represent the image of $\mathcal{A}$ as elements in the canonical torus $(\mathbb{R}/\mathbb{Z})^{2g}$.

Finally, in §4.9.7, we summarize our practical strategy for the computation of $\mathcal{A}$.

4.9.1 Moving between sheets

For the period matrix computation we needed a homology basis; its computation was discussed in 4.6. The Tretkoff algorithm uses the local monodromies to 'draw' a map of the Riemann surface in form of a graph. The cycles that are output by this algorithm tell us exactly how to navigate from sheet 1 to any other sheet $j \in I_m$ and they are expressed as concatenations of lifts of our paths for the fundamental group $\{\gamma_1, \ldots, \gamma_n\}$, see Section 4.3. So, during step (6) of Algorithm 4.8.1 we can easily (in fact, at no additional cost) obtain a matrix

$$T = (t_{i,j}) \in \mathbb{C}^{g \times m} \quad \text{such that} \quad t_{i,j} = \int_{P_0}^{P_0^{(j)}} \omega_i \,, \tag{4.33}$$

where $P_0 = (x_0, y_1(x_0))$ is the base point and $P_0^{(j)} = (x_0, y_j(x_0))$, $j \in I_m$, is another point lying above x_0. Now, the values in the j-th column of T correspond to changing from the 1st sheet to the j-th.

4.9.2 Reaching non-critical points

Assume now that $P = (x_P, y_P) \in X$ is a regular (i.e. finite and non-critical) point. The first step is to find a path in the complex plane that connects x_0 to x_P.

For that we search for the starting point (of all $3n - 2$ path pieces that make up our fundamental group) that is closest to x_P, say $\tilde{x} \in \mathbb{C} \setminus \mathcal{L}$. Afterwards, we construct a path from $\tilde{x}$ to x_P as a combination of an arc and a line segment. An arc is only necessary to avoid a potential exceptional value in the neighborhood of $\tilde{x}$ and can be constructed with radius r_i around the exceptional value x_i, as discussed in §4.3.1. In the case $|x_P - x_i| = r_i$ we only need an arc connecting $\tilde{x}$ to x_P around x_i and no line segment.

If $x_P \notin \mathcal{L}$ is in the neighborhood of an exceptional point (see Figure 4.8 (b)), i.e. $0 < |x_P - x_i| < r_i$ for some $i = 1, \ldots, n$, then we search the point on the circle of radius r_i around x_i that minimizes the distance to x_P, say $\tilde{\tilde{x}}$. The final pieces of our path

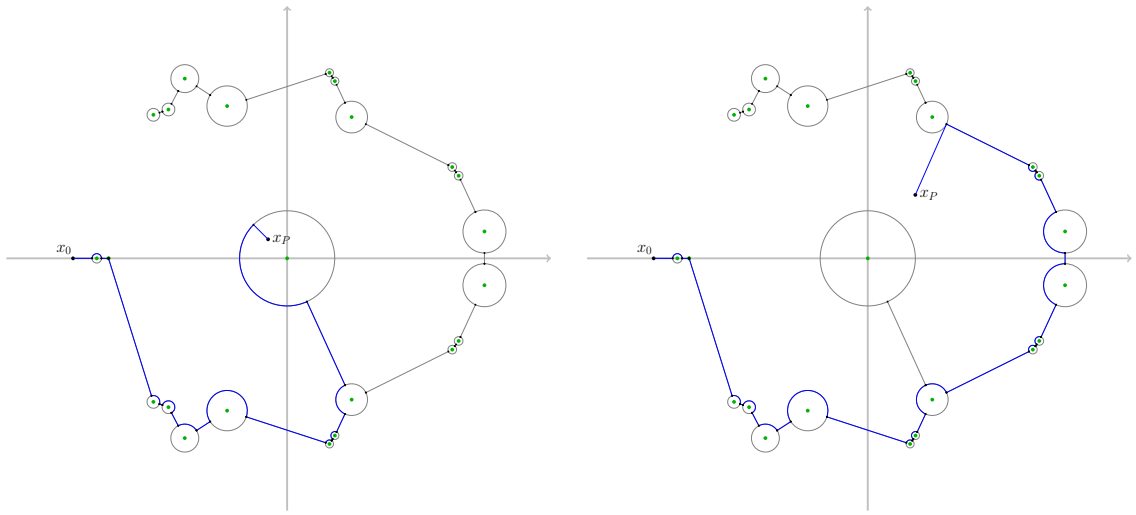
are the corresponding arc joining $\tilde{x}$ to $\tilde{\tilde{x}}$ followed by the line segment $\gamma = [\tilde{\tilde{x}}, x_P]$. This construction is pictured in blue in Figure 4.8 (a) and (b). If $x_P \in \mathcal{L}$ is an exceptional value, but P is not a critical point, we can just integrate along the lift of $\gamma = [\tilde{x}, x_P]$ to the sheet on which y_P lies in a straightforward way.

Since the integrals for all path pieces except the last one (or two) are already known from the period matrix computation, we are left with at most two integrations. In contrast to the integrations for the period matrix, where integration is done on all sheets simultaneously, we only integrate on one sheet here. For this we can use a slight modification of Algorithm 4.7.1 that uses Newton-iteration (§1.3) for analytic continuation instead of simultaneous root approximation methods.

For the sake of simplicity, we briefly assume that no arc is needed here. The idea for integrating the line segment $\gamma = [\tilde{x}, x_P]$ is that we actually integrate along the lift $-\tilde{\gamma}$ of the reverse path $-\gamma$ that is defined by the starting value y_P, i.e. $-\tilde{\gamma}(-1) = (x_P, y_P)$. So we start the Newton-iteration with y_P and it ends at some value $-\tilde{\gamma}(1) \in \varphi_x^{-1}(\tilde{x})$. We then connect $-\gamma$ to the path from x_0 to $\tilde{x}$ using the rules for concatenation §4.7.1. Since we only integrated on one sheet we cannot assign a permutation to γ. Instead, we find the correct sheet by identifying $-\tilde{\gamma}(1)$ with the corresponding value in the ordered fiber $\varphi_x^{-1}(\tilde{x})_{>X}$ and apply the formulas (4.25), (4.26) for concatenation to the resulting vector of integrals.

Remark 4.9.1. Note that integrating backwards here is important. If we start the analytic continuation at any $y \in \varphi_x^{-1}(\tilde{x})$ we can not assure that we end up on the correct sheet (i.e. at y_P). For points $P = (x_P, y_P)$ such that $x_P \notin \mathcal{L}$ is not an exceptional value, we could, of course, integrate on all sheets simultaneously using Algorithm 4.7.1 and then choose the correct integral, but this would result in $g(m-1)$ unnecessary integrations. Moreover, if $x_P \in \mathcal{L}$, but P is a regular point it is mandatory to restrict the integration to the correct sheet.

It is advantageous to use the same integration method that has been used for the preceding period matrix computation (either (GL),(CC) or (DE)), so we can reuse the integration schemes (except when the number of required abscissas N is higher than for all previously computed schemes, then a new scheme has to be computed). If $P = (x_P, y_P)$ is very close to a critical point (see Example 4.7.6) it is useful to stick to the double-exponential method for integration along the line segment $[\tilde{\tilde{x}}, x_P]$.



(a) Regular point close to an exceptional value. (b) Regular point away from exceptional values.

Figure 4.8: Paths for the Abel-Jacobi map of $X : f_1 = 0$.

Complexity By the construction explained above, we need to integrate at most two paths, each on one sheet, for the Abel-Jacobi map of regular points. Our modification of Algorithm 4.7.2 (using Newton's method instead of simultaneous root approximation) results in following changes to the complexity:

- the cost for analytic continuation drops to $O(N(\log D + m)\mathcal{M}(D))$ operations,
- evaluating the differentials at the abscissas takes $O(NgD\mathcal{M}(D)) = O(Nd^3\mathcal{M}(D))$ operations,
- while addition of the individual values costs $O(NgD) = O(Nd^2\mathcal{M}(D))$ operations.

Adding together the correct values from the integrals corresponding to the path from x_0 to $\tilde{x}$ takes $O(ngD) = O(d^4D)$ operations. Therefore, we obtain

Theorem 4.9.2. *For each regular (i.e. finite, non-critical) point $P = (x_P, y_P) \in X$ we (heuristically) compute $\int_{P_0}^{P} \bar{\omega}$ to precision $D > 0$ using*

$$O(N_{min}(D)(\log D + d^3)\mathcal{M}(D) + d^4D + \mathrm{Init}(D)) \quad \textit{operations,}$$

where $N_{min}(D)$ and $\mathrm{Init}(D)$ are given by Table 3.6 and depend on the integration method.

Corollary 4.9.3. *For all but finitely many points $P = (x_P, y_P) \in X$, we (heuristically) compute $\int_{P_0}^{P} \bar{\omega}$ to precision $D > 0$ using*

$$O(c(f, x_P)D^2(\log D + d^3)\log^{1+\varepsilon} D + d^4D) \quad \textit{operations,}$$

where $c(f, x_P) = 1/r$, for some $r > 0$ (see §4.7.2), depends on the distances between x_P and the exceptional values (4.2).

Proof. Except for the factor $c(f, x_P)$, the claim follows from Theorem 4.9.2 using Clenshaw-Curtis quadrature, which implies that $\mathrm{Init}(D) = O(D^2 \log^{2+\varepsilon} D)$ and $N_{\min}(D) = O(D)$.

The points that are exempt from the claim are the points at infinity, y-infinite points and critical points, of which there are fininitely many. The factor $c(f, x_P)$ comes from the minimum of the values $r = r(\gamma, \mathcal{L})$ (see §4.7.2) for the (at most) two paths that have to be integrated. □

4.9.3 Integration into non-singular, critical points

Let now $P = (x_P, y_P)$ be a non-singular, critical point as defined in §4.1.2. There are several ways to compute the Abel-Jacobi map, i.e. to compute

$$\int_{P_0}^{P} \bar{\omega} \mod \Lambda$$

for critical points and here we want to present an elegant way to solve this problem. Recall that we have made the choice to compute with the morphism φ_x and that non-singular, critical points for φ_{x} are regular points for φ_{y}. Therefore, we can simply integrate the differentials with respect to the y-coordinate, i.e.

$$\begin{aligned}\int_{P_0}^{P} a(x,y)\mathrm{d}x &= \int_{P_0}^{P} \left(a(x,y)\frac{\mathrm{d}x}{\mathrm{d}y}\right)\mathrm{d}y \\ &= \int_{P_0}^{P} \tilde{a}(x,y)\mathrm{d}y = \int_{-1}^{1} \tilde{a}(\tilde{\gamma}(u))\gamma'(t)\mathrm{d}t\end{aligned}$$

where γ is a path in the y-plane connecting y_{P_0} and y_P, avoiding the exceptional values $\mathcal{L}(\varphi_y)$ and

$$\tilde{\gamma}(u) = (x_j(\gamma(u)), \gamma(u))$$

where $j \in \{1, \ldots, d_x\}$ and $\tilde{\gamma}(-1) = P_0$, $\tilde{\gamma}(1) = P$. If a non-singular point P is critical for $f(x, y)$ with respect to y, then it cannot be critical for $f(x, y)$ with respect to x. Hence, it will be a regular point and we can compute the Abel-Jacobi map of a regular point as described in §4.9.2 above.

In practice, we switch variables and compute with the affine model $f(y, x) = 0$ instead. Generally, x_{P_0} and x_P will not lie on the same sheet (over the y-plane), so we need the matrix T (4.33) (corresponding to $\varphi_{\mathrm{y}} : X \to \mathbb{P}^1$) to move between the sheets.

If one wants to compute the Abel-Jacobi map for all non-singular, critical points of φ_x, then the easiest solution is to execute the whole period matrix computation for φ_y, applying the algorithms described previously in this chapter. Once this is done we comfortably obtain the corresponding values using just a few extra integrations. As demonstrated in Section 4.8, our algorithms are quite fast, so this is a competitive approach for computing the Abel-Jacobi map of critical points to high precision.

4.9.4 Moving points by strong approximation

Here we describe a method that reduces the computation of the Abel-Jacobi map of non-regular points (with respect to the affine model) back to regular points, similar to §4.9.3. In fact, this is an application of Theorem 2.1 (the approximation theorem) or, alternatively, of Chow's moving lemma.

Although we can, in theory, avoid any finite set of points on X, this approach is particularly interesting for points at infinity, y-infinite points and finite singular points of the affine curve (see §4.1.2).

For this computation we rely heavily on our defining polynomial $f \in K[x, y]$ being defined over a number field. As for the holomorphic differentials in Section 4.2, we utilize the function field functionality of MAGMA here. Recall that the kernel of the Abel-Jacobi map consists of the principal divisors on the Riemann surface and therefore, for any

meromorphic function $b \in \mathbb{C}(X)$ in the function field and any divisor $D \in \mathrm{Div}(X)$, we have that

$$\mathcal{A}(D, P_0) \equiv \mathcal{A}(D + \mathrm{div}(b), P_0) \mod \Lambda\,.$$

We can utilize this to move unwanted points out of the divisor D by adding a suitable principal divisor D_0 and then compute $\mathcal{A}(D + D_0, P_0)$ instead.

For this we switch to the K-rational function field $K(C)$ of the smooth projective curve C/K to which $X = C(\mathbb{C})$ is associated. Let $Q_0 \in K(C)$ denote the place, usually of degree m, in the function field corresponding to the base point $P_0 = (x_0, y_1(x_0))_{>X}$. Here we have to choose $x_0 \in K$ (we may even choose $x_0 \in \mathbb{Z}$ without loss of generality). Further, let $Q \in K(C)$ be a place of degree $\deg(Q) = 1$ that corresponds to a point $P \in X$ for which we want compute the Abel-Jacobi map indirectly (a point at infinity $Q \in \varphi_x^{-1}(\infty)$, a y-infinite points or a finite singular point of C_f; essentially the cases that we have not already covered in this section). Moreover, denote by $\mathcal{S}$ the set of all of places in $K(C)$ that correspond to points (on the Riemann surface) which we want to avoid, including Q.

Note that we can only avoid points $P \in X$ for which $\varphi_{\mathrm{x}}(P) \in \mathbb{P}^1(\overline{K})$, i.e. there exists a place $Q \in K(C)$ lying over the corresponding place $P' \in \mathbb{C}(C)$. In practice, this does not pose a problem since our defining polynomial f is defined over K, so all exceptional values are defined over $\overline{K}$.

The following algorithm was suggested by Florian Hess and computes a suitable rational function $b \in K(C)$ such that $\mathrm{div}(b) = D_0$. The algorithm that MAGMA uses for the computation of Riemann-Roch spaces is also due to Florian Hess [45].

Algorithm 4.9.4. [Strong approximation] For a finite set of places S and a place $Q \in \mathcal{S}$, this algorithm computes a rational function $b \in K(C)$ such that $\mathrm{supp}(\mathrm{div}(b)) \cap \mathcal{S} = \{Q\}$ and $v_Q(\mathrm{div}(b)) = -1$.

(1) Define the divisor $D \leftarrow -\sum_{S \in \mathcal{S} \setminus \{Q\}} S$.

(2) While D is special, add Q_0 to it, i.e. $D \leftarrow D + k \cdot Q_0$ for some $k \geq 0$.

(3) Compute the Riemann-Roch space $L(D)$.

(4) For $S \in \mathcal{S}$ do

- **(4.1)** Compute the Riemann-Roch space $L(D + S)$.
- **(4.2)** Compute a generator $b_S \in K(C)$ of the quotient space $L(D + S)/L(D)$.

(5) Return $b = \sum b_S$.

Correctness

Proof. Recall from Definition 2.3.2 that for a non-special divisor we have that $l(D) = \dim(L(D)) = \deg(D) + 1 - g$. Repeatedly adding Q_0 to D will make D non-special, at latest when $\deg(D) > 2g-2$. If D is non-special, then so is $D+S$ and since $\deg(D+S) = \deg(D) + 1$ we know that $L(D + S)/L(D)$ is a 1-dimensional K-vector space that is generated by the class of any element $b_S \in L(D + S) \setminus L(D)$. In particular, we have that

$$\mathrm{div}(b_S) + D \not\geq 0 \quad \text{and} \quad \mathrm{div}(b_S) + D + S \geq 0\,,$$

which implies that for every $S \in \mathcal{S}$ we have

$$v_T(b_S) = \begin{cases} = 0, & \text{if } T = S \neq Q, \\ \geq 1, & \text{if } T \neq S \neq Q, \\ = -1, & \text{if } T = S = Q, \\ \geq 0, & \text{if } T \neq S = Q. \end{cases}$$

Using the strict triangle inequality for valuations yields that for $b = \sum b_S$ and $T \in S$

$$v_T(b) = \min_{S \in \mathcal{S}} \{v_T(b_S)\} = \begin{cases} 0, & \text{if } T \neq Q, \\ -1, & \text{if } T = Q, \end{cases}$$

as claimed. □

Suppose we want to calculate the Abel-Jacobi map of a point $P \in X$ that corresponds uniquely to a place $Q \in K(C)$, i.e. $\deg(Q) = 1$. Then, we can now use Algorithm 4.9.4 to obtain a rational function $b \in K(C)$ such that

$$\operatorname{supp}(Q + \operatorname{div}(b)) \cap \mathcal{S} = \varnothing \,,$$

for a finite set $\mathcal{S}$ of places of $K(C)$ that we want to avoid.

Now, we can translate the function field places appearing in the divisor $D = Q + \operatorname{div}(b)$ to regular points on X using the MAGMA function 'Conjugates'. Thus, we reduced the problem of computing Abel-Jacobi map $\mathcal{A}(P, P_0)$ to the cases described previously in this section. One major advantage of this approach is that the representation of points at infinity and singular finite points is taken care of by the function field and that we can easily access them, for example by applying the function 'Poles' to x or y for infinite points or applying 'Zeros' to the minimal polynomial of the x-coordinate of finite singular points.

Example 4.9.5. We consider the example of the Riemann surface X of genus $g = 2$ defined by the affine equation $f_7 = 0$ where

$$f_7 = -x^7 + 2x^3y + y^3 \in \mathbb{Q}[x, y] \,.$$

This example has already been considered by Frauendiener and Klein in [35]. It is particularly interesting, because the affine curve C_{f_7} is singular at $(0, 0)$ and its projective closure is singular at the point at infinity $[0 : 1 : 0]$. If we choose the base point $P_0 = [-2, -6.0855]$ and apply the algorithms of this chapter with respect to the ordering $>_X$, we see that $x_P = 0$ is a branch point with local monodromy $(1, 3)$ and ∞ is a branch point with local monodromy $(1, 3, 2)$. In the function field, as computed by MAGMA, the point $(0, 0)$ splits up into 2 points, represented over $\mathbb{Q}$ by the places

$$P_1 = (x, (x^4 - 1/2y^2)/x^3), \quad P_2 = (x, (x^3 + x^2y + 1/2y^2)/x^3),$$

while the point at infinity is represented by the place

$$P_\infty = (1/x, y^2/x^5),$$

and they all have degree one. We can hence use Algorithm 4.9.4 to find suitable principal divisors that help us compute the Abel-Jacobi map. In the first step, we apply the algorithm with $\mathcal{S} = \{P_1, P_2, P_\infty\}$ and $Q = P_\infty$ and obtain the function

$$b_\infty = \frac{x^3y + 2x^3 + 2y^2}{x^5 + 4x^4 + 4x^3} \quad \text{with divisor} \quad \operatorname{div}(b_\infty) = -P_\infty - 2Q_0 + P_3$$

and $P_3 = (x^7 + 8x^5 - 4x^4 + 4x^3 + 8, \frac{1}{33}(-4x^6 - 2x^5 - 33x^4 + 16x^3 - 8x^2 - 4x + 33y - 2))$ corresponds to 7 regular points on X. Knowing the Abel-Jacobi map for P_∞ already, we can apply Algorithm 4.9.4 with $\mathcal{S} = \{P_1, P_2\}$ and $Q = P_2$ and obtain the function

$$b_2 = (x^2 + y)/(x^3 + 2x^2)$$

with divisor

$$\operatorname{div}(b_2) = -P_2 + 2P_\infty - Q_0 + (x^2 + x + 2, -x + y - 2)$$

which allows us to obtain $\mathcal{A}(P_2, P_0)$ by computing $\mathcal{A}$ for the 2 regular points. In the last step we take as input $\mathcal{S} = \{P_1\}$ and $Q = P_1$, resulting in

$$b_1 = (2x^3 + y^2)/(x^5 + 2x^4)$$

with divisor

$$\operatorname{div}(b_1) = -P_1 + P_\infty - Q_0 + 3P_2\,.$$

Thus, recursively applying Algorithm 4.9.4 shifts the problem of computing $\mathcal{A}$ for P_∞, P_1, P_2 to computing it for 9 regular points on X, which is a great trade-off.

For the sake of comparing this method with adaptive double-exponential integration in §4.9.5, we give the corresponding values as elements of $(\mathbb{R}/\mathbb{Z})^4$:

$$\begin{aligned}
\mathcal{A}(P_\infty, P_0) &\equiv \begin{pmatrix}0.2857577928 & 0.09753205101 & 0.2857577928 & -0.1950641020\end{pmatrix}^T,\\
\mathcal{A}(P_1, P_0) &\equiv \begin{pmatrix}0.4857577928 & 0.4975320510 & 0.4857577928 & 0.004935897980\end{pmatrix}^T,\\
\mathcal{A}(P_2, P_0) &\equiv \begin{pmatrix}-0.3142422072 & -0.1024679490 & -0.3142422072 & 0.2049358980\end{pmatrix}^T.
\end{aligned}$$

Constant field extension We can easily find cases where unwanted points do not correspond to places of degree 1 in the function field $K(C)$. Suppose $P \in X$ is a point that corresponds to a place Q of degree $k > 1$ in $K(C)$. In order to successfully apply Algorithm 4.9.4, we have to compute the residue class field $\tilde{K}$ at Q and extend the constant field of $K(C)$ accordingly. In the field $\tilde{K}(C)$ there will be at least two places, say Q_1 and Q_2, corresponding to the point P, one of them having degree one, say Q_1. If Q_1 does not correspond to P, we repeat this process with the residue class field of $\tilde{K}(C)$ at Q_2, until we found the correct place (at most $k-1$ times). The set $\mathcal{S}$ has then to be chosen as subset of places of the final field extension and will become larger than before.

Example 4.9.6. Take for example any hyperelliptic curve $C : y^2 = p(x)$ of even degree such that the leading coefficient c_p of $p(x) \in \mathbb{Q}[x]$ is not a square in $\mathbb{Q}$. The projective closure $\overline{C}$ has two points at infinity that are singular and both correspond to one place at infinity P_∞ of degree 2 in the function field $\mathbb{Q}(C)$. Extending the constant field from $\mathbb{Q}$ to the number field $K = \mathbb{Q}[t]/(t^2 - c_p)$ solves the problem here, as each of the points at infinity will correspond to a unique place of degree 1 in the function field $K(C)$ and we can apply our algorithm for strong approximation.

Complexity We will not try and analyze the complexity of Algorithm 4.9.4 as it is an exact calculation and does not depend on the precision. Nonetheless, we remark that the algorithm becomes quickly impractical whenever the set $\mathcal{S}$ becomes too large. In theory, we could also use this algorithm for the Abel-Jacobi map of critical points, but in practice there are way too many of them. The set of finite singularities and points at infinity is usually small and thus well-suited for Algorithm 4.9.4. Another problem arises when the number fields involved are too complicated and arithmetic operations in the number field becomes costly. However, the version of the algorithm presented here is just a first sketch and could be heavily optimized.

4.9.5 Adaptive double-exponential integration

An alternative way of computing the integrals (4.32) for all $P = (x_P, y_P)$ lying over an exceptional value $x_P = x_i \in \mathcal{L}$ simultaneously is (brute force) numerical integration using the double-exponential method as a non-rigorous, adaptive scheme as explained in §3.4.1.

As for regular points (§4.9.2) we easily find a path from x_0 to x_P by following the closed path γ_i that encircles x_i (and has been constructed for the fundamental group)

starting from x_0 up to the starting point $\tilde{x}$ of the full circle and append a line segment $\gamma = [\tilde{x}, x_i]$ of length r_i. Therefore, the only integral matrix left to compute is

$$\int_\gamma \bar{\omega} \quad \text{where} \quad \gamma(1) = x_i \in \mathcal{L} \quad \text{is an exceptional value.}$$

If $\omega = a(x,y)\mathrm{d}x \in \Omega^1(X)$ is a holomorphic differential, then the definite integral

$$\int_{-1}^{1} a(\tilde{\gamma}(u))\gamma'(u)\mathrm{d}u \tag{4.34}$$

is well-defined (its value is a complex number), but the integrand $a(\tilde{\gamma}(u))$ cannot be evaluated at the end point $u = 1$ since $|a(\tilde{\gamma}(u))| \to \infty$ for $u \to 1$. These end point 'singularities' are not integrable, but 'polynomial' in the sense that $a(x,y)$ is a rational function and its denominator approaches infinity polynomially as u tends to 1.

Nonetheless, we can integrate (4.34) with the double-exponential integration scheme, as presented in §3.4.1, if

- for every step length $h > 0$ we compute N such that $|w_\ell a(\tilde{\gamma}(u_\ell))\gamma'(u_\ell)| \ll e^{-D}$ for $\ell > N$,
- we start with sufficiently high precision $\tilde{D} \gg D$ such that $\tanh(\lambda \sinh(Nh)) < 1$,
- and $a(\tilde{\gamma}(u_N))\gamma'(u) < e^{\tilde{D}}$.

In this way we can still evaluate $a(\tilde{\gamma}(u_N))\gamma'(u_N)$ in MAGMA to precision D. If x_i is a branch point the distance between at least two of the m values in the discrete lifts $\tilde{\gamma}(u)$ approaches zero as $u \to 1$. Similarly, if x_i is a root of the leading coefficient $a_0(x)$, at least one of these values tends to infinity as we approach the end point of γ. Consequently, analytic continuation using Algorithm 4.5.1 becomes problematic as the condition for guaranteed convergence (4.13) is no longer satisfied in a neighborhood of x_i. Although performing analytic continuation using Algorithm 4.5.1 to precision $\tilde{D}$ without checking condition (4.13) still works sufficiently well for most examples, it is no longer certified. Moreover, we will identify the continued fiber at the last abscissa $\tilde{\gamma}(u_N)$ (where all values of $y(x)$ are still different) with the ordering $>_X$, so that we can assign a permutation $\sigma_\gamma \in \mathrm{Sym(m)}$ to that path.

Points at infinity Similarly to finite points, we compute the Abel-Jacobi map for all points lying over $x = \infty \in \mathbb{P}^1$. Without loss of generality we can assume that we have chosen the 'left' base point (§4.3.1) such that $\mathrm{Re}(x_0) < 0$. Then a path from x_0 to ∞, which avoids exceptional values, is simply given by (4.12)

$$\gamma_{\mathrm{inf}} : u \mapsto \frac{2x_0}{(1-u)} \quad \text{and} \quad \gamma'_{\mathrm{inf}} : u \mapsto \frac{2x_0}{(1-u)^2} \tag{4.35}$$

and we can compute

$$\int_{-1}^{1} a(\tilde{\gamma_{\mathrm{inf}}}(u))\gamma'_{\mathrm{inf}}(u)\mathrm{d}u \tag{4.36}$$

by adaptive double-exponential integration (§3.4.1), using Algorithm 4.5.1 (without convergence condition) for heuristic analytic continuation. We run into similar problems as for finite exceptional values, because the values of $|\gamma_{\mathrm{inf}}(u)|$ and $|\gamma'_{\mathrm{inf}}(u)|$ at the last abscissa $u = u_N$ are quite big and we have to counteract this by choosing $\tilde{D} \gg D$ to prevent precision loss. Again, we assign a permutation $\sigma_{\gamma_{\mathrm{inf}}}$ by identifying $\tilde{\gamma_{\mathrm{inf}}}(u_N)$ with $>_X$.

Extending the ordering Using adaptive double-exponential integration, we are able to compute an integral matrix $\int_{x_0}^{x_P} \bar{\omega} \in \mathbb{C}^{g\times m}$ for every $x_P \in \mathbb{P}^1$. The method is completely heuristic and relies on the double-exponential decay of the weights $\{w_l\}$ to control the polynomial growth of the integrand towards the end point. It works perfectly well in practice, even for difficult examples, but it is hard to determine what integrals we have actually computed. Recall that we introduced the ordering $>_X$ on X that numbers the sheets uniquely from 1 to m above the base point x_0 and thus for all regular points. For the Abel-Jacobi map we need to artificially extend that ordering to other points as well. Points on X that are in one-to-one correspondence with a pair $(x, y) \in \mathbb{C}^2$ do not pose a problem here. Even if the fiber above x has less than m values, we can extend it to a list of m values (some of which occur repeatedly) and then order it with respect to the ordering $>_X$. This ordering is not unique, but we still get the correct value of the integral (4.9.3) modulo the period lattice (the Abel-Jacobi map is computed up to homology). Real problems arise when one wants to compute the Abel-Jacobi map of points that cannot be represented using an (x, y) pair, namely infinite points or singularities (finite or infinite). Instead, we can take as input a pair (x, s) where $x \in \mathbb{C} \cup \{\infty\}$ and $s \in \{1, \ldots, m\}$ and identify with this pair the s-th column of the integral matrix $\int_{x_0}^{x} \bar{\omega} + T \mod \Lambda \in (\mathbb{R}/\mathbb{Z})^{2g\times m}$ which defines a unique element in $\mathrm{Jac}(X)$.

Example 4.9.7 (cont.)**.** In Example 4.9.5 we computed the Abel-Jacobi map for the two points P_1 and P_2, that lie on $X : f_7 = 0$ above the finite singular point $(0, 0)$, and the point at infinity P_∞, by moving these points out of the support, using suitable principal divisors. We want to compare this with the results we get from applying adaptive double-exponential integration, as described above, to a path from x_0 to 0 and an infinite line γ_{inf} with $x_0 = -2$. Performing the integration, we obtain

$$\int_{x_0}^{0} \bar{\omega} + T \equiv \begin{pmatrix} -0.3142422072 & 0.4857577928 & -0.3142422072 \\ -0.1024679490 & 0.4975320510 & -0.1024679490 \\ -0.3142422072 & 0.4857577928 & -0.3142422072 \\ 0.2049358980 & 0.004935897980 & 0.2049358980 \end{pmatrix} \in (\mathbb{R}/\mathbb{Z})^{4\times 3},$$

$$\int_{x_0}^{\infty} \bar{\omega} + T \equiv \begin{pmatrix} 0.2857577928 & 0.2857577928 & 0.2857577928 \\ 0.09753205101 & 0.09753205101 & 0.09753205101 \\ 0.2857577928 & 0.2857577928 & 0.2857577928 \\ -0.1950641020 & -0.1950641020 & -0.1950641020 \end{pmatrix} \in (\mathbb{R}/\mathbb{Z})^{4\times 3}.$$

The double-exponential integration neither knows a representation of the points lying above 0 (or ∞), nor their properties. After adding the change of sheet matrix T (4.33) and reducing modulo the period lattice, we see that the results of the integration reflect the actual situation. There are two different points P_1 and P_2 that lie over 0 (corresponding to two different values of $\mathcal{A}$), while the sheets 1 and 3 are glued together (the local monodromy at 0 is $(1, 3)$). Similarly, integrating γ_{inf} on all $m = 3$ sheets leads to equal values modulo Λ indicating that there is only one point at infinity (in fact, the local monodromy at infinity is $(1, 3, 2)$). Furthermore, all the values agree with the values from Example 4.9.5 which were obtained by a completely different technique.

4.9.6 Reduction modulo the period lattice

For the Abel-Jacobi map to be well-defined, we have to reduce the vector integrals in $\mathbb{C}^g$ modulo the period lattice $\Lambda = \Omega\mathbb{Z}^{2g}$, where $\Omega = (\Omega_A, \Omega_B)$ is a big period matrix, computed as explained in Section 4.8.

Let $v = \int_P^Q \bar{\omega} \in \mathbb{C}^g$ be a vector obtained by integrating a basis of $\Omega^1(X)$. We identify $\mathbb{C}^g$ and $\mathbb{R}^{2g}$ via the bijection

$$\iota : v = (v_1, \ldots, v_g)^T \mapsto (\mathrm{Re}(v_1), \ldots, \mathrm{Re}(v_g), \mathrm{Im}(v_1), \ldots, \mathrm{Im}(v_g))^T.$$

Applying ι to the columns of Ω yields the invertible real matrix

$$\Omega_{\mathbb{R}} = \begin{pmatrix} \mathrm{Re}(\Omega_A) & \mathrm{Re}(\Omega_B) \\ \mathrm{Im}(\Omega_A) & \mathrm{Im}(\Omega_B) \end{pmatrix} \in \mathbb{R}^{2g \times 2g}.$$

Now, reduction of v modulo Λ corresponds bijectively to taking the fractional part of $\Omega_{\mathbb{R}}^{-1}\iota(v)$

$$v \bmod \Lambda \leftrightarrow \lfloor \Omega_{\mathbb{R}}^{-1}\iota(v) \rceil.$$

Complexity Reduction modulo the period lattice requires one initial $2g \times 2g$ matrix inversion to obtain $\Omega_{\mathbb{R}}^{-1}$ and afterwards one matrix multiplication for each vector that we need to reduce. Both can be done using $O(g^\eta)$ multiplications of precision D numbers.

4.9.7 Strategy for the Abel-Jacobi map

Let us briefly summarize our strategy for the Abel-Jacobi map here; suppose we want to compute the integral vector $\int_{P_0}^{P} \bar{\omega}$ for $P \in X$.

- If P is a finite, regular point we simply integrate as explained in §4.9.2.
- For all non-singular, critical points we may use the approach of §4.9.3. This is a good idea whenever computing with the morphism $\varphi_{\mathrm{y}} : X \to \mathbb{P}^1$ is not much harder than computing with φ_{x}.
- For all other points (this includes points at infinity, y-infinite points and singular points) that correspond to a place of small degree in the function field, we may apply our algorithm for strong approximation (see §4.9.4). This is particularly useful whenever P corresponds to a place of degree one and the set containing all 'bad' places is rather small.
- In the case where none of the aforementioned methods work, we can still apply adaptive (DE) integration (see §4.9.5) to obtain the integrals by means of numerical integration. This method works surprisingly well in practice, even for high genus, and is more or less independent of the geometry of the Riemann surface. The downside here is that this approach is, by its nature, completely heuristic.

4.9.8 Alternatives

Symbolic integration A promising way of computing the Abel-Jacobi map, in particular of critical points and points at infinity, are Puiseux series expansions. We discuss this possibility in more detail in Section 4.11.

Cauchy formula In their latest work [37] on compact Riemann surfaces, Frauendiener and Klein present a very nice approach for computing the Abel-Jacobi map that uses the Cauchy-formula for evaluation at critical and infinite points. They use Clenshaw-Curtis integration for the period matrix computation and compute the spectral coefficients for each differential and for each lift of each circle corresponding to a exceptional value. In this way, they can expand the integrands in terms of Chebyshev polynomials and obtain integrals along segments of these circles or in the interior by term-wise integration. This approach is very elegant and it avoids many of the problems that come with computing the Abel-Jacobi map above exceptional values and points at infinity. However, the authors rely heavily on the fact that they compute within a fixed precision environment on several occasions: they choose the same radius for all circles and take the same (fixed) number of abscissas N for all integrals. Thus, it is highly questionable whether generalizing their

approach to an arbitrary precision setting would be very efficient, especially because computing the spectral coefficients would involve multiplication of matrices of precision D and dimension $N \times N$. Moreover, choosing the same number of nodes for every integration (i.e. letting N be governed by the worst integral) seems like a horrible idea for a multi-precision implementation, while choosing different values of N would necessitate computing even more spectral coefficients.

4.10 Precision issues

In a multi-precision setting, knowledge about the size of the numbers appearing during the period matrix algorithm is indispensable to guarantee that the output is correct up to the requested precision $D > 0$. As already stated in §1.3, we will not take into account precision loss due to rounding errors or cancellation. Therefore, we cannot certify that the output is correct to the required precision, but this could be done using ball arithmetic. Heuristic estimates for the quality of the output were described in §4.8.1 and work perfectly well in practice.

Taking into account all predictable precision loss, we require $\tilde{D}_1(\varphi_x)$ (4.40) extra digits for analytic continuation and $\tilde{D}_2(\varphi_x, \bar{\omega})$ (4.42) extra digits for evaluating the differentials. Consequently, if we increase the internal precision from D to $\tilde{D} = D + \max(\tilde{D}_1, \tilde{D}_2)$ we can avoid precision loss caused by overflow occurring in the period matrix algorithm (Algorithm 4.8.1). In particular, we show that $\tilde{D}_1$ and $\tilde{D}_2$ are independent of the precision D, i.e. $\tilde{D} - D = O(1)$, as we heuristically assumed in §1.3.

These estimates can easily be modified for the computation of the Abel-Jacobi map as described in Section 4.9 (in the case of adaptive double-exponential integration §4.9.5 this cannot be done rigorously).

4.10.1 Bounding sizes of numbers

We want to find an upper bound for the absolute values of the complex numbers $x, y \in \mathbb{C}$ that will occur during our computations. This part is particularly important for the analytic continuation (see Section 4.5) of the paths making up the fundamental group (see Section 4.3). After embedding the coefficients of our defining polynomial $f(x, y)$ into the complex numbers, we are interested in bounding the values $|y_j(x)|$ for $j \in \{1, \dots, m\}$ in terms of $|x|$, see also [53, §2]. For this we view f as a univariate polynomial in $y_j(x)$, i.e. the affine equation for X as

$$f(x, y_j(x)) = \sum_{k=0}^{m} a_k(x) y_j(x)^{m-k} = 0.$$

Denote by $\mathcal{L}_0 := \{ z \in \mathbb{C} \mid a_0(z) = 0 \} \subset \mathcal{L}$ the set of zeros of the leading term $a_0(x)$ with respective multiplicities ν_z. For all $x \in \mathbb{C}$ such that $\operatorname{dist}(x, \mathcal{L}_0) > 0$, we can bound $|y_j(x)|$ by [53, Lemma 2.5] in terms of the coefficients

$$|y_j(x)| \leq 2 \max \left\{ \left| \frac{a_k(x)}{a_0(x)} \right|^{\frac{1}{k}} \mid k = 1, \dots, m \right\}.$$

Moreover, we bound the values of the polynomials $a_k(x) = \sum_{i=0}^{n_k} a_{ki} x^i \in \mathbb{C}[x]$ from above by the triangle inequality

$$|a_k(x)| \leq \sum_i |a_{ki}| \, |x|^i \leq \sum_{i=0}^{n_k} |a_{ki}| \, |B_x|^i =: A_k \quad \text{for all } |x| \leq B_x$$

where $B_x > 0$ is some constant, and $a_0(x)$ from below by

$$|a_0(x)| = |a_{0n_0}| \prod_{z \in \mathcal{L}_0} |x - z|^{\nu_z} \geq |a_{0n_0}| \, \rho^{n_0} \quad \text{for all } \operatorname{dist}(x, \mathcal{L}_0) \geq \rho > 0$$

such that we obtain

$$\max_{j=1,\ldots,m} |y_j(x)| \leq 2 \max \left\{ \left(\frac{A_k}{|a_{mn_m}| \, \rho^{n_m}} \right)^{\frac{1}{k}} \mid k = 0, \ldots, m-1 \right\} =: B_y \tag{4.37}$$

for all $|x| \leq B_x$ and $\operatorname{dist}(x, \mathcal{L}_0) \geq \rho$. Practical values for B_x and ρ emerge from the choices we made for the paths making up the fundamental group which are given in §4.3, namely

$$B_x := \max_{x \in \mathcal{L}} |x| + 1/4 \tag{4.38}$$

and

$$\rho := \min\{ \, r_i \mid x_i \in \mathcal{L}_0 \, \}$$

where the r_i are the radii chosen for the arcs and circles, see §4.3.1. Now, we can take into account the size of the numbers $x, y \in \mathbb{C}$ that satisfy our affine equation and will appear in Algorithm 4.5.1 by increasing the internal precision from D to $D + \tilde{D}_1$ where

$$\tilde{D}_1 := \lceil \log(\max\{B_y, A_0, \ldots, A_m\}) \rceil \, . \tag{4.39}$$

Besides the choices we get to make, B_x and B_y depend mainly on the quantities

$$\max\{ \, |x| \mid x \in \mathcal{L} \, \} \quad \text{and} \quad \min\{ \, \operatorname{dist}(x, \mathcal{L} \setminus \{x\}) \mid x \in \mathcal{L} \, \}$$

which are completely determined by the choice of the morphism $\varphi_x : X \to \mathbb{P}^1$ (this includes the coefficients of the defining polynomial $f(x, y)$). Henceforth, we denote

$$\tilde{D}_1(\varphi_x) := \tilde{D}_1 \tag{4.40}$$

4.10.2 Bounding differentials

The second source of predictable precision loss that cannot be neglected may appear during numerical integration of the differentials

$$\int_{\tilde{\gamma}} \omega = \int_{\tilde{\gamma}} a(x, y) \mathrm{d}x = \int_{-1}^{1} a(\tilde{\gamma}_j(u)) \gamma'(u) \mathrm{d}u$$

which was the topic of Section 4.7. This is closely connected to the bound M (§4.7.2) or the bounds M_1 and M_2 (§4.7.2) respectively. In the case of double-exponential (DE) integration, the bound $M_1 = M_1(\gamma, \bar{\omega})$ (4.30) is exactly the value we seek as it bounds the integrand on the interval $[-1, 1]$. We can thus take into account the predictable precision loss by defining

$$\tilde{D}_2 := \lceil \log(\max\{M_1(\gamma_1, \bar{\omega}), \ldots, M_1(\gamma_{3n-2}, \bar{\omega})\} \rceil \tag{4.41}$$

where the γ_i are the $3n - 2$ path pieces that make up the fundamental group computed by Algorithm 4.3.1.

In the case of Gauss-Legendre (and Clenshaw-Curtis) quadrature, the value of $M = M(\gamma, \bar{\omega}, r)$ bounds the differentials on the ellipse ε_r. Then, by the maximum modulus principle for holomorphic functions, M bounds the differentials on the interval $[-1, 1]$ as well, since it is contained in the ellipse ε_r. So we simply adapt the definition of $\tilde{D}_2$ with $M_1 = M$ in that case. Usually, the bound M on the ellipse will be much worse than on

the interval $[-1,1]$, so if we want to choose $\tilde{D}_2$ optimally, we would have to compute M_1 separately as required for (DE) integration.

The value of $\tilde{D}_2$ depends on the morphism $\varphi_{\mathrm{x}} : X \to \mathbb{P}^1$ and the basis of differentials. Therefore, we write

$$\tilde{D}_2(\varphi_{\mathrm{x}}, \bar{\omega}) := \tilde{D}_2 \,. \tag{4.42}$$

We already mentioned in Section 4.7 that we cannot rigorously compute such bounds in the current state of the MAGMA implementation, but they can be computed, for example, by using ball arithmetic as explained in §4.7.2. Note that the heuristics that we discussed in Section 4.7 reliably prevent precision loss in practice.

4.11 Symbolic integration

In this section we want to give a quick survey on an alternative way of integrating differential forms that is fundamentally different from numerical integration, as presented in Section 4.7. This approach is particularly useful when we want to integrate in the neighborhood of exceptional values.

Suppose, as before, that our Riemann surface X is given by an affine equation $f(x,y) = 0$ and that we project onto the x-coordinate using the morphism $\varphi_x : X \to \mathbb{P}^1$. The fiber ${\varphi_{\mathrm{x}}}^{-1}(\tilde{x})$ has less than $m = \deg(\varphi_{\mathrm{x}})$ elements for critical values $\tilde{x}$ (see §4.1.2). Above branch points the sheets of the Riemann surface are glued together and we cannot separate them easily. For singular points of the affine curve $C_f : f = 0$ it is even worse, as their coordinates may not even correspond to points on the Riemann surface. However, we can desingularize C_f locally by the transformation

$$x(t) = t^e + \tilde{x} \quad \text{with derivative} \quad \mathrm{d}x = et^{e-1}\mathrm{d}t \tag{4.43}$$

where e is the least common multiple of the ramification indices of the points lying over $\tilde{x}$. Afterwards, we can expand $y(x) = (y_1(x), \ldots, y_m(x))$ in terms of the local parameter t as Laurent series

$$y_j(t) = \sum_{i=i_0}^{\infty} a_{i,j} t^i \in \mathbb{C}((t)) \quad \text{for some } i_0 \in \mathbb{Z} \text{ and } j = 1, \ldots, m. \tag{4.44}$$

For the point at infinity $\infty \in \mathbb{P}^1_{\mathbb{C}}$ we can use the transformation

$$x(t) = t^{-e} \quad \text{with derivative} \quad \mathrm{d}x = -et^{-(e+1)}\mathrm{d}t$$

and then expand $y(t)$ as Laurent series (4.44). If we transform back via

$$t(x) = (x - \tilde{x})^{\frac{1}{e}} \quad \text{or} \quad t(x) = x^{-\frac{1}{e}} \quad \text{respectively,}$$

then $y_j(t(x))$, $j = 1, \ldots, m$ are exactly the *Puiseux series expansions* of $y(x)$ around $\tilde{x}$ (we refer the reader to [96] for a general introduction). The most common algorithm that is used to compute these expansion is called the *Newton-Puiseux algorithm*. The radius of convergence of these series is $\rho = \mathrm{dist}(\tilde{x}, \mathcal{L} \setminus \{\tilde{x}\})$, the distance to the nearest critical value (see [73, Lemma 1]). Consequently, in a disc $D(\tilde{x}; \rho)$ of radius ρ around $\tilde{x}$, the functions $y_j(t(x))$ are analytic. Thus, for any path $\gamma : [-1,1] \to D(\tilde{x}; \rho)$ the lifts of γ to the Riemann surface are given by

$$\tilde{\gamma_j} = \{(x, y_j(t(x))) \mid x \in \gamma([-1,1])\} \subset X.$$

Similar to Algorithm 4.5.1 these Puiseux series expansions can be used to determine the discrete lifts to precision D, as described in Section 4.5. Assume that for the series $y_j(t)$ we have $i_0 \geq 0$ (this is true whenever $\tilde{x} \neq \infty$ or $\tilde{x} \notin \mathcal{L}_0$) and denote by $y_j^N(t)$ the series truncated at $i = N$. The following Proposition provides an error term for the evaluation of this truncated series:

Proposition 4.11.1. *Let* $\beta = \left(\frac{|x-\tilde{x}|}{\rho}\right)^{\frac{1}{e}} < 1$ *and* $\max_{x\in D(\tilde{x},\rho)} |y_j(t(x))| \leq M$. *Then, for*

$$N+1 \geq \frac{D+\log(M)-\log(1-\beta)}{-\log(\beta)} \quad \text{we have} \quad \left|y_j(t(x)) - y_j^N(t(x))\right| \leq e^{-D}. \tag{4.45}$$

Proof. See [74, Proposition 34]. □

It follows that, asymptotically, we need to compute $N = O(D)$ terms of the series expansions $y_j(t)$ if we want obtain the fiber $y(x)$ to precision D in a neighborhood of $\tilde{x}$. From equation (4.45) we can exhibit that the distance $|x - \tilde{x}|$ and the radius of convergence ρ strongly influence the value of N.

4.11.1 Integration of differentials

Suppose now that we want to integrate a holomorphic differentials $\omega_i = a_i(x,y)\mathrm{d}x \in \Omega^1(X)$ along the line segment $\gamma : [\tilde{x}, z]$ with $z \in D(\tilde{x},\rho) \subset \mathbb{C}$ on all sheets $j \in \{1,\ldots,m\}$ simultaneously, i.e. we compute the integral matrix (4.24). Using the parametrizations (4.43) for $x(t)$ and (4.44) for $y(t)$, we get

$$\int_\gamma \omega_i = \int_{\tilde{x}}^{z} a_i(x,y)\mathrm{d}x = e\int_0^{(z-\tilde{x})^{\frac{1}{e}}} a_i(x(t),y_j(t))t^{e-1}\mathrm{d}t = e\int_0^{(z-\tilde{x})^{\frac{1}{e}}} \tilde{a}_{i,j}(t)\,\mathrm{d}t \tag{4.46}$$

where $\tilde{a}_{i,j} \in \mathbb{C}[[t]]$ is now a power series and can be integrated term-wise. For evaluation at $t(z)$ we can simply choose the principal branch of the e-th root. Once we have computed the power series $\tilde{a}_{i,j}$ for every $i = 1,\ldots,g$ and $j = 1,\ldots,m$ we obtain the matrix (4.24) by symbolic integration for every $z \in D(\tilde{x},\rho)$. Thus, we can use Puiseux expansions above a exceptional value $\tilde{x} \in \mathcal{L}$ for symbolic integration of holomorphic differentials. This could, for example, replace numerical integration along arcs and circles around exceptional values that were used for the period matrix algorithm and the Abel-Jacobi map.

The big advantage here is that every power series $\tilde{a}_{i,j} \in \mathbb{C}[[t]]$ (for each sheet and each differential) needs to be computed only once for the period matrix; we can then obtain the Abel-Jacobi map in the neighborhood of exceptional values at almost no additional cost by simply evaluating the corresponding series.

4.11.2 Practical issues and experiments

The approach that we indicated in the previous paragraph sounds very nice in theory, but there are some serious practical issues that would have to be overcome.

We experimented in MAGMA with a Puiseux series algorithm, due to Florian Hess, that computes the Laurent series (4.44) exactly as introduced in this section reliably for any $\tilde{x} \in \mathbb{P}^1$, in the function field setting where $f \in K[x,y]$ is defined over a number field K. The drawback is that arithmetic in the number fields that occur during this computation becomes very costly and it is not suited to compute $O(D)$ terms of such series. It would be interesting to optimize this by splitting up this algorithm into two parts: the exact part (computing the singular part of the Puiseux series) could be done over number fields, while expanding the series could be done over the complex numbers using Newton-iteration (in this way any algebraic function $f(x,y(x))$ can be computed in $O(mN\log N)$ if we use the FFT for polynomial multiplication [54]). Sadly, further investigation of this interesting approach was beyond the scope of this thesis. However, we implemented symbolic integration of differentials, as described in §4.11.1, and numerical experiments have shown that this approach does work, at least for simple examples and small precisions (we tested up to genus 4 and precision 50).

As suggested by Proposition 4.11.1, we want a small value β and large value of ρ, so it is important to expand above exceptional values and evaluate the series quite close to them such that N stays reasonably small. Naturally, this limits the method to small neighborhoods of exceptional values. Moreover, as experiments have indicated, once the rational functions $a_i(x, y)$ become more involved, computing the series approximations $\tilde{a}_{i,j}^N$ (4.46) is a numerical nightmare due to severe cancellation errors caused by evaluating polynomials at truncated Laurent series with $O(D)$ terms. It is questionable whether this problem can be solved, for example by careful and thorough implementation.

4.11.3 Existing work

In the context of compact Riemann surfaces, Puiseux series expansions possess a wide range of applications and have been utilized by many authors before.

- In their paper [24], Deconinck and Patterson actually compute the Abel-Jacobi map above exceptional values in MAPLE 'by computing series approximations of the holomorphic differentials' [24, Section 3.3], but they say nothing about precision issues or truncation orders of these series approximations or how they are computed.

- Frauendiener and Klein [35] use Puiseux series in their MATLAB implementation for the computation of holomorphic differentials. Due to the finite precision arithmetic in MATLAB, they quickly run into the expected precision issues, so they do not even attempt to actually perform symbolic integration, but switch to another method in [37] for the Abel-Jacobi map, see §4.9.8.

- In his thesis [74] Poteaux gave a symbolic-numeric algorithm for Puiseux expansions above critical values for plane algebraic curves defined by $f \in \mathbb{Z}[x, y]$; the algorithm is implemented in MAPLE and seems to work quite well in practice. Poteaux was not really interested in high precision evaluation of Puiseux expansions (see [73]), simply because he aimed at computing monodromy representations. He uses Puiseux expansions for analytic continuation which can be done at low precision. Nonetheless, it would be interesting to see whether his algorithm could be used for symbolic integration.

- An entirely different approach is given by Chudnovsky & Chudnovsky in their papers [17] and [18]. They describe an algorithm that computes Puiseux series expansions using differential equations. They focus on high precision evaluation and claim a complexity of $O(mN)$ for computing the first N terms. Having such an algorithm at our disposal would certainly be advantageous.

- Duval [31] gave an algorithm for rational Puiseux expansions, that is implemented in MAGMA. Since the exceptional values in our situation are generally not rational numbers, this is not really helpful.

4.12 Outlook

Throughout this chapter we mentioned possible extensions to our algorithms and methods and discussed different approaches. In this final section, we briefly summarize the most important improvements that could be realized in the future.

Extension to inexact defining polynomials It would certainly be nice to extend our algorithms to Riemann surfaces defined by irreducible polynomials $f \in \mathbb{C}[x, y]$ over the complex numbers. In particular, this requires computing the holomorphic differentials

without using MAGMA's function fields, as discussed in Section 4.2. Besides that only minor adjustments would have to be made for our period matrix algorithm to work. Another downside is that we could no longer use Algorithm 4.9.4 to avoid unwanted points for the Abel-Jacobi map, but there are several other options available that were discussed in detail in Section 4.9.

Rigorous implementation We already mentioned (§4.7.2, §4.7.4) that numerical integration of differential forms can be made rigorous, once ball arithmetic is available. As a consequence of this, our period matrix algorithm and the Abel-Jacobi map would become rigorous as well. This has been realized in the ARB [47] implementation of our algorithm for the superelliptic case, which is the topic of the next chapter. There is also a recent paper [48] by Johansson where he present his ARB implementation for rigorous arbitrary-precision numerical integration.

Symbolic integration As explained in Section 4.11, an efficient algorithm for high precision Puiseux series expansions would open up new possibilites, e.g. for analytic continuation, integration of differentials, the Abel-Jacobi map as well as local desingularization. However, serious issues would have to be overcome (see §4.11.2) and it is unclear whether this approach could offer any advantage in the range of hundred to a few thousand decimal digits.

Integration of meromorphic differentials So far, we restricted ourselves to integration of holomorphic differentials on compact Riemann surfaces. Using our strategy for numerical integration, described in Section 4.7, we could just as well integrate a meromorphic 1-form ω along paths on X that avoid the poles of ω. For example, abelian differentials of the second kind (see [7, §1.4.2] for a definition) are of interest in mathematics and physics [32]. In particular, this means that we have to extend the set of exceptional values $\mathcal{L}$ by the x-coordinates of the poles ω in order to compute the integration parameter r (see §4.7.2 and §4.7.3). In fact, we apply this strategy in §6.4.4 to compute a certain meromorphic differential, that is used to define the regulator pairing for the second K-group of algebraic curves.

Chapter 5

Computing period matrices & the Abel-Jacobi map: superelliptic case

This chapter is based on the paper 'Computing period matrices and the Abel-Jacobi map of superelliptic curves', that is joint work with Pascal Molin. It has been accepted by the journal Mathematics of Computation and is available on arχiv [65]. In many instances this chapter exceeds and refines the content of the original paper: we extended numerical integration by Gauss-Jacobi integration (§5.5.2) and compared its performance to double-exponential integration. Moreover, we added examples and timings for the Abel-Jacobi map (§5.7.2). Although the author carefully integrated the content of the paper into this thesis, it might still be noticeable that it was originally written in a very self-contained fashion. In contrast to the Chapters 3 and 4, we will analyze the complexity of our algorithms and methods not immediately following their appearance, but altogether in a unified setting in Section 5.6.1.

We still address the problem of computing period matrices and the Abel-Jacobi map, but in this chapter we restrict to the special case of algebraic curves given by an affine equation of the form (see Definition 5.1.1)

$$y^m = p(x), \quad m > 1, p \in \mathbb{C}[x] \text{ separable of degree } \deg(p) = n \geq 3. \tag{5.1}$$

They generalize hyperelliptic curves and we refer to them as *superelliptic curves*. Our main results for superelliptic curves have already been summarized in §1.4.2.

We remark that there is no clear definition of superelliptic curves in the literature and some authors will allow p to be non-separable in their definition. In this chapter, we rely on the fact that p has no multiple roots in several places. This restriction could be removed though, this is discussed in Section 5.8.

Remark 5.0.1 (Cyclic coverings of the projective line). Note that by [69, Theorem IV.3.2], over a number field that contains an m-th root of unity, all cyclic coverings of $\mathbb{P}^1$ have a model of the form $y^m = p(x)$ where $m > 1$ and each root of p has order $< m$. However, requiring that p is separable imposes the restriction that the cyclic covering has to be totally ramified at all but one branch point (which can be moved to infinity).

Rigorous implementation The algorithm (only period matrices) has been implemented in C using the ARB library [47]. This implementation is almost entirely due to Pascal Molin, with minor contributions from the author's side. This system represents complex numbers as floating point approximations plus an error bound, and automatically takes into account all precision loss occurring through the execution of the program. With

this model we can certify the accuracy of the numerical results of our algorithm, up to human or even compiler errors.

Another implementation, that includes a fully functioning Abel-Jacobi map, has been done in MAGMA. Both are publicly available on github at [66].

Structure of this chapter In Section 5.1 we give a very brief introduction to superelliptic curves and make explicit the ingredients to obtain a period matrix, namely a basis of holomorphic differentials and a homology basis. We give formulas for the computation of periods in Section 5.2 and explain how to obtain from them the standard period matrices using symplectic reduction. In Section 5.3 we give explicit formulas for the intersection numbers of our homology basis. The actual computation of the Abel-Jacobi map is explained in detail in Section 5.4. For numerical integration we employ two different integration schemes that are explained in Section 5.5: the double-exponential integration and Gauss-Jacobi quadrature, which is called Gauss-Chebychev quadrature in the special case of in the case of hyperelliptic curves. In Section 5.6 we analyze the complexity of our algorithm and share some insights on the implementation. Section 5.7 contains some tables with running times to demonstrate the performance of the code. Finally, in Section 5.8 we conclude with an outlook on what could be done in the future.

5.1 Superelliptic curves

5.1.1 Definition & properties

Definition 5.1.1. A complex superelliptic curve $C/\mathbb{C}$ is a smooth projective curve that has an affine model given by an equation of the form

$$\mathcal{C} : y^m = p(x) = c_p \cdot \prod_{k=1}^{n} (x - x_k), \tag{5.2}$$

where $m > 1$ and $p \in \mathbb{C}[x]$ is separable of degree $n \geq 3$. Note that we do not assume that $\gcd(m, n) = 1$.

There are $\delta = \gcd(m, n)$ points $P_\infty^{(1)}, \ldots, P_\infty^{(\delta)} \in C$ at infinity, that behave differently depending on m and n (see [85, §1] for details). In particular, $\infty \in \mathbb{P}^1 := \mathbb{P}^1_{\mathbb{C}}$ is a branch point for $\delta \neq m$. Thus, we introduce the set of finite branch points

$$\mathcal{B} = \{ x_1, \ldots, x_n \}$$

as well as the set of all branch points

$$\hat{\mathcal{B}} = \begin{cases} \mathcal{B} \cup \{\infty\}, & \text{if } m \nmid d, \\ \mathcal{B}, & \text{otherwise.} \end{cases} \tag{5.3}$$

The ramification indices at the branch points are given by $e_x = m$ for all $x \in \mathcal{B}$ and $e_\infty = \frac{m}{\delta}$. Using the Riemann-Hurwitz formula (2.3), we obtain the genus of C as

$$g = \frac{1}{2}((m-1)(n-1) - \delta + 1). \tag{5.4}$$

We denote the corresponding finite ramification points $P_k = (x_k, 0) \in \mathcal{C}$ for $k = 1, \ldots, n$.

Remark 5.1.2. Without loss of generality we may assume $c_p = 1$ (if not, apply the transformation $(x, y) \mapsto (x, \sqrt[m]{c_p} y)$).

Remark 5.1.3. For any $\begin{pmatrix} a & b \\ c & d \end{pmatrix} \in \mathrm{PSL}(2,\mathbb{C})$, the Moebius transform $\psi : u \mapsto \frac{au+b}{cu+d}$ is an automorphism of $\mathbb{P}^1$. By a change of coordinate $x = \psi(u)$ we obtain a different affine model of C given by the equation

$$\tilde{v}^m = \tilde{p}(u)$$

where $\tilde{p}(u) = p(\psi(u))(cu+d)^{\ell m}$ and $v = y(cu+d)^\ell$ for the smallest value ℓ such that $\ell m \geq n$. If the curve was singular at infinity, the singularity is moved to $u = -d/c$ in the new model. This happens when $\delta < m$ (so that $\ell m > n$). When $\delta = m$ we may apply such a transformation to improve the configuration of affine branch points.

5.1.2 Complex roots and branches of the curve

The complex m-th root

Working over the complex numbers we encounter several multi-valued functions which we will briefly discuss here. Closely related to superelliptic curves over $\mathbb{C}$ is the complex m-th root. Before specifying a branch it is a multi-valued function $y^m = x$ that defines an m-sheeted Riemann surface, whose only branch points are at $x = 0, \infty$, and these are totally ramified.

For $x \in \mathbb{C}$, it is natural and computationally convenient to use the *principal branch* of the m-th root $\sqrt[m]{x}$ defined by

$$-\frac{\pi}{m} < \arg(\sqrt[m]{x}) \leq \frac{\pi}{m}$$

which has a branch cut along the negative real axis $]-\infty, 0]$. Crossing it in positive orientation corresponds to multiplication by the primitive m-th root of unity

$$\zeta := \zeta_m := e^{\frac{2\pi i}{m}}.$$

on the surface. In particular, the monodromy at $x = 0$ is cyclic of order m.

The Riemann surface

In Chapter 2 we explained the connections between the theories of Riemann surfaces, algebraic curves and holomorphic covering maps. Here, we briefly summarize the situation in the case of complex superelliptic curves. Over $\mathbb{C}$ we can identify the curve C with the compact Riemann surface

$$X = C(\mathbb{C})\,.$$

Since our defining equation has the nice form $\mathcal{C} : y^m = \prod_{k=1}^n (x - x_k)$ it is compelling to do all computations in the x-plane. We denote by $\varphi_{\mathrm{x}} : X \to \mathbb{P}^1$ the corresponding holomorphic cyclic ramified covering of degree m of the projective line that is defined by the x-coordinate.

Exactly as in the general case, there are m possibilities to lift a path in x-plane to the Riemann surface $X = C(\mathbb{C})$ using analytic continuation, which is crucial for the integration of differentials on X. Due to the cyclic structure of superelliptic curves, these lifts are now related in a convenient way. We call a *branch of* $\mathcal{C}$ a function $y(x)$ such that $y(x)^m = p(x)$ for all $x \in \mathbb{C}$. At every x, the branches of $\mathcal{C}$ only differ by a factor ζ^l for some $l \in \{0, \ldots, m-1\}$. Thus, following a path, it is sufficient to know *one* branch that is analytic in a suitable neighborhood. In the next paragraph, we will introduce locally analytic branches very explicitly.

We obtain an ordering of the sheets relative to the analytic branches of $\mathcal{C}$ by imposing that multiplication by ζ, i.e. applying the map $(x, y(x)) \mapsto (x, \zeta y(x))$, corresponds to

moving one sheet up on the Riemann surface. The local monodromy of the covering φ_x is cyclic of order m and equal for all $x_k \in \mathcal{B}$ and the monodromy group is, up to conjugation, the cyclic group C_m. This makes it possible to find explicit generators for the homology group $H_1(X, \mathbb{Z})$ without specifying a base point, as shown in §5.1.3.

Locally analytic branches

In order to integrate differential forms on X it is sufficient to be able to follow *one* explicit analytic continuation of y along a path joining two branch points $a, b \in \mathcal{B}$.

One could of course consider the *principal branch* of the curve

$$y(x) = \sqrt[m]{p(x)}, \tag{5.5}$$

but this is not a good model to compute with: it has discontinuities along the curves $p^{-1}(]-\infty, 0])$, all wandering around the x-plane in an unpredictable way (see Figure 5.1a). These are the *branch cuts* of $y(x)$, crossing them in positive direction requires multiplying by ζ in order to follow an analytic continuation.

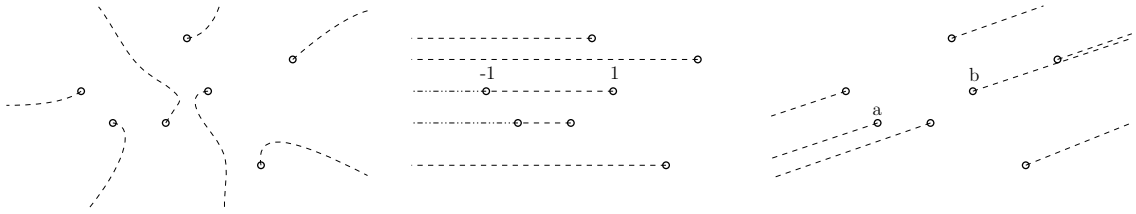
A better option is to split the product as follows: assume that $(a, b) = (-1, 1)$. Then the function

$$y(x) = \prod_{x_k \in \mathcal{B}} \sqrt[m]{x - x_k} \tag{5.6}$$

has n branch cuts parallel to the real line (see Figure 5.1b). However, one of them lies exactly on the interval $[-1, 1]$ we are interested in. We work around this by taking the branch cut towards $+\infty$ for each branch point x_k with positive real part, writing

$$y(x) = e^{\frac{i\pi r^+}{m}} \prod_{\mathrm{Re}(x_k) \le 0} \sqrt[m]{x - x_k} \prod_{\mathrm{Re}(x_k) > 0} \sqrt[m]{x_k - x},$$

where r^+ is the number of points with positive real part.



(a) principal branch (5.5) (b) product (5.6) (c) $y_{a,b}$ (5.9)

Figure 5.1: Branch cuts of different m-th roots.

In general we proceed in the same way: For branch points $a, b \in \mathcal{B}$ we consider the affine linear transformation

$$x_{a,b} : u \mapsto \frac{b-a}{2}\left(u + \frac{b+a}{b-a}\right),$$

which maps $[-1, 1]$ to the complex line segment $[a, b]$, and denote the inverse map by

$$u_{a,b} : x \mapsto \frac{2x - a - b}{b - a}.$$

We split the image of the branch points under $u_{a,b}$ into the following subsets

$$\{\, u_{a,b}(x) \mid x \in \mathcal{B} \,\} = \{\, -1, 1 \,\} \cup U^+ \cup U^-, \tag{5.7}$$

where points in U^+ (resp. U^-) have strictly positive (resp. non-positive) real part.

Then the product

$$\tilde{y}_{a,b}(u) = \prod_{u_k \in U^-} \sqrt[m]{u - u_k} \prod_{u_k \in U^+} \sqrt[m]{u_k - u} \tag{5.8}$$

is holomorphic on a neighborhood $\varepsilon_{a,b}$ of $[-1, 1]$ which we can take as an ellipse (such a neighborhood has been exhibited in 3.1) containing no point $u_k \in U^- \cup U^+$, while the term corresponding to a, b

$$\sqrt[m]{1 - u^2}$$

has two branch cuts $]-\infty, -1]$ and $[1, \infty[$, and is holomorphic on the complement $\overline{U}$ of these cuts.

We can now define a branch of the curve

$$y_{a,b}(x) = C_{a,b}\, \tilde{y}_{a,b}(u_{a,b}(x))\, \sqrt[m]{1 - u_{a,b}(x)^2} \tag{5.9}$$

by setting $r = 1 + \#U^+ \bmod 2$ and choosing the constant

$$C_{a,b} = \left(\frac{b-a}{2}\right)^{\frac{n}{m}} e^{\frac{\pi i}{m} r} \tag{5.10}$$

such that $y_{a,b}(x)^m = p(x)$ (any choice of the m-th root is valid here).

The function $y_{a,b}(x)$ has n branch cuts all parallel to $[a, b]$ in outward direction and is holomorphic inside $]a, b[$ (see Figure 5.1c). More precisely,

$$V_{a,b} = x_{a,b}(\varepsilon_{a,b} \cap \overline{U})$$

is an ellipse-shaped neighborhood of $]a, b[$ with two segments removed (see Figure 5.2) on which the local branch $y_{a,b}$ is well defined and holomorphic.

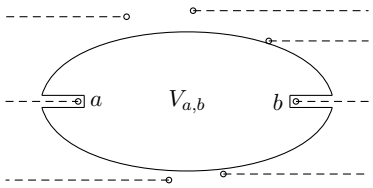


Figure 5.2: Holomorphic neighborhood of $y_{a,b}$.

We sum up the properties of these local branches:

Proposition 5.1.4. *Let $a, b \in \mathcal{B}$ be branch points such that $\mathcal{B} \cap]a, b[= \varnothing$. Then, with the notation as above, the functions $\tilde{y}_{a,b}$ (5.8) and $y_{a,b}$ (5.9) satisfy*

- *$\tilde{y}_{a,b}$ is holomorphic and does not vanish on $\varepsilon_{a,b}$,*
- *$y_{a,b}(x) = C_{a,b}\, \tilde{y}_{a,b}(u_{a,b}(x))\, \sqrt[m]{1 - u_{a,b}(x)^2}$ is holomorphic on $V_{a,b}$,*
- *$y_{a,b}(x)^m = p(x)$ for all $x \in \mathbb{C}$,*
- *$y_{a,b}(x), \zeta y_{a,b}(x), \ldots, \zeta^{m-1} y_{a,b}(x)$ are the m different analytic continuations of y on $V_{a,b}$.*

Moreover, we can assume that for $x \in V_{a,b}$, applying the map $(x, y_{a,b}(x)) \mapsto (x, \zeta^l y_{a,b}(x))$ corresponds to moving up $l \in \mathbb{Z}/m\mathbb{Z}$ sheets on the Riemann surface.

5.1.3 Cycles and homology

Recall that a *cycle* on X is a representative of a (homology) cycle class in $H_1(X,\mathbb{Z}) = \pi_1(X)/[\pi_1(X),\pi_1(X)]$, i.e. the homotopy class of a smooth oriented closed path on X. For simplicity we identify all cycles with their homology classes.

In the following we present an explicit generating set of $H_1(X,\mathbb{Z})$ that relies on the locally analytic branches $y_{a,b}$ as defined in (5.9) and the superelliptic structure of X.

Let $a, b \in \mathcal{B}$ be branch points such that $\mathcal{B} \cap]a,b[= \varnothing$, where $[a,b]$ is the oriented line segment connecting a and b.

By Proposition 5.1.4 the lifts of $[a,b]$ to X are given by

$$\gamma_{[a,b]}^{(l)} = \{\, (x, \zeta^l y_{a,b}(x)) \mid x \in [a,b] \,\}, \quad l \in \mathbb{Z}/m\mathbb{Z}.$$

Similarly, we obtain lifts of $[b,a]$ by reversing the orientation of $\gamma_{[a,b]}^{(l)}$. We denote

$$-\gamma_{[a,b]}^{(l)} = \{\, (x, \zeta^l y_{a,b}(x)) \mid x \in [b,a] \,\}, \quad l \in \mathbb{Z}/m\mathbb{Z}.$$

These are smooth oriented paths that connect $P_a = (a,0)$ and $P_b = (b,0)$ on X. We obtain cycles by concatenating these lifts in the following way:

$$\gamma_{a,b}^{(l)} = \gamma_{[a,b]}^{(l)} \cup -\gamma_{[a,b]}^{(l+1)} \in \pi_1(X). \tag{5.11}$$

Definition 5.1.5 (Elementary cycles)**.** We say $\gamma_{a,b} = \gamma_{a,b}^{(0)}$ is an *elementary cycle* and call $\gamma_{a,b}^{(l)}$ its *shifts* for $l \in \mathbb{Z}/m\mathbb{Z}$.

In $\pi_1(X)$ shifts of elementary cycles are homotopic to cycles that encircle a in negative and b in positive orientation, once each and do not encircle any other branch point. This is possible because we can always find an open neighborhood V of $[a,b]$ such that $V \cap \mathcal{B} = \{a,b\}$ and thus the homotopy class of a cycle is not changed by deformations within V. By definition of $y_{a,b}$ the branch cuts at the end points are outward and parallel to $[a,b]$. Thus, we have the following useful visualizations of $\gamma_{a,b}^{(l)}$ on X:

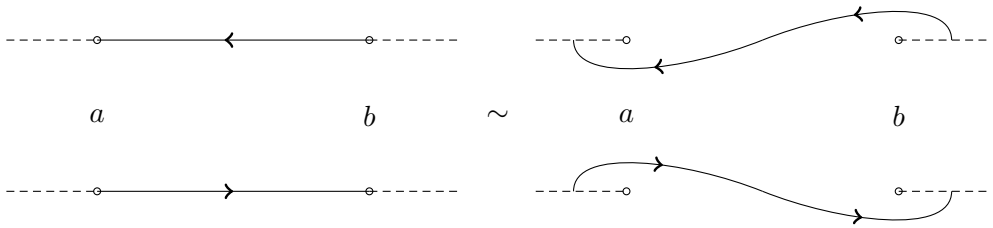


Figure 5.3: Homotopic representations of a cycle $\gamma_{a,b}^{(l)}$.

As it turns out, we do not need all elementary cycles and their shifts to generate $H_1(X,\mathbb{Z})$, but only those that correspond to edges in a *spanning tree*, that is a subset $E \in \mathcal{B} \times \mathcal{B}$ of directed edges (a,b) such that all branch points are connected without producing any cycle. It must contain exactly $n-1$ edges. The actual tree will be chosen in §5.2.3 in order to minimize the complexity of numerical integration.

For an edge $e = (a,b) \in E$, we denote by $\gamma_e^{(l)}$ the shifts of the corresponding elementary cycle $\gamma_{a,b}$.

Theorem 5.1.6. *Let E be a spanning tree for the branch points $\mathcal{B}$. The set of cycles $\Gamma = \left\{ \gamma_e^{(l)} \mid 0 \leq l < m-1, e \in E \right\}$ generates $H_1(X, \mathbb{Z})$.*

Proof. Denote by $\alpha_a \in \pi_1(\mathbb{P}^1 \setminus \hat{\mathcal{B}})$ a closed path that encircles the branch point $a \in \hat{\mathcal{B}}$ exactly once. Then, due to the relation $1 = \prod_{a \in \hat{\mathcal{B}}} \alpha_a$, $\pi_1(\mathbb{P}^1 \setminus \hat{\mathcal{B}})$ is freely generated by $\{\alpha_a\}_{a \in \mathcal{B}}$, i.e. in the case $\delta \neq m$ we can omit α_∞. Since our covering is cyclic, we have that

$$\pi_1(X \setminus {\varphi_x}^{-1}(\hat{\mathcal{B}})) \cong \ker(\pi_1(\mathbb{P}^1 \setminus \hat{\mathcal{B}}) \xrightarrow{\Phi} \mathrm{Aut}(X \setminus {\varphi_x}^{-1}(\hat{\mathcal{B}})))$$

where $\mathrm{Aut}(X \setminus {\varphi_x}^{-1}(\hat{\mathcal{B}})) \cong C_m \subset S_m$ and $\Phi(\alpha_a)$ is cyclic of order m for all $a \in \mathcal{B}$. Hence, for every word $\alpha = \alpha_1^{s_1} \dots \alpha_n^{s_n} \in \pi_1(\mathbb{P}^1 \setminus \hat{\mathcal{B}})$ we have that $\alpha \in \ker(\Phi) \Leftrightarrow \sum_{i=1}^n s_i \equiv 0 \bmod m$. We now claim that $\pi_1(X \setminus {\varphi_x}^{-1}(\hat{\mathcal{B}})) = \langle\, (-\alpha_a)^s \alpha_b^s,\, \alpha_a^m \mid s \in \mathbb{Z}, a, b \in \mathcal{B} \,\rangle$ and prove this by induction on n: for $\alpha = \alpha_1^{s_1}$, m divides s_1 and therefore α is generated by α_1^m. For $n > 1$ we write $\alpha = \alpha_1^{s_1} \dots \alpha_n^{s_n} = (\alpha_1^{s_1} \dots \alpha_{n-1}^{s_{n-1}+s_n})((-\alpha_{n-1})^{s_n} \alpha_n^{s_n})$.

We obtain the fundamental group of X as $\pi_1(X) \cong \pi_1(X \setminus {\varphi_x}^{-1}(\hat{\mathcal{B}}))/\langle\, \alpha_a^{e_a} \mid a \in \hat{\mathcal{B}} \,\rangle$, which is generated by $\{\, (-\alpha_a)^s \alpha_b^s \mid s \in \mathbb{Z}/m\mathbb{Z}, a, b \in \mathcal{B} \,\}$.

All branch points $a, b \in \mathcal{B}$ are connected by a path $(a, v_1, \dots, v_t, b)$ in the spanning tree, so we can write $(-\alpha_a)^s \alpha_b^s = ((-\alpha_a)^s \alpha_{v_1}^s)((-\alpha_{v_1})^s \alpha_{v_2}^s) \dots ((-\alpha_{v_{t-1}})^s \alpha_{v_t}^s)((-\alpha_{v_t})^s \alpha_b^s)$ and hence we have that $\{\, (-\alpha_a)^s \alpha_b^s \mid s \in \mathbb{Z}/m\mathbb{Z}, (a, b) \in E \,\}$ generates $\pi_1(X)$ and therefore $H_1(X, \mathbb{Z})$.

If we choose basepoints $p_0 \in \mathbb{P}^1 \setminus \hat{\mathcal{B}}$ for $\pi_1(\mathbb{P}^1 \setminus \hat{\mathcal{B}})$ and $P_0 \in {\varphi_x}^{-1}(p_0)$ for $\pi_1(X \setminus {\varphi_x}^{-1}(\hat{\mathcal{B}}))$ and $\pi_1(X)$ respectively, then, depending on the choice of P_0, for all $e = (a, b) \in E$ there exists $l_0 \in \mathbb{Z}/m\mathbb{Z}$ such that $\gamma_e^{(l_0)}$ is homotopic to $(-\alpha_a)\alpha_b$ in $\pi_1(X, P_0)$. In $H_1(X, \mathbb{Z})$ we have that $(-\alpha_a)^s \alpha_b^s = ((-\alpha_a)\alpha_b)^s$, so we obtain the other powers by concatenating the shifts $\prod_{l=0}^{s-1} \gamma_e^{(l_0+l)} = ((-\alpha_a)\alpha_b)^s$. This implies $1 = \prod_{l=0}^{m-1} \gamma_e^{(l_0+l)} = \prod_{l=0}^{m-1} \gamma_e^{(l)}$ and

$$\{\, (-\alpha_a)^s \alpha_b^s \mid s \in \mathbb{Z}/m\mathbb{Z} \,\} \subset \langle\, \gamma_e^{(l)} \mid 0 \leq l < m-1 \,\rangle,$$

and therefore $H_1(X, \mathbb{Z}) = \langle\, \Gamma \,\rangle$. □

Remark 5.1.7.

- For $\delta = 1$, we have that $\#\Gamma = (m-1)(n-1) = 2g$. Therefore, Γ is a basis for $H_1(X, \mathbb{Z})$ in that case.
- In the case $\delta = m$, the point at infinity is not a branch point. Leaving out one finite branch point in the spanning tree results in only $n-2$ edges. Hence, we easily find a subset $\Gamma' \subset \Gamma$ such that $\#\Gamma' = (m-1)(n-2) = 2g$ and Γ' is a basis for $H_1(X, \mathbb{Z})$.

5.1.4 Differential forms

The computation of the period matrix and the Abel-Jacobi map requires a basis of $\Omega^1(X)$ as a $\mathbb{C}$-vector space. In this section we provide a basis that only depends on m and n and is suitable for numerical integration.

Among the meromorphic differentials

$$\mathcal{W}^{\mathrm{mer}} = \{\, \omega_{i,j} \,\}_{1 \leq i \leq n-1,\, 1 \leq j \leq m-1} \quad \text{with} \quad \omega_{i,j} = \frac{x^{i-1}}{y^j} \mathrm{d}x,$$

there are exactly g that are holomorphic and they can be found by imposing a simple combinatorial condition on i and j. The following proposition is basically a more general version of [85, Proposition 2].

Proposition 5.1.8. *A $\mathbb{C}$-basis of the space of holomorphic differentials $\Omega^1(X)$ is given by*

$$\mathcal{W} = \{\, \omega_{i,j} \in \mathcal{W}^{mer} \mid -mi + jn - \delta \geq 0 \,\} \quad \text{where } \delta = \gcd(m,n).$$

Proof. First we show that the differentials in $\mathcal{W}$ are indeed holomorphic. For that let $\omega_{i,j} = x^{i-1}y^{-j}\mathrm{d}x \in \mathcal{W}^{\mathrm{mer}}$. We write down the relevant divisors

$$\begin{aligned}
\operatorname{div}(x) &= \sum_{k=1}^{m} \left(0, \zeta^k \sqrt[m]{f(0)}\right) - \frac{m}{\delta} \cdot \sum_{l=1}^{\delta} P_\infty^{(l)}, \\
\operatorname{div}(y) &= \sum_{k=1}^{n} P_k - \frac{n}{\delta} \cdot \sum_{l=1}^{\delta} P_\infty^{(l)}, \\
\operatorname{div}(\mathrm{d}x) &= (m-1)\sum_{k=1}^{n} P_k - \left(\frac{m}{\delta} + 1\right) \cdot \sum_{l=1}^{\delta} P_\infty^{(l)}.
\end{aligned}$$

Putting together the information, for $P \in X$ lying over $x_0 \in \mathbb{P}^1$, we obtain

$$v_P(\omega_{i,j}) = (i-1)v_P(x) + v_P(\mathrm{d}x) - jv_P(y) = \begin{cases} \geq 0, & \text{if } x_0 \neq x_k, \infty, \\ m-1-j \geq 0, & \text{if } x_0 = x_k, \\ \frac{(-mi-\delta+jn)}{\delta}, & \text{if } x_0 = \infty. \end{cases} \tag{5.12}$$

We conclude: $\omega_{i,j} \in \mathcal{W}^{\mathrm{mer}}$ is holomorphic if and only if $\omega_{i,j} \in \mathcal{W}$.

Since the differentials in $\mathcal{W}$ are clearly $\mathbb{C}$-linearly independent, it remains to show that there are enough of them, i.e. $\#\mathcal{W} = g$.

Counting the elements in $\mathcal{W}$ corresponds to counting lattice points $(i,j) \in \mathbb{Z}^2$ in the trapezoid given by the faces

$$\begin{aligned}
&1 \leq i \leq n-1, \\
&1 \leq j \leq m-1, \\
&i \leq \frac{n}{m}j - \frac{\delta}{m}.
\end{aligned}$$

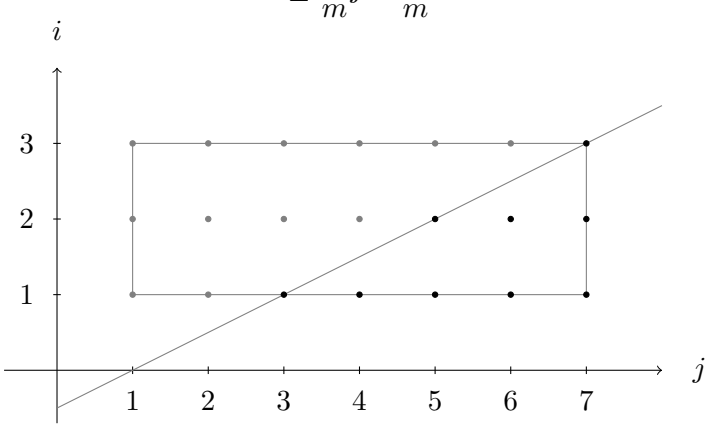


Figure 5.4: The points on and below the line correspond to holomorphic differentials. Illustrated is the case $n = 4, m = 8$, and thus $g = 9$.

Summing over the vertical lines of the trapezoid, we find the following formula that counts the points.

$$\#\mathcal{W} = \sum_{j=1}^{m-1} \left\lfloor \frac{n}{m}j - \frac{\delta}{m} \right\rfloor = \sum_{j=1}^{m-1} \frac{nj - \delta - r_j}{m} = \frac{n}{m}\sum_{j=1}^{m-1} j - \frac{m-1}{m}\delta - \frac{1}{m}\sum_{j=1}^{m-1} r_j, \tag{5.13}$$

where $r_j = nj - \delta \bmod m$.

The desired equality $\#\mathcal{W} = \frac{1}{2}((n-1)(m-1) - \delta + 1) = g$ immediately follows from

Lemma 5.1.9.

$$\sum_{j=1}^{m-1} r_j = \frac{1}{2}(m^2 - (\delta+2)m + 2\delta).$$

Proof. Let $l := \frac{m}{\delta}$. First we note that $r_j = r_{j+l}$:

$$r_{j+l} = n(j+l) - \delta \bmod m = nj + \frac{n}{\delta}m - \delta \bmod m = nj - \delta \bmod m = r_j,$$

and hence

$$\sum_{j=1}^{m-1} r_j = \delta \cdot \sum_{j=1}^{l} r_j - r_m = \delta \cdot \sum_{j=1}^{l} r_j - (-\delta + m). \tag{5.14}$$

Furthermore, r_j can be written as a multiple of δ:

$$r_j = \delta \left(\frac{n}{\delta} j - 1\right) \bmod m.$$

From $\gcd(\frac{n}{\delta}, l) = 1$ we conclude $\left\{ (\frac{n}{\delta} j - 1) \bmod l \mid 1 \le j \le l \right\} = \{0, \dots, l-1\}$. Therefore, we have

$$\sum_{j=1}^{l} r_j = \sum_{j=0}^{l-1} \delta j = \delta \cdot \frac{l(l-1)}{2}. \tag{5.15}$$

Finally, equations (5.14) and (5.15) imply

$$\sum_{j=1}^{m-1} r_j = \delta \cdot \sum_{j=1}^{l} r_j + \delta - m = \delta^2 \cdot \frac{l(l-1)}{2} + \delta - m = \frac{1}{2}(m^2 - (\delta+2)m + 2\delta).$$

□

Remark 5.1.10.

- Note that from equation (5.12) it follows that the meromorphic differentials in $\mathcal{W}^{\text{mer}}$ are homolorphic at all finite points.
- In practice we order the differentials in $\mathcal{W}$ lexicographically by j, i:

$$\omega_{i,j} < \omega_{\tilde{i},\tilde{j}} \quad \text{iff.} \quad j < \tilde{j} \text{ or } (j = \tilde{j} \text{ and } i < \tilde{i}).$$

5.2 Strategy for the period matrix

In this section we present our strategy to obtain period matrices $\Omega_\Gamma, \Omega_A, \Omega_B$ and τ as defined in §2.9.1. Although our superelliptic curves are not restricted to the case $\gcd(m, n) = 1$, we will briefly assume it in this paragraph to simplify notation.

The main ingredients were already described in Section 5.1: we integrate the holomorphic differentials in $\mathcal{W}$ (§5.1.4) over the cycles in Γ (§5.1.3) using numerical integration (§5.5), which results in a period matrix (§5.2.1)

$$\Omega_\Gamma = \left(\int_\gamma \omega\right)_{\substack{\omega \in \mathcal{W}, \\ \gamma \in \Gamma}} \in \mathbb{C}^{g \times 2g}.$$

The matrices Ω_A and Ω_B require a canonical basis of $H_1(X,\mathbb{Z})$. So, we compute the intersection pairing on Γ, as explained in Section 5.3, which results in a intersection matrix $K_\Gamma \in \mathbb{Z}^{2g\times 2g}$. After computing a symplectic base change $S \in \mathrm{GL}(\mathbb{Z}, 2g)$ for K_Γ (§5.2.4), we obtain a big period matrix

$$(\Omega_A, \Omega_B) = \Omega_\Gamma S, \tag{5.16}$$

and finally a small period matrix in the Siegel upper half-space

$$\tau = \Omega_A^{-1}\Omega_B \in \mathcal{H}_g. \tag{5.17}$$

5.2.1 Periods of elementary cycles

The following theorem provides a formula for computing the periods of the curve. It relates integration of differential forms on the Riemann surface to numerical integration in $\mathbb{C}$.

Note that the statement is true for all differentials in $\mathcal{W}^{\mathrm{mer}}$, not just the holomorphic ones. We continue to use the notation from Section 5.1.

Theorem 5.2.1. *Let $\gamma_e^{(l)} \in \Gamma$ be a shift of an elementary cycle corresponding to an edge $e = (a,b) \in E$. Then, for all differentials $\omega_{i,j} \in \mathcal{W}^{mer}$, we have*

$$\int_{\gamma_e^{(l)}} \omega_{i,j} = \zeta^{-lj}(1-\zeta^{-j})C_{a,b}^{-j}\left(\frac{b-a}{2}\right)^i \int_{-1}^{1} \frac{\varphi_{i,j}(u)}{(1-u^2)^{\frac{j}{m}}}\mathrm{d}u, \tag{5.18}$$

where

$$\varphi_{i,j} = \left(u + \frac{b+a}{b-a}\right)^{i-1} \tilde{y}_{a,b}(u)^{-j}$$

is holomorphic in a neighbourhood $\epsilon_{a,b}$ of $[-1,1]$.

Proof. By the definition in (5.11) we can write $\gamma_e^{(l)} = \gamma_{[a,b]}^{(l)} \cup \gamma_{[b,a]}^{(l+1)}$. Hence we split up the integral and compute

$$\begin{aligned}\int_{\gamma_{[a,b]}^{(l)}} \omega_{i,j} &= \int_{\gamma_{[a,b]}^{(l)}} \frac{x^{i-1}}{y^j}\mathrm{d}x = \zeta^{-lj}\int_a^b \frac{x^{i-1}}{y_{a,b}(x)^j}\mathrm{d}x \\ &= \zeta^{-lj}C_{a,b}^{-j}\int_a^b \frac{x^{i-1}}{\tilde{y}_{a,b}(u_{a,b}(x))^j(1-u_{a,b}(x)^2)^{\frac{j}{m}}}\mathrm{d}x.\end{aligned}$$

Applying the transformation $x \mapsto x_{a,b}(u)$ introduces the derivative $\mathrm{d}x = \left(\frac{b-a}{2}\right)\mathrm{d}u$ yields

$$\begin{aligned}\int_{\gamma_{[a,b]}^{(l)}} \omega_{i,j} &= \zeta^{-lj}C_{a,b}^{-j}\left(\frac{b-a}{2}\right)\int_{-1}^{1} \frac{x_{a,b}(u)^{i-1}}{\tilde{y}_{a,b}(u)^j(1-u^2)^{\frac{j}{m}}}\mathrm{d}u \\ &= \zeta^{-lj}C_{a,b}^{-j}\left(\frac{b-a}{2}\right)^i\int_{-1}^{1} \frac{\left(u+\frac{b+a}{b-a}\right)^{i-1}}{\tilde{y}_{a,b}(u)^j(1-u^2)^{\frac{j}{m}}}\mathrm{d}u\end{aligned}$$

Similarly, we obtain

$$\int_{\gamma_{[b,a]}^{(l+1)}} w_{i,j} = -\zeta^{-j}\int_{\gamma_{[a,b]}^{(l)}} w_{i,j}.$$

By Proposition 5.1.4 , $\tilde{y}_{a,b}$ is holomorphic and has no zero on $\epsilon_{a,b}$, therefore $\varphi_{i,j} = \left(u+\frac{b+a}{b-a}\right)^{i-1}\tilde{y}_{a,b}(u)^{-j}$ is holomorphic on $\epsilon_{a,b}$. □

5.2.2 Numerical integration

In order to compute a period matrix Ω_Γ the only integrals that have to be numerically evaluated are the *elementary integrals*

$$\int_{-1}^{1} \frac{\varphi_{i,j}(u)}{(1-u^2)^{\frac{j}{m}}} \mathrm{d}u \tag{5.19}$$

for all $\omega_{i,j} \in \mathcal{W}$ and $e \in E$. By Theorem 5.2.1, all the periods in Ω_Γ are then obtained by multiplication of elementary integrals with constants.

As explained in §5.6.3, the actual computations will be done on integrals of the form

$$I_{a,b}(i,j) = \int_{-1}^{1} \frac{u^{i-1}\mathrm{d}u}{(1-u^2)^{\frac{j}{m}} \tilde{y}_{a,b}(u)^j} \tag{5.20}$$

(that is, replacing $(u + \frac{b+a}{b-a})^{i-1}$ by u^{i-1} in the numerator of $\phi_{i,j}$), the value of elementary integrals being recovered by the polynomial shift

$$\int_{-1}^{1} \frac{\varphi_{i,j}(u)}{(1-u^2)^{\frac{j}{m}}} \mathrm{d}u = \sum_{l=0}^{i-1} \binom{i-1}{l} \left(\frac{b+a}{b-a}\right)^{i-1-l} I_{a,b}(l,j). \tag{5.21}$$

The rigorous numerical evaluation of (5.20) is addressed in Section 5.5: for any edge (a,b), Theorems 5.5.10 and 5.5.5 (or 5.5.1 if $m = 2$) provide explicit schemes allowing to attain any prescribed precision.

5.2.3 Minimal spanning tree

From the a priori analysis of all numerical integrals $I_{a,b}$ along the interval $[a,b]$, we choose an optimal set of edges forming a spanning tree as follows:

- Consider the complete graph on the set of finite branch points $G' = (\mathcal{B}, E')$ where $E' = \{\, (a,b) \mid a,b \in \mathcal{B} \,\}$.
- Each edge $e = (a,b) \in E'$ gets assigned a capacity r_e that indicates the cost of numerical integration along the interval $[a,b]$.
- Apply a standard 'maximal-flow' algorithm from graph theory, based on a greedy approach. This results in a spanning tree $G = (\mathcal{B}, E)$, where $E \subset E'$ contains the $n-1$ best edges for integration that connect all vertices without producing cycles.

Note that the integration process is most favorable between branch points that are far away from the others (this notion is made explicit in Section 5.5).

5.2.4 Symplectic reduction

A big period matrix (Ω_A, Ω_B) requires integration along a canonical basis of $H_1(X,\mathbb{Z})$. In §5.1.3 we gave a generating set Γ for $H_1(X,\mathbb{Z})$, namely

$$\Gamma = \left\{ \gamma_e^{(l)} \mid 0 \le l < m-1, e \in E \right\},$$

where E is the spanning tree chosen above. This generating set is in general not a (canonical) basis.

We resolve this by computing the intersection pairing on Γ, that is all intersections $\left(\gamma_e^{(k)} \circ \gamma_{e'}^{(l)}\right) \in \{0, \pm 1\}$ for $e, e' \in E$ and $k, l \in \{0, \dots m-1\}$, as explained in Section 5.3.

The resulting intersection matrix K_Γ is a skew-symmetric matrix of dimension $(n-1)(m-1)$ and has rank $2g$. We then compute a symplectic base change matrix S, exactly as we did in §4.6.2 for the general case.

5.3 Intersections

Let (a,b) and (c,d) be two edges of the spanning tree E. The formulas in Theorem 5.3.1 allow to compute the intersection between shifts of elementary cycles $\left(\gamma_{a,b}^{(k)} \circ \gamma_{c,d}^{(l)}\right)$.

Note that by construction of the spanning tree, we can restrict the analysis to intersections $\left(\gamma_{a,b}^{(k)} \circ \gamma_{c,d}^{(l)}\right)$ such that c is either a or b. Moreover, we may discard the case $(a,b) = (c,d)$.

Theorem 5.3.1 (Intersection numbers)**.** *Let $(a,b),(c,d) \in E$. The intersections of the corresponding cycles $\gamma_{a,b}^{(k)}, \gamma_{c,d}^{(l)} \in \Gamma$ are given by*

$$\left(\gamma_{a,b}^{(k)} \circ \gamma_{c,d}^{(l)}\right) = \begin{cases} 1 & \text{if } l-k \equiv s_+ \bmod m, \\ -1 & \text{if } l-k \equiv s_- \bmod m, \\ 0 & \text{otherwise,} \end{cases}$$

where s_+, s_- are given by the following table, which covers all cases occurring in the algorithm

	case	s_+	s_-
(i)	$a=c$ and $b=d$	1	-1
(ii)	$b=c$	$-s_b$	$1-s_b$
(iii)	$a=c$ and $\varphi>0$	$1-s_a$	$-s_a$
(iv)	$a=c$ and $\varphi<0$	$-s_a$	$-1-s_a$
(v)	$\{a,b\}\cap\{c,d\}=\varnothing$	no intersection	

and where $s_x \in \mathbb{Z}$ for $x \in \{a,b\}$ is given by

$$s_x := \frac{1}{2\pi}\left(\varphi + m \cdot \arg\left(\frac{C_{c,d}\tilde{y}_{c,d}(x)}{C_{a,b}\tilde{y}_{a,b}(x)}\right)\right)$$

and

$$\varphi = \arg\left(\frac{b-a}{d-c}\right) + \delta_{b=c}\pi.$$

Remark 5.3.2. Note that the intersection matrix K_Γ is composed of $(n-1)^2$ blocks of dimension $m-1$, each block corresponding to the intersection of shifts of two elementary cycles in the spanning tree. It is very sparse.

The proof of Theorem 5.3.1 is contained in the following exposition.

Consider two cycles $\gamma_{a,b}^{(k)}, \gamma_{c,d}^{(l)} \in \Gamma$ and recall from Definition 5.1.5 that

$$\gamma_{a,b}^{(k)} = \{\,(x, \zeta^k y_{a,b}(x)) \mid x \in [a,b]\,\} \cup \{\,(x, \zeta^{k+1} y_{a,b}(x)) \mid x \in [b,a]\,\},$$
$$\gamma_{c,d}^{(l)} = \{\,(x, \zeta^l y_{c,d}(x)) \mid x \in [c,d]\,\} \cup \{\,(x, \zeta^{l+1} y_{c,d}(x)) \mid x \in [c,d]\,\},$$

where $\zeta^k y_{a,b}(x), \zeta^l y_{c,d}(x)$ are branches of $\mathcal{C}$ that are analytic on open sets $V_{a,b}$ and $V_{c,d}$ (see Figure 5.2) respectively.

From the definition we see that $\gamma_{a,b}^{(k)} \cap \gamma_{c,d}^{(l)} = \varnothing$, whenever $[a,b] \cap [c,d] = \varnothing$. For edges in a spanning tree this is equivalent to $\{a,b\} \cap \{c,d\} = \varnothing$, thus proving (v).

Henceforth, we can assume $\{a,b\} \cap \{c,d\} \neq \varnothing$. In order to prove (i)-(iv) we have to introduce some machinery. Since the $y_{a,b}(x), y_{c,d}(x)$ are branches of $\mathcal{C}$, on the set $\mathbb{C} \setminus \mathcal{B}$ we can define the *shifting function* $s(x)$, that takes values in $\mathbb{Z}/m\mathbb{Z}$, implicitly via

$$\zeta^{s(x)} = \frac{y_{c,d}(x)}{y_{a,b}(x)}. \tag{5.22}$$

Naturally, (5.22) extends to the other analytic branches via

$$\zeta^{s(x)+l-k} = \frac{\zeta^l y_{c,d}(x)}{\zeta^k y_{a,b}(x)}.$$

We can now define the non-empty, open set

$$V := V_{a,b} \cap V_{c,d} \subset \mathbb{C} \setminus \mathcal{B}.$$

The shifting function $s(x)$ is well-defined on V and, since $y_{a,b}(x)$ and $y_{c,d}(x)$ are both analytic on V, $s(x)$ is constant on its connected components.

In §5.1.2 we established that multiplication of a branch by ζ corresponds to moving one sheet up on the Riemann surface. We can interpret the value of the shifting function geometrically as $\gamma_{c,d}^{(l)}$ running $s(\tilde{x}) + l - k$ sheets above $\gamma_{a,b}^{(k)}$ at a point $\tilde{x} \in V$.

This can be used to determine the intersection number in the following way. The homotopy class of a cycle on X is not changed by deformations that avoid encircling additional branch points. Since $V \cap \mathcal{B} = \varnothing$ we can deform the cycles homotopically (as shown in Figure 5.3) such that

$$\varphi_{\mathrm{x}}\left(\gamma_{a,b}^{(k)}\right) \cap \varphi_{\mathrm{x}}\left(\gamma_{c,d}^{(l)}\right) = \{\tilde{x}\} \text{ for some } \tilde{x} \in V.$$

Consequently, the cycles can at most intersect at the points in the fiber above $\tilde{x}$, i.e.

$$\gamma_{a,b}^{(k)} \cap \gamma_{c,d}^{(l)} \subset {\varphi_{\mathrm{x}}}^{-1}(\tilde{x}).$$

Note that, by definition, any cycle in Γ only runs on two neighbouring sheets, which already implies

$$\left(\gamma_{a,b}^{(k)} \circ \gamma_{c,d}^{(l)}\right) = 0, \text{ if } s(\tilde{x}) + l - k \notin \{-1, 0, 1\}.$$

In the other cases we can determine the sign of possible intersections by taking into account the orientation of the cycles.

We continue the proof with case (i): Here we have $[a, b] = [c, d]$. Trivially, $\left(\gamma_{a,b}^{(k)} \circ \gamma_{a,b}^{(k)}\right) = 0$ holds. For $k \neq l$ we deform the cycles such that they only intersect above $\tilde{x} = \frac{b+a}{2} \in V_{a,b} = V$. We easily see that $s(\tilde{x}) = 0$ and therefore $s(\tilde{x}) + l - k = l - k$. The remaining non-trivial cases ($l = k \pm 1$), are shown in Figure 5.5 below where the cycles $\gamma_{a,b}^{(k)}$ (black), $\gamma_{a,b}^{(k+1)}$ (red) and $\gamma_{a,b}^{(k-1)}$ (green) are illustrated.

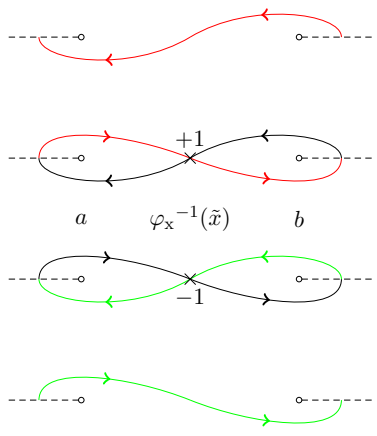


Figure 5.5: Intersections of self-shifts.

We see that, independently of $s(\tilde{x})$, $s_+ = (k+1) - k = 1$ and $s_- = (k-1) - k = -1$ are as claimed.

For (ii)-(iv) we have that $[a,b] \cap [c,d] = \{c\}$, where c is either a or b. Unfortunately, in these cases $s(c)$ is not well-defined.

Instead, we choose a point $\tilde{x} \in \mathbb{C} \setminus \mathcal{B}$ on the bisectrix of $[a,b]$ and $[c,d]$ that is close enough to c such that $[\tilde{x}, c[\subset V = V_{a,b} \cap V_{c,d}$ (see Figure 5.6 below), and where

$$s(\tilde{x}) = \frac{m}{2\pi} \arg\left(\frac{y_{c,d}(\tilde{x})}{y_{a,b}(\tilde{x})}\right). \tag{5.23}$$

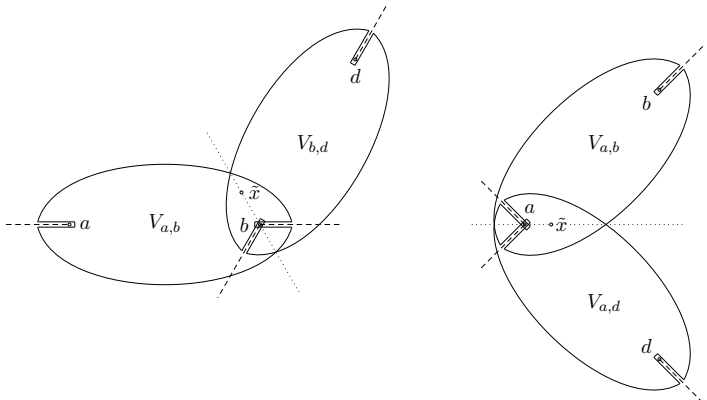


Figure 5.6: The set $V = V_{a,b} \cap V_{c,d}$ for $b = c$ (left) and $a = c$ (right).

Case (ii):
In this case we have $b = c$. Choosing $\tilde{x}$ on the upper bisectrix (as shown in Figure 5.6) and computing $s(\tilde{x})$ with (5.23) makes it possible to determine the intersection numbers geometrically.

Figure 5.7 shows the non-trivial cases $s(\tilde{x}) + l - k \in \{-1, 0, 1\}$. There the cycles $\gamma_{a,b}^{(k)}$ (black), $\gamma_{b,d}^{(k-s(\tilde{x}))}$ (gray), $\gamma_{b,d}^{(k-s(\tilde{x})+1)}$ (green) and $\gamma_{b,d}^{(k-s(\tilde{x})-1)}$ (red) are illustrated.

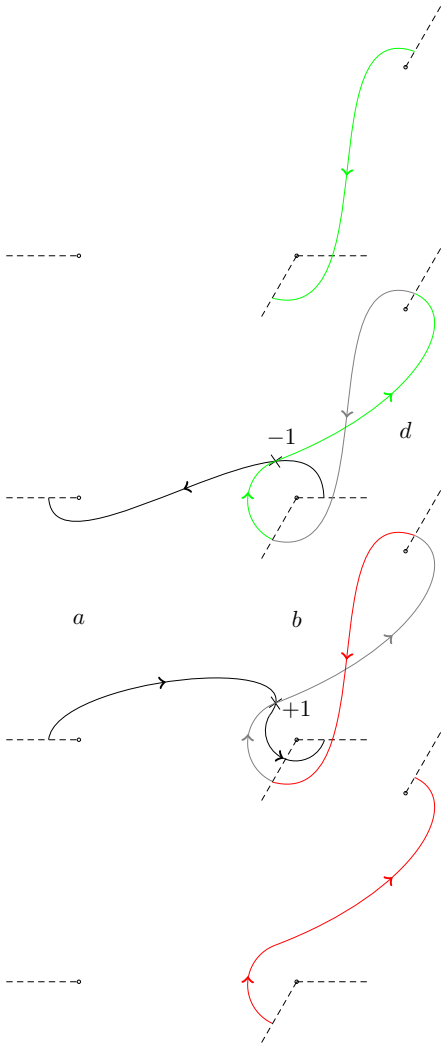


Figure 5.7: Intersections for $b = c$.

By Lemma 5.3.3 (1) we have $s(\tilde{x}) \equiv s_b$, which implies (as claimed)

$$\begin{aligned} s_+ &\equiv k - s(\tilde{x}) - k \equiv -s_b \mod m, \\ s_- &\equiv k - s(\tilde{x}) + 1 - k \equiv 1 - s_b \mod m. \end{aligned}$$

Case (iii):
In this case we have $a = c$. We choose $\tilde{x}$ on the inner bisectrix (as shown in Figure 5.6) and compute $s(\tilde{x})$ with (5.23).

For $\varphi = \arg\left(\frac{b-a}{d-a}\right) > 0$, the non trivial cases, i.e. $s(\tilde{x}) + l - k \in \{-1, 0, 1\}$, are shown in Figure 5.8 We illustrate the cycles $\gamma_{a,b}^{(k)}$ (black), $\gamma_{a,d}^{(k-s(\tilde{x}))}$ (gray), $\gamma_{a,d}^{(k-s(\tilde{x})+1)}$ (green) and $\gamma_{a,d}^{(k-s(\tilde{x})-1)}$ (red).

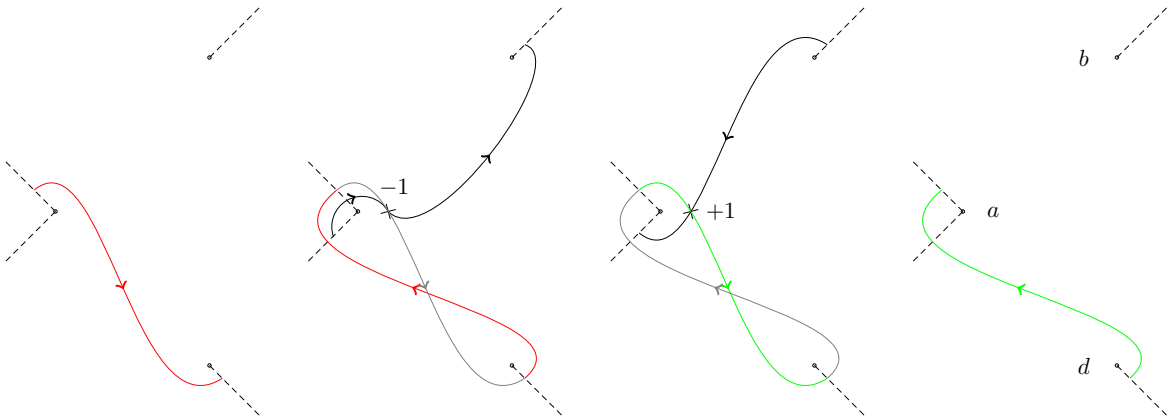


Figure 5.8: Intersections for $a = c$ and $\varphi > 0$.

Lemma 5.3.3 (2) gives us $s(\tilde{x}) \equiv s_a$, which implies (as claimed for $\varphi > 0$)

$$s_+ = k - s(\tilde{x}) + 1 - k \equiv 1 - s_a \mod m,$$
$$s_- = k - s(\tilde{x}) - k \equiv -s_a \mod m.$$

Case (iv):
When $\varphi < 0$ we can use the antisymmetry of the intersection pairing to fall back to case (iii) by looking the intersection of the swapped cycles

$$\left(\gamma_{a,b}^{(k)} \circ \gamma_{a,d}^{(l)}\right) = -\left(\gamma_{a,d}^{(l)} \circ \gamma_{a,b}^{(k)}\right).$$

The intersection on the right is then determined by case (iii) with the quantities $\varphi' = -\varphi > 0$, $s'_a = -s_a$ and $s'_\pm = s_\mp$.

Alternatively, we can see this directly from the picture: if we mirror Figure 5.8 at the horizontal line through a we are in case (iv). There, the intersection is positive if $\gamma_{a,b}^{(k)}$ and $\gamma_{a,d}^{(l)}$ start on the same sheet and negative if $\gamma_{a,d}^{(l)}$ starts one sheet below $\gamma_{a,b}^{(k)}$.

Lemma 5.3.3. *With the choices made in the proof of Theorem 5.3.1 the following statements hold*

(1) $s(\tilde{x}) \equiv s_b \mod m$ *in case* (ii),

(2) $s(\tilde{x}) \equiv s_a \mod m$ *in the case* (iii).

Proof. Starting from equation (5.23), for all $x \in \mathbb{C} \setminus \mathcal{B}$ we have

$$\begin{aligned} s(x) &= \frac{m}{2\pi} \arg\left(\frac{y_{c,d}(x)}{y_{a,b}(x)}\right) \equiv \frac{m}{2\pi}\left(\arg\left(\frac{(1-u_{c,d}(x)^2)^{\frac{1}{m}}}{(1-u_{a,b}(x)^2)^{\frac{1}{m}}}\right) + \arg\left(\frac{C_{c,d}\tilde{y}_{c,d}(x)}{C_{a,b}\tilde{y}_{a,b}(x)}\right)\right) \\ &\equiv \frac{1}{2\pi}\left(\arg(1+u_{c,d}(x)) + \arg(1-u_{c,d}(x)) - \arg(1+u_{a,b}(x)) - \arg(1-u_{a,b}(x))\right) \\ &\quad + \frac{m}{2\pi}\left(\arg\left(\frac{C_{c,d}\tilde{y}_{c,d}(x)}{C_{a,b}\tilde{y}_{a,b}(x)}\right)\right) \mod m. \end{aligned}$$

In case (ii) we have $b = c$ and denote $\varphi_0 = \arg\left(\frac{b-a}{d-b}\right)$. Then, we can parametrize all points $\tilde{x} \neq b$ on the upper bisectrix of $[a, b]$ and $[a, d]$ (see Figure 5.6) via

$$\tilde{x} = x_{b,d}(-1 + t\exp(i(\pi + \varphi_0)/2)) \text{ as well as}$$
$$\tilde{x} = x_{a,b}(1 - t\exp(-i(\pi + \varphi_0)/2))$$

for some $t > 0$, where $x_{b,d}$ and $x_{a,b}$ are defined as in (5.1.2). Therefore,

$$\arg(1 + u_{b,d}(\tilde{x})) = \frac{\pi + \varphi_0}{2} \text{ and}$$
$$\arg(1 - u_{a,b}(\tilde{x})) = -\frac{\pi + \varphi_0}{2}.$$

For $\tilde{x}$ chosen close enough to b we have that $[\tilde{x}, b[\subset V$ and the shifting function $s(\tilde{x})$ is constant as $\tilde{x}$ tends towards b. Hence, we can compute its value at $\tilde{x}$ as

$$\begin{aligned} s(\tilde{x}) &\equiv \frac{1}{2\pi}\left(\pi + \varphi_0 + \arg(1 - u_{b,d}(\tilde{x})) - \arg(1 + u_{a,b}(\tilde{x})) + m\arg\left(\frac{C_{b,d}\tilde{y}_{b,d}(\tilde{x})}{C_{a,b}\tilde{y}_{a,b}(\tilde{x})}\right)\right) \\ &\equiv \frac{1}{2\pi}\left(\varphi + \arg(1 - u_{b,d}(b)) - \arg(1 + u_{a,b}(b)) + m\arg\left(\frac{C_{b,d}\tilde{y}_{b,d}(b)}{C_{a,b}\tilde{y}_{a,b}(b)}\right)\right) \\ &\equiv \frac{1}{2\pi}\left(\varphi + \arg(2) - \arg(2) + m\arg\left(\frac{C_{b,d}\tilde{y}_{b,d}(b)}{C_{a,b}\tilde{y}_{a,b}(b)}\right)\right) \equiv s_b \mod m, \end{aligned}$$

thus proving (1).

In the cases (iii) and (iv) we have $a = c$ and denote $\varphi = \arg\left(\frac{b-a}{d-a}\right)$. For $\varphi > 0$ we can parametrize all points $\tilde{x} \neq a$ on the inner bisectrix of $[a, b]$ and $[a, d]$ (see Figure 5.6) via

$$\tilde{x} = x_{a,d}(-1 + t\exp(i\varphi/2)) \text{ as well as}$$
$$\tilde{x} = x_{a,b}(-1 + t\exp(-i\varphi/2))$$

for some $t > 0$, where $x_{a,d}$ and $x_{a,b}$ are defined as in (5.1.2). Therefore,

$$\arg(1 + u_{a,d}(\tilde{x})) = \frac{\varphi}{2} \text{ and}$$
$$\arg(1 + u_{a,b}(\tilde{x})) = -\frac{\varphi}{2}.$$

As before, we let $\tilde{x}$ tend towards a and compute the shifting function at $\tilde{x}$ as

$$\begin{aligned} s(\tilde{x}) &\equiv \frac{1}{2\pi}\left(\varphi + \arg(1 - u_{a,d}(\tilde{x})) - \arg(1 + u_{a,b}(\tilde{x})) + m\arg\left(\frac{C_{a,d}\tilde{y}_{a,d}(\tilde{x})}{C_{a,b}\tilde{y}_{a,b}(\tilde{x})}\right)\right) \\ &\equiv \frac{1}{2\pi}\left(\varphi + \arg(1 - u_{a,d}(a)) - \arg(1 - u_{a,b}(a)) + m\arg\left(\frac{C_{a,d}\tilde{y}_{a,d}(a)}{C_{a,b}\tilde{y}_{a,b}(a)}\right)\right) \\ &\equiv \frac{1}{2\pi}\left(\varphi + \arg(2) - \arg(2) + m\arg\left(\frac{C_{a,d}\tilde{y}_{a,d}(a)}{C_{a,b}\tilde{y}_{a,b}(a)}\right)\right) \equiv s_a \mod m. \end{aligned}$$

□

Remark 5.3.4. The intersection numbers given by Theorem 5.3.1 are independent of the choices of $\tilde{x}$ that were made in the proof. This approach works for any $\tilde{x} \in V$.

Even though the value of $s(\tilde{x})$ changes, if we choose $\tilde{x}$ in a different connected component of V, e.g. on the lower bisectrix in case (ii), the parametrization of the bisectrix and the corresponding arguments will change accordingly.

5.4 Computing the Abel-Jacobi map

Here we are concerned with explicitly computing the Abel-Jacobi map of degree zero divisors; for a general introduction see Section 2.9.

Assume for this section that we have already computed a big period period matrix (and all related data) following the Strategy from Section 5.2.

Let $D = \sum_{P \in X} v_P P \in \mathrm{Div}^0(X)$. After choosing a basepoint $P_0 \in X$, the computation of $\mathcal{A}$ reduces (using linearity) to

$$\mathcal{A}(D) \equiv \sum_{P \in X} v_P \int_{P_0}^{P} \bar{\omega} \mod \Lambda.$$

For every $P \in X$, $\int_{P_0}^{P} \bar{\omega}$ is a linear combination of vector integrals of the form

$$\int_{P_0}^{P_k} \bar{\omega} \quad \text{(see §5.4.1)}, \quad \int_{P_k}^{P} \bar{\omega} \quad \text{(see §5.4.2)} \quad \text{and} \quad \int_{P_0}^{P_\infty} \bar{\omega} \quad \text{(see §5.4.3)}, \quad \text{where}$$

- $\bar{\omega}$ is the vector of differentials in $\mathcal{W}$,
- $P = (x_P, y_P) \in X$ is a finite point,
- $P_k = (x_k, 0) \in X$ is a finite ramification point, i.e. $x_k \in \mathcal{B}$, and
- $P_\infty \in X$ is a point at infinity.

Typically, we choose as basepoint the ramification point $P_0 = (x_0, 0)$, where $x_0 \in \mathcal{B}$ is the root of the spanning tree $G = (\mathcal{B}, E)$.

Finally, the resulting vector integral has to be reduced modulo the period lattice Λ, which is done exactly as in the general case, and has already been covered in §4.9.6.

Remark 5.4.1 (Image of Abel-Jacobi map). For practical reasons, we will compute the image of the Abel-Jacobi map in the canonical torus $(\mathbb{R}/\mathbb{Z})^{2g}$. This representation has the following advantages:

- Operations on the Jacobian $\mathrm{Jac}(X)$ correspond to operations in $(\mathbb{R}/\mathbb{Z})^{2g}$.
- m-torsion divisors under $\mathcal{A}$ are mapped to vectors of rational numbers with denominator dividing m.

5.4.1 Between ramification points

Suppose we want to integrate $\bar{\omega}$ from $P_0 = (x_0, 0)$ to $P_k = (x_k, 0)$. By construction there exists a path $(x_0 = x_{k_0}, x_{k_1}, \ldots, x_{k_{n-1}}, x_{k_t} = x_k)$ in the spanning tree which connects x_0 and x_k. Thus, the integral splits into

$$\int_{P_0}^{P_k} \bar{\omega} = \sum_{j=0}^{t-1} \int_{P_{k_j}}^{P_{k_{j+1}}} \bar{\omega}.$$

Denote $a = x_{k_j}, b = x_{k_{j+1}} \in \mathcal{B}$. From §5.1.3 we know that for $(a, b) \in E$ a smooth path between $P_a = (a, 0)$ and $P_b = (b, 0)$ is given by

$$\gamma^{(0)}_{[a,b]} = \{\, (x, y_{a,b}(x)) \mid x \in [a, b] \,\}.$$

Let $\omega_{i,j} \in \mathcal{W}$ be a differential. According to the proof of Theorem 5.2.1 the corresponding integral is given by

$$\int_{\gamma^{(0)}_{[a,b]}} \omega_{i,j} = C_{a,b}^{-j} \left(\frac{b-a}{2}\right)^i \int_{-1}^{1} \frac{\varphi_{i,j}(u)}{(1-u^2)^{\frac{j}{m}}} \mathrm{d}u,$$

which is (up to the constants) an elementary integral (5.19) and has already been evaluated during the period matrix computation.

Remark 5.4.2. Moreover, the image of the Abel-Jacobi map between ramification points is m-torsion, i.e. for any two $k, j \in \{1, \dots, n\}$ we have

$$m \int_{P_j}^{P_k} \bar{\omega} \equiv \mathcal{A}(mP_k - mP_j) \equiv 0 \mod \Lambda, \tag{5.24}$$

since $\operatorname{div}\left(\frac{x-x_k}{x-x_j}\right) = mP_k - mP_j \in \operatorname{Prin}(X)$ is a principal divisor. In practice, this gives us a strong test for the correctness of the superelliptic Abel-Jacobi map.

5.4.2 Reaching non-ramification points

Let $P = (x_P, y_P) \in X$ be a finite point and $P_a = (a, 0)$ a ramification point such that $\mathcal{B} \cap]a, x_P] = \varnothing$. In order to define a smooth path between P and P_a we need to find a suitable analytic branch of $\mathcal{C}$.

This can be done following the approach in §5.1.2, the only difference being that x_P is not a branch point. Therefore, we are going to adjust the definitions and highlight the differences.

Let u_{a,x_P} be the affine linear transformation that maps $[a, x_P]$ to $[-1, 1]$. Similar to (5.7) we split up the image of $\mathcal{B}$ under u_{a,x_P} into subsets, but this time

$$u_{a,x_P}(\mathcal{B}) = \{-1\} \cup U^+ \cup U^-.$$

Then, $\tilde{y}_{a,x_P}(u)$ can be defined exactly as in (5.8) and is holomorphic in a neighbourhood ϵ_{a,x_P} of $[-1, 1]$. The term corresponding to a, that is

$$\sqrt[m]{1+u},$$

has a branch cut $]-\infty, -1]$ and is holomorphic on the complement of this cut.

Now we can define a branch of the curve, that is analytic in a neighbourhood V_{a,x_P} of $]a, x_P]$, by

$$y_{a,x_P}(x) = C_{a,x_P} \tilde{y}_{a,x_P}(u_{a,x_P}(x)) \sqrt[m]{1 + u_{a,x_P}(x)},$$

where

$$C_{a,x_P} = \left(\frac{x_P - a}{2}\right)^{\frac{n}{m}} e^{\frac{\pi i}{m}(\#U^+ \bmod 2)},$$

so that the statements of Proposition 5.1.4 continue to hold for $\tilde{y}_{a,x_P}$ and y_{a,x_P}, if we choose the sets ϵ_{a,x_P} and V_{a,x_P} as if x_P was a branch point.

Therefore, the lifts of $[a, x_P]$ to X are given by

$$\gamma^{(l)}_{[a,x_P]} = \{\, (x, \zeta^l y_{a,x_P}(x)) \mid x \in [a, x_P] \,\}, \quad l \in \mathbb{Z}/m\mathbb{Z}.$$

In order to reach $P = (x_P, y_P)$ we have to pick the correct lift. This is done by computing a *shifting number* $s \in \mathbb{Z}/m\mathbb{Z}$ at the endpoint x_P:

$$\zeta^s = \frac{y_P}{y_{a,x_P}(x_P)} = \frac{y_P}{C_{a,x_P} \tilde{y}_{a,x_P}(u_{a,x_P}(x_P)) \sqrt[m]{2}}$$

Consequently, $\gamma^{(s)}_{[a,x_P]}$ is a smooth path between P_a and P on X. We can now state the main theorem of this section.

Theorem 5.4.3. *Let $\omega_{i,j} \in \mathcal{W}^{mer}$ be a differential. With the choices and notation as above we have*

$$\int_{P_a}^{P} \omega_{i,j} = \zeta^{-sj} C_{a,x_P}^{-j} \left(\frac{x_P - a}{2}\right)^i \int_{-1}^{1} \frac{\varphi_{i,j}(u)}{(1+u)^{\frac{j}{m}}} \mathrm{d}u,$$

where

$$\varphi_{i,j} = \left(u + \frac{x_P + a}{x_P - a}\right)^{i-1} \tilde{y}_{a,x_P}(u)^{-j}$$

is holomorphic in a neighbourhood ϵ_{a,x_P} *of* $[-1,1]$ *and*

$$s = \frac{m}{2\pi} \arg\left(\frac{y_P}{C_{a,x_P}\tilde{y}_{a,x_P}(u_{a,x_P}(x_P))}\right).$$

Proof. We have

$$\begin{aligned}\int_{P_a}^{P} \omega_{i,j} &= \int_{\gamma_{[a,x_P]}^{(s)}} \frac{x^{i-1}}{y^j}\mathrm{d}x = \zeta^{-sj}\int_a^{x_P} \frac{x^{i-1}}{y_{a,x_P}(x)^j}\mathrm{d}x \\ &= \zeta^{-sj}C_{a,x_P}^{-j}\int_a^{x_P} \frac{x^{i-1}}{(1+u_{a,x_P}(x))^{\frac{j}{m}}\tilde{y}_{a,x_P}(u_{a,x_P}(x))^j}\mathrm{d}x\end{aligned}$$

Applying the transformation $u = u_{a,x_P}(x)$ introduces the derivative $\mathrm{d}x = \left(\frac{x_P-a}{2}\right)\mathrm{d}u$. Hence

$$\begin{aligned}\int_{P_a}^{P} \omega_{i,j} &= \zeta^{-sj}C_{a,x_P}^{-j}\left(\frac{x_P-a}{2}\right)\int_a^{x_P} \frac{x_{a,x_P}(u)^{i-1}}{(1+u)^{\frac{j}{m}}\tilde{y}_{a,x_P}(u)^j}\mathrm{d}u \\ &= \zeta^{-sj}C_{a,x_P}^{-j}\left(\frac{x_P-a}{2}\right)^i\int_a^{x_P} \frac{\left(u+\frac{x_P+a}{x_P-a}\right)^{i-1}}{(1+u)^{\frac{j}{m}}\tilde{y}_{a,x_P}(u)^j}\mathrm{d}u.\end{aligned}$$

The statement about holomorphicity of $\varphi_{i,j}$ is implied, since Proposition 5.1.4 holds for $\tilde{y}_{a,x_P}$ and y_{a,x_P} as discussed above. □

Remark 5.4.4. By Theorem 5.4.3, the problem of integrating $\bar{\omega}$ from P_0 to P reduces to numerical integration of

$$\int_{-1}^{1} \frac{\varphi_{i,j}(u)}{(1+u)^{\frac{j}{m}}}\mathrm{d}u. \tag{5.25}$$

Although these integrals are singular at only one end-point, they can still be computed using the double-exponential estimates presented in Section 3.4. This is explained in more detail in §5.5.3. Another option is to use non-symmetric Gauss-Jacobi integration with the estimates of §3.2.1, see also §5.5.2 for an explicit explanation.

5.4.3 Points at infinity

Recall from §5.1.1 that there are $\delta = \gcd(m,n)$ points at infinity $P_\infty^{(s)}$ on the projective curve C and therefore on the Riemann surface $X = C(\mathbb{C})$, so we introduce the set

$$\mathcal{P} = \{\, P_\infty^{(1)}, \ldots, P_\infty^{(\delta)} \,\} \subset X\,.$$

Suppose we want to integrate from P_0 to $P_\infty \in \mathcal{P}$, which is equivalent to computing the Abel-Jacobi map of the divisor $D_\infty = P_\infty - P_0$.

For the Abel-Jacobi map of points at infinity we employ the same strategy as in the general case (see §4.9.4), namely applying the moving lemma to the divisor D_∞.

In contrast to the general case this can be done much more explicitly for superelliptic curves, which is due to the special form of our affine model $\mathcal{C}$. We construct a principal divisor $D \in \mathrm{Prin}(X)$ such that $\mathrm{supp}(D) \cap \mathcal{P} = \{P_\infty\}$ and $v_{P_\infty}(D) = \pm 1$. Then, by definition of the Abel-Jacobi map,

$$\mathcal{A}(D_\infty \mp D) \equiv \mathcal{A}(D_\infty) \equiv \int_{P_0}^{P_\infty} \bar{\omega} \mod \Lambda$$

and $\operatorname{supp}(D_\infty \mp D) \cap \mathcal{P} = \varnothing$.

The exposition in this paragraph will explain the construction of D, while distinguishing three different cases.

In the following we denote by $\mu, \nu > 0$ the coefficients of the Bézout identity

$$-\mu m + \nu n = \delta.$$

Remark 5.4.5. Note that there are other ways of computing $\mathcal{A}(D_\infty)$. For instance, using transformations or direct numerical integration. Especially in the case $\delta = m$ a transformation (see Remark 5.1.3) is a good option and may be used in practice. The advantage of the approach that is presented here is that we can stay within our setup, i.e. we can compute solely on $\mathcal{C}$ and keep the integration scheme.

Coprime degrees

For $\delta = 1$ there is only one point at infinity $\mathcal{P} = \{P_\infty\}$ and we can easily compute $\mathcal{A}(D_\infty)$ by adding a suitable principal divisor D

$$\begin{aligned} \operatorname{div}(y^\nu) &= \nu \sum_{k=1}^{n} P_k - \nu n P_\infty, \\ \operatorname{div}((x - x_0)^{-\mu}) &= \mu m P_\infty - \mu m P_0, \\ D := \operatorname{div}(y^\nu (x - x_0)^{-\mu}) &= \nu \sum_{k=1}^{n} P_k - \mu m P_0 - P_\infty. \end{aligned}$$

We immediately obtain

$$\begin{aligned} \mathcal{A}(D_\infty) \equiv \mathcal{A}(D_\infty + D) &= \mathcal{A}(\nu \sum_{k=1}^{n} P_k - (\mu m + 1) P_0) \\ &\equiv \nu \sum_{k=1}^{n} \int_{P_0}^{P_k} \bar{\omega} \mod \Lambda \end{aligned}$$

and conclude that $\mathcal{A}(D_\infty)$ can be expressed in terms of integrals between ramification points (see §5.4.1).

Remark 5.4.6. In general, the principal divisor

$$D = \operatorname{div}(y^\nu (x - x_0)^{-\mu}) = \nu \sum_{k=1}^{n} P_k - \mu m P_0 - \sum_{l=1}^{\delta} P_\infty^{(l)}$$

yields the useful relation

$$\nu \sum_{k=1}^{n} \int_{P_0}^{P_k} \bar{\omega} \equiv \sum_{l=1}^{\delta} \int_{P_0}^{P_\infty^{(l)}} \bar{\omega} \mod \Lambda\,. \tag{5.26}$$

Once we have computed $\mathcal{A}(D_\infty)$ for all but one point at infinity, we may obtain the Abel-Jacobi map of the last one via equation (5.26).

Non-coprime degrees

For $\delta > 1$ the problem becomes a lot harder. First we need a way to distinguish the points in $\mathcal{P} = \{ P_\infty^{(1)}, \ldots, P_\infty^{(\delta)} \}$ and second they are singular points on the projective closure of our affine model $\mathcal{C}$ whenever $m \neq \{n, n \pm 1\}$.

As shown in [85, §1] we obtain a second affine patch of C that is non-singular along $\mathcal{P}$ in the following way:

Denoting $M = \frac{m}{\delta}$ and $N = \frac{n}{\delta}$, we consider the birational transformation

$$(x, y) = \Phi(r, t) = \left(\frac{1}{r^\nu t^M}, \frac{1}{r^\mu t^N}\right)$$

which results in an affine model

$$\tilde{\mathcal{C}} : r^\delta = \prod_{k=1}^{n}(1 - x_k r^\nu t^M).$$

The inverse transformation is given by

$$(r, t) = \Phi^{-1}(x, y) = \left(\frac{y^M}{x^N}, \frac{x^\mu}{y^\nu}\right).$$

Under this transformation the points at infinity on $\mathcal{C}$ are mapped to points on $\tilde{\mathcal{C}}$ with either $r = 0$ or $t = 0$. Since there are no points with $r = 0$ on $\tilde{\mathcal{C}}$, all points in $\mathcal{P}$ are mapped to points with $t = 0$, namely the finite non-singular points

$$(r, t) = (\zeta_\delta^s, 0), \quad s = 1, \ldots, \delta,$$

where $\zeta_\delta = e^{\frac{2\pi i}{\delta}}$. Hence, we can describe the points in $\mathcal{P} \subset X$ via

$$P_\infty^{(s)} = \Phi(\zeta_\delta^s, 0).$$

Note that the infinite points with $r = \infty$ on $\tilde{\mathcal{C}}$ are exactly the images of points with $x = 0$ on $\mathcal{C}$ (i.e. the fiber ${\varphi_x}^{-1}(0)$) under Φ^{-1}, while the infinite points with $t = \infty$ correspond to points with $y = 0$ (i.e. the ramification points P_k).

Suppose we want to compute the Abel-Jacobi map of $D_\infty^{(s)} = P_\infty^{(s)} - P_0$ for $s \in \{1, \ldots, \delta\}$. Again following our strategy, this time working on $\tilde{\mathcal{C}}$, we look at the intersection of the vertical line through $(\zeta_\delta^s, 0)$ with $\tilde{\mathcal{C}}$. We write down the corresponding principal divisor

$$E_1 = \operatorname{div}(r - \zeta_\delta^s) = \sum_{i=1}^{d}\left(\zeta_\delta^s, t_i^{(s)}\right) - NE_1'$$

where the $t_i^{(s)}$ are the zeros (up to multiplicity) of $h(t) = \prod_{k=1}^{n}(1 - x_k\zeta_\delta^{s\nu}t^M) - 1 \in \mathbb{C}[t]$, $d = \deg(h)$ and

$$E_1' = \begin{cases} (m - M)\Phi^{-1}(0, 0), & \text{if } 0 \in \mathcal{B}, \\ \sum_{Q \in \mathrm{pr}_x^{-1}(0)} \Phi^{-1}(Q) & \text{otherwise.} \end{cases} \tag{5.27}$$

Note that E_1 satisfies $\operatorname{supp}(E_1) \cap \Phi^{-1}(\mathcal{P}) = \{(\zeta_\delta^s, 0)\}$. Now, we can define the corresponding principal divisor on $\mathcal{C}$ by

$$D_1 := \operatorname{div}\left(\frac{y^M}{x^N} - \zeta_\delta^s\right);$$

then $v_{P_\infty^{(s)}}(D_1) \geq 1$ by construction.

Theorem 5.4.7. *Assume $v_{P_\infty^{(s)}}(D_1) = 1$ and $0 \notin \mathcal{B}$. Then, for $s = 1, \ldots, \delta$, there exist points $Q_1^{(s)}, \ldots, Q_{n-1}^{(s)} \in X \setminus \mathcal{P}$ such that*

$$\mathcal{A}(D_\infty^{(s)}) \equiv -\sum_{i=1}^{n-1}\int_{P_0}^{Q_i^{(s)}} \bar{\omega} \mod \Lambda. \tag{5.28}$$

Proof. First note that $v_{P_\infty^{(s)}}(D_1) = 1$ implies $M = 1$, i.e. $m = \delta$. Together with the assumption $0 \notin \mathcal{B}$, this gives us $\deg(h) = n$. Moreover, we can assume that $t_n^{(s)} = 0$ and $t_i^{(s)} \neq 0$ for $i = 1, \ldots, n-1$. Therefore,

$$\mathcal{A}(D_\infty^{(s)}) \equiv \mathcal{A}(D_\infty^{(s)} - D_1) \equiv -\mathcal{A}\Big(\sum_{i=1}^{n-1} \Phi(\zeta_\delta^s, t_i^{(s)}) - N \sum_{Q \in \varphi^{-1}(0)} Q\Big) \mod \Lambda.$$

Since $0 \notin \mathcal{B}$ the sum over the integrals from P_0 to all $Q \in \varphi^{-1}(0)$ vanishes modulo the period lattice Λ (in fact this is true for any non-branch point). Let x_k be the branch point that is closest to 0, then for every $\omega_{\tilde{i},j} \in \mathcal{W}$ we have

$$\begin{aligned}\sum_{Q \in \varphi^{-1}(0)} \int_{P_0}^{Q} \omega_{\tilde{i},j} &= \sum_{l=0}^{m-1} \int_{P_0}^{(0, \zeta^l \sqrt[m]{f(0)})} \omega_{\tilde{i},j} \\ &\equiv m \int_{P_0}^{P_k} \omega_{\tilde{i},j} + \Big(1 + \zeta^{-j} + \cdots + \zeta^{-j(m-1)}\Big) \int_{P_k}^{(0, \sqrt[m]{f(0)})} \omega_{\tilde{i},j} \\ &\equiv 0 \mod \Lambda\end{aligned}$$

by equation (5.24) and Theorem 5.4.3. If we take $Q_i^{(s)} = \Phi(\zeta_\delta^s, t_i^{(s)}) \in X \backslash \mathcal{P}$, $i = 1, \ldots, n-1$, we are done:

$$-\mathcal{A}\Big(\sum_{i=1}^{n-1} \Phi(\zeta_\delta^s, t_i^{(s)}) - N \sum_{Q \in \varphi^{-1}(0)} Q\Big) \equiv -\sum_{i=1}^{n-1} \int_{P_0}^{Q_i^{(s)}} \bar{\omega} \mod \Lambda.$$

□

In the case of Theorem 5.4.7 there exist additional relations between the vector integrals in (5.28) which we are going to establish now. Given $i \in \{1, \ldots, n-1\}$ and denoting $t^{(s)} = t_i^{(s)}$ we have that on $\tilde{\mathcal{C}}$

$$(\zeta_\delta^s, t^{(s)}) = (\zeta_\delta^s, \zeta_\delta^{-\nu s} t^{(\delta)}) \quad \text{for all } s = 1, \ldots, \delta.$$

Therefore, if we write $(x^{(s)}, y^{(s)}) := \Phi(\zeta_\delta^s, t^{(s)})$ and denote $Q^{(s)} = Q_i^{(s)}$, then

$$Q^{(s)} = (x^{(s)}, y^{(s)}) = (x^{(\delta)}, \zeta_\delta^{(\mu+\nu N)s} y^{(\delta)})).$$

The $Q^{(s)}$ having identical x-coordinates implies that there exists a $k \in \{1, \ldots, n\}$ such that

$$\int_{P_0}^{Q^{(s)}} \bar{\omega} \equiv \int_{P_0}^{P_k} \bar{\omega} + \int_{P_k}^{Q^{(s)}} \bar{\omega} \mod \Lambda,$$

while the relation between their y-coordinates yields

$$\int_{P_k}^{Q^{(s)}} \omega_{\tilde{i},j} = \zeta_\delta^{-(\mu+\nu N)sj} \int_{P_k}^{Q^{(\delta)}} \omega_{\tilde{i},j}$$

for all $\omega_{\tilde{i},j} \in \mathcal{W}$ and $s = 1, \ldots, \delta$. This proves the following corollary:

Corollary 5.4.8. *Under the assumptions of Theorem 5.4.7 and with the above notation we can obtain the image of $D_\infty^{(s)}$ under the Abel-Jacobi map for all $s = 1, \ldots, \delta$ from the $n-1$ vector integrals*

$$\int_{P_k}^{Q_i^{(\delta)}} \bar{\omega}, \quad i = 1, \ldots, n-1.$$

Unfortunately, this is just a special case. If $v_{P_\infty^{(s)}}(D_1)$ is greater than 1 (for instance, if $\delta \neq m$), the vertical line defined by $r - \zeta_\delta^s$ is tangent to the curve $\tilde{\mathcal{C}}$ at $(\zeta_\delta^s, 0)$ and cannot be used for our purpose.

Consequently, we must find another function. One possible choice here is the line defined by $r - t - \zeta_\delta^s$, which is now guaranteed to have a simple intersection with $\tilde{\mathcal{C}}$ at $(\zeta_\delta^s, 0)$ and does not intersect $\tilde{\mathcal{C}}$ in $(\zeta_\delta^{s'}, 0)$, $s \neq s'$.

The corresponding principal divisor is given by

$$E_2 = \operatorname{div}(r - t - \zeta_\delta^s) = \sum_{i=1}^{d} (t_i^{(s)} + \zeta_\delta^s, t_i^{(s)}) - \nu \sum_{k=1}^{n} \Phi^{-1}(x_k, 0) - N E_2',$$

where the $t_i^{(s)}$ are the zeros (up to multiplicity) of $h(t) = \prod_{k=1}^{n}(1 - x_k(t + \zeta_\delta^{(s)})^\nu t^M) - 1 \in \mathbb{C}[t]$, $d = \deg(h)$ and

$$E_2' = \begin{cases} (m - \frac{M+\nu}{N})\Phi^{-1}(0,0), & \text{if } 0 \in \mathcal{B}, \\ \sum_{Q \in \mathrm{pr}_x^{-1}(0)} \Phi^{-1}(Q), & \text{otherwise.} \end{cases} \tag{5.29}$$

Now,

$$D_2 := \operatorname{div}\left(\frac{y^M}{x^N} - \frac{x^\mu}{y^\nu} - \zeta_\delta^s\right)$$

is a principal divisor on $\mathcal{C}$ such that $v_{P_\infty^{(s)}}(D_2) = 1$.

Theorem 5.4.9. *Assume $v_{P_\infty^{(s)}}(D_1) > 1$ and $0 \notin \mathcal{B}$. Then, for $s = 1, \ldots, \delta$, there exist points $Q_1^{(s)}, \ldots, Q_{d-1}^{(s)} \in X \setminus \mathcal{P}$ such that*

$$\mathcal{A}(D_\infty^{(s)}) \equiv -\sum_{i=1}^{d-1} \int_{P_0}^{Q_i^{(s)}} \bar{\omega} + \nu \sum_{k=1}^{n} \int_{P_0}^{P_k} \bar{\omega} \mod \Lambda,$$

where $d = n(\nu + M)$.

Proof. First note that $0 \notin \mathcal{B}$ implies $d = \deg(h) = n(\nu + M)$. Moreover, our assumption implies $v_{P_\infty^{(s)}}(D_2) = 1$ so that we may assume $t_d^{(s)} = 0$ and $t_i^{(s)} \neq 0$ for $i = 1, \ldots, d-1$. Then,

$$\begin{aligned} \mathcal{A}(D_\infty^{(s)}) &\equiv \mathcal{A}(D_\infty^{(s)} - D_2) \\ &\equiv -\mathcal{A}\Big(\sum_{i=1}^{d-1} \Phi(t_i^{(s)} + \zeta_\delta^s, t_i^{(s)}) - \nu \sum_{k=1}^{n} (x_k, 0) - N \sum_{Q \in \varphi^{-1}(0)} Q\Big) \mod \Lambda. \end{aligned}$$

Choosing the points $Q_i^{(s)} = \Phi(t_i^{(s)} + \zeta_\delta^s, t_i^{(s)}) \in X \setminus \mathcal{P}$ and using the same reasoning as in the proof of Theorem 5.4.7 proves the statement. □

Remark 5.4.10. We can easily modify the statements of the Theorems 5.4.7 and 5.4.9 to hold for $0 \in \mathcal{B}$, i.e. when 0 is a branch point. Using equation (5.27), the statement of Theorem 5.4.7 becomes

$$\mathcal{A}(D_\infty^{(s)}) \equiv -\sum_{i=1}^{n-2} \int_{P_0}^{Q_i^{(s)}} \bar{\omega} + N(m - M) \int_{P_0}^{(0,0)} \bar{\omega} \mod \Lambda,$$

whereas, using equation (5.29), the statement of Theorem 5.4.9 becomes

$$\mathcal{A}(D_\infty^{(s)}) \equiv -\sum_{i=1}^{d-1} \int_{P_0}^{Q_i^{(s)}} \bar{\omega} + \nu \sum_{k=1}^{n} \int_{P_0}^{P_k} \bar{\omega} + (Nm - M - \nu) \int_{P_0}^{(0,0)} \bar{\omega} \mod \Lambda,$$

with $d = (n-1)(\nu + M)$.

5.5 Numerical integration

As explained in Section 5.2.2, the periods of the generating cycles $\gamma \in \Gamma$ are expressed in terms of elementary integrals (5.20)

$$I_{a,b}(i,j) = \int_{-1}^{1} \frac{u^{i-1}\mathrm{d}u}{(1-u^2)^{\frac{j}{m}}\tilde{y}_{a,b}(u)^j}\,, \tag{5.30}$$

where $(a,b) \in E$ and $\omega_{i,j} \in \mathcal{W}$. We restrict the numerical analysis to this case.

In this section, we denote by α the value $1-\frac{j}{m}$, which is the crucial parameter for numerical integration. Note that $\alpha = 1/2$ for hyperelliptic curves, while for general superelliptic curves α ranges from $\frac{1}{m}$ to $\frac{m-1}{m}$ depending on the differential form $\omega_{i,j}$ considered.

We study here three numerical integration schemes which are suitable for arbitrary precision computations:

- The double-exponential change of variable as introduced in Section 3.4 (see also [64] for more details) is completely general and its robustness allows to rigorously compute all the integrals (5.30) in a unified setting, even with different values of α (see Corollary 5.5.10).
- In the special case of hyperelliptic curves ($m = 2$), however, the Gauss-Chebyshev method (introduced in §3.2.3, see also [1, 25.4.38]) applies and provides a better scheme (fewer and simpler integration points).
- For $m > 2$, the integrals (5.30) can also be computed using Gauss-Jacobi integration with weight $(1-\alpha, 1-\alpha)$ (see §3.2.1 or [42]). However, a different scheme has to be computed for each α and it now involves computing roots of general Jacobi polynomials to large accuracy, which makes it hard to compete with the double-exponential scheme.

5.5.1 Gauss-Chebyshev quadrature

In the case of hyperelliptic curves ($m = 2$), we have $\alpha = \frac{1}{2}$ (and $j = 1$) and the integrals

$$\int_{-1}^{1} \frac{\varphi_{i,1}(u)}{\sqrt{1-u^2}}\mathrm{d}u,$$

can be efficiently handled by Gaussian quadrature with weight function $w(u) = 1/\sqrt{1-u^2}$, called Gauss-Chebyshev quadrature, as presented in §3.2.3. We recall the corresponding error bound on the ellipse

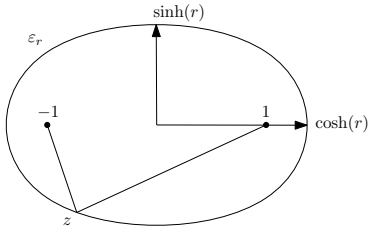


Figure 5.9: Parametrized ellipse.

Theorem 5.5.1. *Let Gaussian-Chebyshev quadrature on $N > 0$ points be applied to a function $g : [-1,1] \to \mathbb{C}$ that is holomorphic inside an ellipse ϵ_r (5.9) with foci ± 1 and*

$r > 0$. *Then, the error* (3.3) *satisfies*

$$E(N) \leq \frac{2M\pi}{e^{r2N} - 1} \tag{5.31}$$

where $M = \max\{\, |g(z)| \mid z \in \varepsilon_r \}$.

Now we can apply this theorem to the function

$$g_i(u) = \frac{u^{i-1}}{\tilde{y}_{a,b}(u)}\,, \tag{5.32}$$

where the roots of the polynomial $\tilde{y}_{a,b}(u)$ are exactly the u_k and the error can be controlled a priori.

Lemma 5.5.2. *Let* $r > 0$ *be such that* $2\cosh(r) < |u_k - 1| + |u_k + 1|$ *for all roots* u_k *of* $\tilde{y}_{a,b}(u)$*, then there exists an explicitly computable constant* $M(r)$ *such that*

$$|g_i(u)| \leq M(r) \quad \textit{for all } u \in \varepsilon_r\,.$$

Proof. We simply compute the distance $d_r(u_k) = \inf_{z\in\varepsilon_r} |z - u_k|$ from a root u_k to the ellipse ε_r, and let $M(r) = \frac{\cosh(r)^{i-1}}{\sqrt{\prod d_r(u_k)}}$. For simplicity, we can use the triangle inequality $d_r(u_k) \geq \cosh(r_k) - \cosh(r)$, where $2\cosh(r_k) = |u_k - 1| + |u_k + 1|$. □

More details on the choice of r and the computation of $M(r)$ are given in the paragraph below.

Corollary 5.5.3. *With* r *and* $M(r)$ *satisfying Lemma 5.5.2, for all* N *such that*

$$N \geq \frac{D + \log(2\pi M(r)) + 1}{2r}, \tag{5.33}$$

we have

$$E(N) = \left| I_{a,b}(i,1) - \frac{\pi}{N} \sum_{l=1}^{N} \frac{u_\ell^{i-1}}{\tilde{y}_{a,b}(u_\ell)} \right| \leq e^{-D},$$

where

$$u_\ell = \cos\left(\frac{2l-1}{2N}\pi\right)\,.$$

Remark 5.5.4. Generally we can expect Gauss-Chebyshev quadrature to be much better than (DE) integration for the evaluation of these hyperelliptic integrals. Nonetheless, it can happen that the double exponential scheme outperforms Gauss-Chebychev on certain integrals, in particular when r is close to zero and the domain ε_r becomes very small. This is easy to detect in practice and we can always switch to the better method.

Gauss-Chebyshev parameters

Here we show how to compute $r > 0$ such that the number of abscissas required for precision D (according to Corollary 5.5.3) is minimized. Yet another way to parametrize the ellipse ε_r is given by

$$\varepsilon_r = \{\, \cosh(r + it) = \cos(t - ir) \mid t \in]-\pi, \pi] \,\}.$$

The distance $d_k = \mathrm{dist}(u_k, \varepsilon_r)$ from a branch point $u_k \in U^+ \cup U^-$ to the ellipse ε_r can be computed applying Newton's method to the scalar product function

$$s(t) = \mathrm{Re}(\overline{z'(t)}(u_k - z(t))) \text{ where } z(t) = \cos(t - ir)$$

and we take $t = \mathrm{Re}(\arccos(u_k))$ as a starting point. By convexity of the ellipse, the solution is unique on the quadrant containing u_k. This has already been covered in the general case (see §4.7.2) with corresponding Algorithm A.1.1.

Optimal choice of r Let $|u_k - 1| + |u_k + 1| = 2\cosh(r_k)$. As already explained in §4.7.2, we want to choose $r < r_0 = \min_k r_k$ such that $u_k \notin \varepsilon_r$ and the number of integration points N from (5.33) is minimized. Without restriction assume here that the integrand is the function $g_1(u) = \frac{1}{\tilde{y}_{a,b}(u)}$ (5.32). We first estimate how the bound $M = M(r)$ varies for $r < r_0$.

- For all k such that $r_k > r_0$, we compute explicitly the distance $d_k = \operatorname{dist}(u_k, \varepsilon_{r_0}) < \operatorname{dist}(u_k, \varepsilon_r)$.
- For all k such that $r_k = r_0$, we use first order approximation $\operatorname{dist}(u_k, Z_{r-\eta}) = \eta D_k + O(\eta^2)$, where $D_k = \left|\frac{\partial u_k}{\partial r_k}\right| = |\sin(t_k - ir_k)|$.

Let K be the number of branch points $u_k \in U^+ \cup U^-$ such that $r_k = r_0$ and

$$M_0 = \sqrt{\prod_{r_k = r_0} D_k \prod_{r_k > r_0} d_k}^{-1},$$

then the integrand is bounded on $\varepsilon_{r_0 - \eta}$ by

$$M(r_0 - \eta) = M_0 \sqrt{\eta}^{-K} (1 + O(\eta)).$$

Plugging this into (5.33), the number of integration points satisfies

$$2N = \frac{D + \log(2\pi M_0) - K/2 \log(\eta)}{r_0 - \eta}(1 + O(\eta)).$$

The main term is minimized for η satisfying $\eta\left(2\frac{D+\log(2\pi M_0)}{K} + 1 - \log(\eta)\right) = r_0$. The solution can be written as a Lambert function or we use the approximation

$$r = r_0 - \eta = r_0\left(1 - \frac{1}{A + \log\frac{A}{r_0}}\right),$$

where $A = 1 + \frac{2}{K}(D + \log(2\pi M_0))$.

5.5.2 Gauss-Jacobi quadrature

For general $m > 2$ we can also compute the integrals (5.30) by Gaussian-Jacobi quadrature with weight function

$$w(u) = (1 - u^2)^{-\frac{j}{m}}$$

as introduced in §3.2.1. We can now use the following error bound given by Theorem 3.2.4:

Theorem 5.5.5. *Let Gaussian-Jacobi quadrature with weight $(\alpha - 1, \alpha - 1)$ on $N > 0$ points be applied to a function $g : [-1, 1] \to \mathbb{C}$ that is holomorphic inside an ellipse ϵ_r (5.9) with foci ± 1 and $r > 0$. Then, the error (3.3) satisfies*

$$E(N) \leq \frac{4MC}{(1 - e^{-2r})e^{r2N}}$$

where $M = \max\{\, |g(z)| \mid z \in \varepsilon_r \}$ and $C = 2^{2\alpha-1}\Gamma(\alpha)^2/\Gamma(2\alpha)$.

Similar to the previous section, we apply this theorem $\alpha - 1 = -j/m$ to the function

$$g_{i,j}(u) = \frac{u^{i-1}}{\tilde{y}_{a,b}(u)^j}\,. \tag{5.34}$$

We adjust Lemma 5.5.2 to the more general situation:

Lemma 5.5.6. *Let $r > 0$ be such that $2\cosh(r) < |u_k - 1| + |u_k + 1|$ for all roots u_k of $\tilde{y}_{a,b}(u)$, then there exists an explicitly computable constant $M(r)$ such that for all $u \in \varepsilon_r$*

$$\left| \frac{u^{i-1}}{\tilde{y}_{a,b}(u)^j} \right| \leq M(r).$$

Proof. Analogous to the proof of Lemma 5.5.2, but with

$$M(r) = \frac{\cosh(r)^{i-1}}{(\prod d_r(u_k))^{\frac{j}{m}}}.$$

□

Corollary 5.5.7. *With r and $M(r)$ satisfying Lemma 5.5.6 and $C = 2^{2\alpha-1}\Gamma(\alpha)^2/\Gamma(2\alpha)$, for all N such that*

$$N \geq \frac{D + \log(4M(r)C) - \log(1 - e^{-2r})}{2r},$$

we have

$$E(N) = \left| I_{a,b}(i,j) - \sum_{l=1}^{N} w_l \frac{u_\ell^{i-1}}{\tilde{y}_{a,b}(u_\ell)^j} \right| \leq e^{-D},$$

where the u_ℓ and w_l are the abscissas and weights of the Gauss-Jacobi quadrature (see §3.2.1) on N points with weight $\left(-\frac{j}{m}, -\frac{j}{m}\right)$.

More details on the choice of r and the computation of $M(r)$ are given in §5.5.2

Abel-Jacobi map

We can also use Gauss-Jacobi integration with weight function

$$w(u) = (1+u)^{-\frac{j}{m}}$$

to compute the integrals appearing in the computation of the Abel-Jacobi map §5.4.2. After the polynomial shift (5.21) the integrals (5.25) are of the form

$$I_{a,x_P}(i,j) = \int_{-1}^{1} \frac{u^{i-1}\mathrm{d}u}{(1+u)^{\frac{j}{m}} \tilde{y}_{a,x_P}(u)^j}.$$

Note that in this case the integration scheme is not symmetric and we have to use the error bound from Theorem 3.2.1:

Theorem 5.5.8. *Let Gauss-Jacobi quadrature with weight $(\alpha - 1, 0)$ on $N > 0$ points be applied to a function $g : [-1, 1] \to \mathbb{C}$ that is holomorphic inside an ellipse ϵ_r (5.9) with foci ± 1 and $r > 0$. Then, the error (3.3) satisfies*

$$E(N) \leq \frac{4MC}{(1 - e^{-r})e^{r2N}}$$

where $M = \max\{\, |g(z)| \mid z \in \varepsilon_r \}$ and $C = 2^\alpha\Gamma(\alpha)/\Gamma(\alpha+1)$.

We apply this theorem to the function

$$g_{i,j}(u) = \frac{u^{i-1}}{\tilde{y}_{a,x_P}(u)^j}, \tag{5.35}$$

where the values of r and $M = M(r)$ can be taken exactly as above.

Gauss-Jacobi parameters We adapt our method of choosing an optimal $0 < r < r_0$ from the Gauss-Chebyshev case for the superelliptc integrals in §5.5.2. Assume we want to bound the integrand $g_{1,j}(u)$ (5.34). Then with d_k, D_k and K as in §5.5.1 we take

$$M_0 = \left(\prod_{r_k = r_0} D_k \prod_{r_k > r_0} d_k \right)^{-\frac{j}{m}}$$

such that the integrand is bounded on $\varepsilon_{r_0-\eta}$ by

$$M(r_0 - \eta) = M_0 \eta^{-K\frac{j}{m}}(1 + O(\eta)).$$

If we use that $-\log(1 - e^{-2r}) < 1$ and plug this in (5.5.7), then

$$2N = \frac{D + \log(4CM_0) - Kj/m \log(\eta) + 1}{r_0 - \eta}(1 + O(\eta)).$$

Again, we approximate μ such that the main term is minimized. This strategy applies to the Abel-Jacobi map as well.

5.5.3 Double-exponential integration

We can apply the double-exponential integration scheme, as presented in Section 3.4, to the integrals (5.30). In particular, we want to apply Theorem 3.4.1 with $\alpha = 1 - \frac{j}{m}$ and $\lambda = \frac{\pi}{2}$ to the function

$$g_{i,j}(u) = \frac{u^{i-1}}{\tilde{y}_{a,b}(u)^j} \,. \tag{5.36}$$

If we choose the parameter $r \in]0, \frac{\pi}{2}[$ such that

$$r < r_0 := \min_k\{r_k\} \quad \text{where} \quad u_k = \tanh(\lambda \sinh(t_k + i r_k)) \tag{5.37}$$

we ensure that $u_k \notin Z_r$ for all roots u_k of $\tilde{y}_{a,b}(u)$. Then $g_{i,j}(u)$ is holomorphic on Z_r and we can calculate the quantities X_r and $B(r, \alpha)$ from (3.66). Since we can compute the distance of each branch point u_k to both $[-1, 1]$ and its neighborhood Z_r (see §5.5.3), we obtain

Lemma 5.5.9. *There exist explicitly computable constants $M_1^{i,j}$, $M_2^{i,j}$ such that*

- $|g_{i,j}(u)| \leq M_1^{i,j}$ *for all* $u \in [-1, 1]$,
- $|g_{i,j}(u)| \leq M_2^{i,j}$ *for all* $u \in Z_r$.

and we obtain a rigorous integration scheme by Theorem 3.4.1.

What makes the double-exponential integration even more valuable in this setting is that we can compute the integrals $I_{a,b}(i, j)$ for all $\omega_{i,j} \in \mathcal{W}$ using only <u>one</u> set of abscissas and weights. By Lemma 5.5.9 we can compute the constants M_1, M_2 for all $w_{i,j} \in \mathcal{W}$ and define constants $M_1(\mathcal{W}), M_2(\mathcal{W})$ as the maxima

$$\begin{aligned} M_1(\mathcal{W}) &= \max\{M_1^{i,j} \mid \omega_{i,j} \in \mathcal{W}\}, \\ M_2(\mathcal{W}) &= \max\{M_2^{i,j} \mid \omega_{i,j} \in \mathcal{W}\} \end{aligned}$$

such that we obtain

Corollary 5.5.10 (Double-exponential integration)**.** *For all $D > 0$, if we choose $\alpha = 1/m$,*

$$h = \frac{2\pi r}{D + \log(2M_2(\mathcal{W})B(r,\alpha) + e^{-D})} \quad \textit{and} \quad N = \left\lceil \frac{1}{h} \operatorname{asinh}\left(\frac{D + \log(\frac{2^{2\alpha+1}M_1(\mathcal{W})}{\alpha})}{2\alpha\lambda} \right) \right\rceil,$$

then we have that, for all $\omega_{i,j} \in \mathcal{W}$,

$$\left| I_{a,b}(i,j) - h \sum_{l=-N}^{N} w_l g_{i,j}(u_\ell) \right| \leq e^{-D},$$

where

$$u_\ell = \tanh(\lambda \sinh(lh)) \quad \textit{and} \quad w_l = \frac{\lambda \cosh(lh)}{\cosh(\lambda \sinh(lh))^{2\alpha}}.$$

Proof. With fixed $D, r, M_1 = M_1(\mathcal{W}), M_2 = M_2(\mathcal{W})$ we can consider the parameters from (3.67) as functions of $\alpha \in \left[\frac{1}{m}, \frac{m-1}{m}\right]$, say $h(\alpha)$ and $N(\alpha)$. Then, for all $j = 1, \ldots, m-1$, we have

$$h = h\left(\frac{1}{m}\right) \leq h\left(1 - \frac{j}{m}\right) \quad \text{and} \quad N \geq N\left(\frac{1}{m}\right) \geq N\left(1 - \frac{j}{m}\right).$$

The claim follows from Theorem 3.4.1. □

Abel-Jacobi map

We can also use Theorem 5.5.10 to compute the integrals appearing in §5.4.2. Via the polynomial shift (5.21) the integrals (5.25) become

$$I_{a,x_P}(i,j) = \int_{-1}^{1} \frac{u^{i-1}\mathrm{d}u}{(1+u)^{\frac{j}{m}} \tilde{y}_{a,x_P}(u)^j}.$$

Choosing $\alpha = 1 - \frac{j}{m}$ and $\lambda = \frac{\pi}{2}$, we now integrate the function

$$g_{i,j}(u) = \frac{(1-u)^{\frac{j}{m}} u^{i-1}}{\tilde{y}_{a,x_P}(u)^j}. \tag{5.38}$$

This is not a problem for the double-exponential integration as presented in §3.4 and we can continue to use the estimates of Theorem 5.5.10. Although our integrand $g_{i,j}(u)$ is not holomorphic at the end point -1, we can take exactly the same r as before since the domain Z_r is not affected by this. The constants M_1, M_2 can be chosen analogous to the previous case, see also §5.5.3.

Double-exponential parameters

We can easily bound the function $g_{i,j}(u)$ on the interval $[-1,1]$ by computing the distance to a branch point $u_k \in U^+ \cup U^-$. We have that

$$\operatorname{dist}(u_k, [-1,1]) = \begin{cases} |\operatorname{Im}(u_k)|, & \text{if } |\operatorname{Re}(u_k)| \leq 1, \\ ||\operatorname{Re}(u_k)| + i \cdot \operatorname{Im}(u_k) - 1|, & \text{otherwise.} \end{cases}$$

Therefore, for all $u \in [-1,1]$ we have that

$$|g_{i,j}(u)| = \left(\prod |u - u_k|\right)^{-j/m} \leq \left(\prod \operatorname{dist}(u_k, [-1,1])\right)^{-j/m} =: M_1^{i,j}.$$

The bound M_2 depends on the the parameter $r < r_0$. Once $r > 0$ is chosen (see paragraph below). we need to compute the distance of a branch point u_k to the zone Z_r. Note that the boundary of Z_r is parametrized by

$$\partial Z_r = \{\, z = \tanh(\lambda \sinh(t + ir)) \mid t \in \mathbb{R} \,\},$$

so that we can find the distance $\mathrm{dist}(u_k, Z_r)$ by numerically solving the equation

$$\arg(p_k - \tanh(\lambda \sinh(t + ir))) = \frac{\pi}{2},$$

where $p_k = |\mathrm{Re}(u_k)| + i\,|\mathrm{Im}(u_k)|$ for $t \in [0, \cosh^{-1}(\frac{\pi}{2}\lambda \sin(r))]$.

This can be done using Newton's method, but unfortunately the solution may not be unique. Instead, we can use ball arithmetic to compute a rigorous bound of the integrand on the boundary of Z_r. The process consists in recursively subdividing the interval until the images of the subintervals by the integrand form an ε-covering.

Optimal choice of r We adapt the method used for Gauss-Chebychev in §5.5.1. This time the number of integration points N is obtained from equation (3.67).

Writing $u_k = \tanh(\lambda \sinh(t_k + ir_k))$, we must choose $r < r_0 = \min_k\{r_k\}$ to ensure $u_k \notin Z_r$. Let

$$M_0 = (\prod_{r_k = r_0} D_k \prod_{r_k > r_0} d_k)^{-j/m},$$

where $d_k = \mathrm{dist}(u_k, Z_{r_0}) < \mathrm{dist}(u_k, Z_r)$ and

$$D_k = \left|\frac{\partial u_k}{\partial r_k}\right| = \left|\frac{\lambda \cosh(t_k + ir_k)}{\cosh(\lambda \sinh(t_k + ir_k))^2}\right|$$

is such that $\mathrm{dist}(u_k, Z_{r-\eta}) = \eta D_k + O(\eta^2)$, then the integrand $g_{1,j}(u)$ is bounded on $Z_{r_0-\eta}$ by

$$M_2 = M_0 \eta^{-\frac{jK}{m}}(1 + O(\eta)).$$

This yields

$$h = \frac{2\pi(r_0 - \eta)}{D + \log(2B(r_0, \alpha)M_0) - jK/m \log(\eta)} + O(\eta)$$

and the maximum is obtained for the solution η of $\eta(A - \log \eta) = r_0$ where $A = 1 + \frac{m}{jK}(D + \log(2B(r_0, \alpha)M_0))$.

5.6 Computational aspects

The main point of this section is the complexity analysis of our algorithms for superelliptic curves. We continue to use the label 'heuristic' to remind us of the heuristic assumptions made in §1.3. Besides that, we discuss precision issues and share some insights on how we actually implemented parts of the algorithm. Finally, we give some ideas on what could be done in the future.

5.6.1 Complexity analysis

We recall the parameters of the problem: we consider a superelliptic curve $C/\mathbb{C}$ given by an affine model $\mathcal{C} : y^m = p(x)$ where $m > 1$ and $p \in \mathbb{C}[x]$ is separable of degree $n \geq 3$, with associated compact Riemann surface $X = C(\mathbb{C})$. The genus g of C (resp. of X) satisfies

$$g \leq \frac{(m-1)(n-1)}{2} = O(mn).$$

For our analysis we continue to use the complexities given in Section 1.3. The computation of the Abel-Jacobi map of X has been decomposed into the tasks of computing

- the $(n-1)$ vectors of elementary integrals,
- the big period matrix $\Omega = (\Omega_A, \Omega_B)$ (5.16),
- (optional) the small period matrix $\tau = \Omega_A^{-1}\Omega_B$ (5.17),
- the Abel-Jacobi map at a point $P \in X$,

all of these to precision $D > 0$ digits.

Computation of elementary integrals

For each elementary cycle $\gamma_e \in \Gamma$, we numerically evaluate the vector of g elementary integrals from (5.20) as sums of the form

$$I_{a,b} \approx \sum_{\ell=1}^{N} w_\ell \frac{u_\ell^{i-1}}{y_\ell^j},$$

where $N = N_{\min}(D)$ is the number of integration points required for precision D, $\{u_\ell, w_\ell\}$ are integration points and weights, and $y_\ell = \tilde{y}_{a,b}(u_\ell)$. Our algorithm proceeds as follows:

- First we compute an integration scheme, as discussed in Chapter 3. With $N(D) = N_{\min}(D)$, the computational cost is given by
  - $O(N(D)\mathcal{T}(D))$ with $N(D) = O(D \log D)$ in the case of double-exponential integration (§3.4.1) and $N(D) = O(D)$ for Gauss-Chebyshev integration (§3.2.3).
  - $O(N(D)^2\mathcal{M}(D))$ with $N(D) = O(D)$ in the case of Gauss-Jacobi integration with $m > 2$ (§3.2.1).
- For every $\ell = 1, \ldots, N(D)$ we compute $y_\ell = \tilde{y}_{a,b}(u_\ell)$ using $n-2$ multiplications and one m-th root, as shown in §5.6.3 below.
- Starting from $\frac{w_\ell}{y_\ell}$, we evaluate all g terms $w_\ell \frac{u_\ell^{i-1}}{y_\ell^j}$ simultaneously, by either multiplying with u_ℓ or with $\frac{1}{y_\ell}$, and then adding the value to the corresponding entry of the vector integral.

Altogether, the computation of one vector of elementary integrals takes

$$\begin{aligned}\mathcal{E}(D) =& N(D)(n-2+\log D)\mathcal{M}(D) + N(D)g\mathcal{M}(D) \\ &+ \begin{cases} N(D)\mathcal{T}(D) & \text{operations, using (DE) or (GC),} \\ N(D)^2\mathcal{M}(D) & \text{operations, using (GJ),} \end{cases}\end{aligned} \tag{5.39}$$

so that depending on the integration scheme we obtain:

Theorem 5.6.1. *Each of the $(n-1)$ elementary vector integrals can be (heuristically) computed to precision $D > 0$ using*

$$O(N(D)\mathcal{M}(D)(g+\log D)) = \begin{cases} O(D^2 \log^{1+\varepsilon} D(g+\log D)) \text{ operations, if } m = 2, \\ O(D^2 \log^{2+\varepsilon} D(g+\log D)) \text{ operations, if } m > 2. \end{cases}$$

Proof. Plugging in $N(D), T(D), M(D)$ in equation (5.39) and using that $n = O(g)$ we obtain the first complexity using (GC) and the second complexity using (DE). □

Remark 5.6.2. In the case of Gauss-Jacobi integration we have

$$\mathcal{E}(D) = O(D^2 \log^{1+\varepsilon} D(g+D)),$$

which is worse than the complexity coming from the double-exponential integration. This is due to the slow initialization of the Gauss-Jacobi scheme, see §3.2.1.

Big and small period matrices

One of the nice aspects of the superelliptic case is that we do not need to compute the dense matrix $\Omega_\Gamma \in \mathbb{C}^{g\times 2g}$ from Section 5.2, but keep the decomposition of periods in terms of the elementary integrals

$$\int_{\gamma_e} \omega_{i,j} \in \mathbb{C}^{g\times(n-1)}\,.$$

Using the symplectic base change matrix S introduced in §5.2.4, the canonical homology basis is given by cycles of the form

$$\alpha_i = \sum_{e\in E,\, l\in\mathbb{Z}/m\mathbb{Z}} s_{e,l}\gamma_e^{(l)}\,, \tag{5.40}$$

where $\gamma_e^{(l)} \in \Gamma$ is a generating cycle and $s_{e,l} \in \mathbb{Z}$ is the corresponding entry of S. We use (5.18) to compute the coefficients of the big period matrix (Ω_A, Ω_B), so that each term of (5.40) involves only a fixed number of multiplications: each of the $O(g^2)$ periods is a linear combination of $O(g)$ elementary integrals, the coefficients involving precision D roots of unity.

For the complexity we run into the same problems that were already discussed in the general case in §4.8.1. Hence we can fall back to the statement of Theorem 4.8.5.

Abel-Jacobi map

This part of the complexity analysis is based on the results of Section 5.4 and assumes that we have already computed a big period matrix and all related data. We obtain the following complexity for computing the Abel-Jacobi map of superelliptic curves:

Theorem 5.6.3.

(i) *For each finite point $P \in X$ we can (heuristically) compute $\int_{P_0}^{P} \bar\omega$ to precision $D > 0$ using*

$$\mathcal{E}(D) = O(D^2 \log^{2+\varepsilon} D(g + \log D) + ngD) \quad \textit{operations.}$$

(ii) *For each infinite point $P_\infty \in X$ we can (heuristically) compute a representative of $\int_{P_0}^{P_\infty} \bar\omega \mod \Lambda$ to precision $D > 0$ using*

- *$O(ngD)$ operations, if $\delta = \gcd(m, n) = 1$,*
- *$n\mathcal{E}(D)$ operations in the case of Theorem 5.4.7,*
- *$n(n + \frac{m}{\delta})\mathcal{E}(D)$ operations in the case of Theorem 5.4.9.*

(iii) *Reducing a vector $v \in \mathbb{C}^g$ modulo Λ can be done using $O(g^\omega)$ multiplications.*

Proof.

(i) Follows from combining the results from §5.4.1 and Remark 5.4.4.

(ii) The statements follow immediately from §5.4.3, Theorem 5.4.7 and Theorem 5.4.9.

(iii) By §4.9.6, the reduction modulo the period lattice requires one $2g \times 2g$ matrix inversion and one multiplication.

□

5.6.2 Precision issues

As explained in §5, the ball arithmetic model allows to certify that the results returned by the ARB program [47] are correct. It does not guarantee that the result actually achieves the desired precision.

As a matter of fact, we cannot prove a priori that bad accuracy loss will not occur while summing numerical integration terms or during matrix inversion.

However, we take into account all predictable loss of precision:

- While computing the periods using equations (5.18) and (5.21), we compute a sum with coefficients
$$C_{a,b}^{-j}\left(\frac{b-a}{2}\right)^{i}\binom{i-1}{l}\left(\frac{b+a}{b-a}\right)^{i-1-l} \tag{5.41}$$
whose magnitude can be controlled a priori. It has size $O(g)$.

- The size of the coefficients of the symplectic reduction matrix are tiny (less than m in practice), but we can take their size into account before entering the numerical steps. Notice that generic HNF estimates lead to a very pessimistic estimate of size $O(g)$ coefficients.

- Matrix inversion of size g needs $O(g)$ extra bits.

This means we need to increase the internal precision from D to $\tilde{D} = D + O(c(p)g)$, where the constant $c(p)$ depends on the branch points as can be seen from equation (5.41). In particular, we see that $\tilde{D} - D$ is independent of D, as we heuristically assumed in §1.3.

Remark 5.6.4. In case the end result is imprecise by d bits, the user simply needs to run another instance to precision $D + d$ to reach the desired accuracy.

In fact, the mathematical quantities and the sequence of arithmetic operations performed in the algorithm remain the same. Now if the absolute error is reduced by d bits on input of an elementary operation this remains true on output; by induction this is true for the final result.

5.6.3 Implementation tricks

Here we simply give some ideas that we used in our implementation(s) to improve constant factors hidden in the big-O notation, i.e. the absolute run time.

In practice, 80 to 90% of the run time is spent on numerical integration of integrals (5.18) (see Table 5.3). According to §5.6.1, for each integration point $u_\ell \in]-1, 1[$ one first evaluates the y-value $y_\ell = \tilde{y}_{a,b}(u_\ell)$, then adds the contributions $w_\ell u_\ell^i / y_\ell^j$ to the integral of each of the g differential forms.

We shall improve on these two aspects, the former being prominent for hyperelliptic curves, and the latter when the $g \gg n$.

Computing products of complex roots

Following our definition (5.9), computing $\tilde{y}_{a,b}(u_\ell)$ involves $(n-2)$ m-th roots for each integration point.

Instead, we fall back to one single (usual) m-th root by computing $q(u) \in \frac{1}{2}\mathbb{Z}$ such that
$$\tilde{y}_{a,b}(u) = \zeta^{q(u)}\Big(\prod_{u_k \in U^-}(u-u_k)\prod_{u_k \in U^+}(u_k-u)\Big)^{\frac{1}{m}}. \tag{5.42}$$

This can be done by tracking the winding number of the product while staying away from the branch cut of the m-th root. For complex numbers $z_1, z_2 \in \mathbb{C}$ we can make a diagram of the quotient

$$\frac{\sqrt[m]{z_1}\,\sqrt[m]{z_2}}{\sqrt[m]{z_1 z_2}} \in \{1, \zeta, \zeta^{-1}\}$$

depending on the position of z_1, z_2 and their product $z_1 z_2$ in the complex plane, resulting in the following lemma:

Lemma 5.6.5. *Let $z_1, z_2 \in \mathbb{C}\backslash]\infty, 0]$. Then,*

$$\frac{\sqrt[m]{z_1}\,\sqrt[m]{z_2}}{\sqrt[m]{z_1 z_2}} = \begin{cases} \zeta, & \text{if} \quad \mathrm{Im}(z_1), \mathrm{Im}(z_2) > 0 \quad \text{and} \quad \mathrm{Im}(z_1 z_2) < 0, \\ \zeta^{-1}, & \text{if} \quad \mathrm{Im}(z_1), \mathrm{Im}(z_2) < 0 \quad \text{and} \quad \mathrm{Im}(z_1 z_2) > 0, \\ 1, & \text{otherwise.} \end{cases}$$

For $z \in]\infty, 0]$ we use $\sqrt[m]{z} = \zeta^{\frac{1}{2}} \cdot \sqrt[m]{-z}$.

Proof. Follows from the choices for $\sqrt[m]{\cdot}$ and ζ that were made in §5.1.2. □

Lemma 5.6.5 can easily be turned into an algorithm that computes $q(u)$.

Doing real multiplications

Another possible bottleneck comes from the multiplication by the numerator u_ℓ, which is usually done $g - m - 1$ times for each of the N integration points. More precisely, as we saw in the proof of Proposition 5.1.8, for each exponent j we use the exponents

$$0 \le i \le n_i = \left\lfloor \frac{nj - \delta}{m} \right\rfloor \quad \text{with} \sum n_i = g.$$

Without polynomial shift (5.21), this numerator would be $x_\ell = u_\ell + \frac{b+a}{b-a}$. While x_ℓ is a complex number, u_ℓ is real, so computing with u_ℓ saves a factor almost 2 on this aspect.

5.6.4 Further ideas

Improving branch points

As we saw in Section 5.5, the number of integration points closely depends on the configuration of branch points. In practice, when using double-exponential integration, the constant r is usually bigger than 0.5 for random points, but we can exhibit bad configurations with $r \approx 0.1$. In this case however, we can perform a change of coordinate by a Moebius transform $x \mapsto \frac{ax+b}{cx+d}$, as explained in Remark 5.1.3, to redistribute the points more evenly. Improving r from 0.1 to say 0.6 immediately saves a factor 6 on the run time.

Near-optimal tree

As explained in §5.1.3, we integrate along the edges of a maximal-flow spanning tree $G = (\mathcal{B}, E)$, where the capacity r_e of an edge $e = (a, b) \in E$ is computed as

$$r_e = \min_{c \in \mathcal{B}\backslash\{a,b\}} \left\{ \left| \mathrm{Im}(\sinh^{-1}(\tanh^{-1}(\tfrac{2c-b-a}{b-a})/\lambda) \right|, \text{ if } m > 2. \right.$$

Although this can be done in low precision, computing r_e for all $(n-1)(n-2)/2$ edges of the complete graph requires $O(n^3)$ evaluations of elementary costs (involving transcendental functions if $m > 2$).

For large values of n (comparable to the precision), the computation of these capacities has a noticable impact on the run time. This can be avoided by computing a *minimal spanning tree* that uses the euclidean distance between the end points of an edge as capacity, i.e. $r_e = |b - a|$, which reduces the complexity to $O(n^2)$ multiplications.

Given sufficiently many branch points that are randomly distributed in the complex plane, the shortest edges of the complete graph tend to agree with the edges that are well suited for integration. This approach has already been used in the general case, see Section 4.3.

Taking advantage of rational equation

In case the equation (5.2) is given by a polynomial $p(x)$ with small rational coefficients, one can still improve the computation of $\tilde{y}_{a,b}(u)$ in (5.42) by going back to the computation of $y(x_{a,b}(u)) = p(x)^{\frac{1}{m}}$. The advantage is that baby-step giant-step splitting can be used for the evaluation of $p(x)$, reducing the number of multiplications to $O(\sqrt{n})$. In order to recover $\tilde{y}_{a,b}(u)$, one needs to divide by $\sqrt[m]{1-u^2}$ and adjust a multiplicative constant including the winding number $q(u)$, which can be evaluated at low precision. This technique must not be used when u gets close to ± 1.

Splitting integrals

Similar to the situation explained in §4.7.5, numerical integration becomes quite inefficient when there are other branch points relatively close to an edge. Even the spanning tree optimization does not help if some branch points tend to cluster while others are far away. A simple example is given by the elliptic curve

$$y^2 = x(x - I)(x - 1000)\,.$$

The branch point i is very close to the integration path $[0, 1000]$ and imposes a value of $r(\varepsilon) = 0.045$ for Gauss-Chebyshev integration and a better but still small $r(Z) = 0.20$ for double-exponential integration.

In a case like this, one can always split the bad integrals to improve the relative distances to the singularities: in the case of double-exponential integration, writing

$$\int_0^{1000} = \int_0^{6} + \int_6^{1000}$$

gives two integrals with $r(Z) \geq 0.47 > 2 \cdot 0.20$ each and therefore reduces the total number of integration point. Alternatively, splitting in 2 and 33 results in three integrals each with $r(Z) \geq 0.63 > 3 \cdot 0.20$. In practice, splitting integrals almost never improves the situation for the double-exponential integration since the spanning tree avoids such edges by construction, when possible.

Gauss-Chebychev integration would greatly profit from splitting integrals. Adapting our splitting strategy used for Gauss-Legendre in 4.7.5, would result in four integrals, with splitting occuring at 3.8, 24.4 and 156.3, and respective values $0.75, 0.84, 0.83, 0.83$ of $r(\varepsilon)$ (compared to the 0.045 before). Sadly, splitting is not a good possibility here: the end points of the new integrals are not (both) singularities so that we would have to compute three different Gauss-Jacobi schemes in order to deal with the new integrals. The same is true when $m > 2$ for general Gauss-Jacobi integration. Since this would completely throw us out of our integration setting, it is much more convenient to switch to double-exponential integration for such integrals.

Shifting the integration path Another option with double exponential integration, as explained in [64, II.3.5], is to shift the integration path that is used for the change of variable.

5.7 Examples, timings and comparison

In this final section we want to compare all the different algorithms, implementatios and integration methods to find out which combination is optimal for period matrix computations of superelliptic curves. For these testing purposes we consider a family of curves given by Bernoulli polynomials

$$\mathcal{B}_{m,n} : y^m = B_n(x) = \sum_{k=0}^{n} \binom{n}{k} b_{n-k} x^k \tag{5.43}$$

as well as their reciprocals

$$\widetilde{\mathcal{B}}_{m,n} : y^m = x^n B_n\left(\frac{1}{x}\right). \tag{5.44}$$

The branch points of these curves present interesting patterns which can be respectively considered as good and bad cases from a numerical integration perspective (see Figure 5.10).

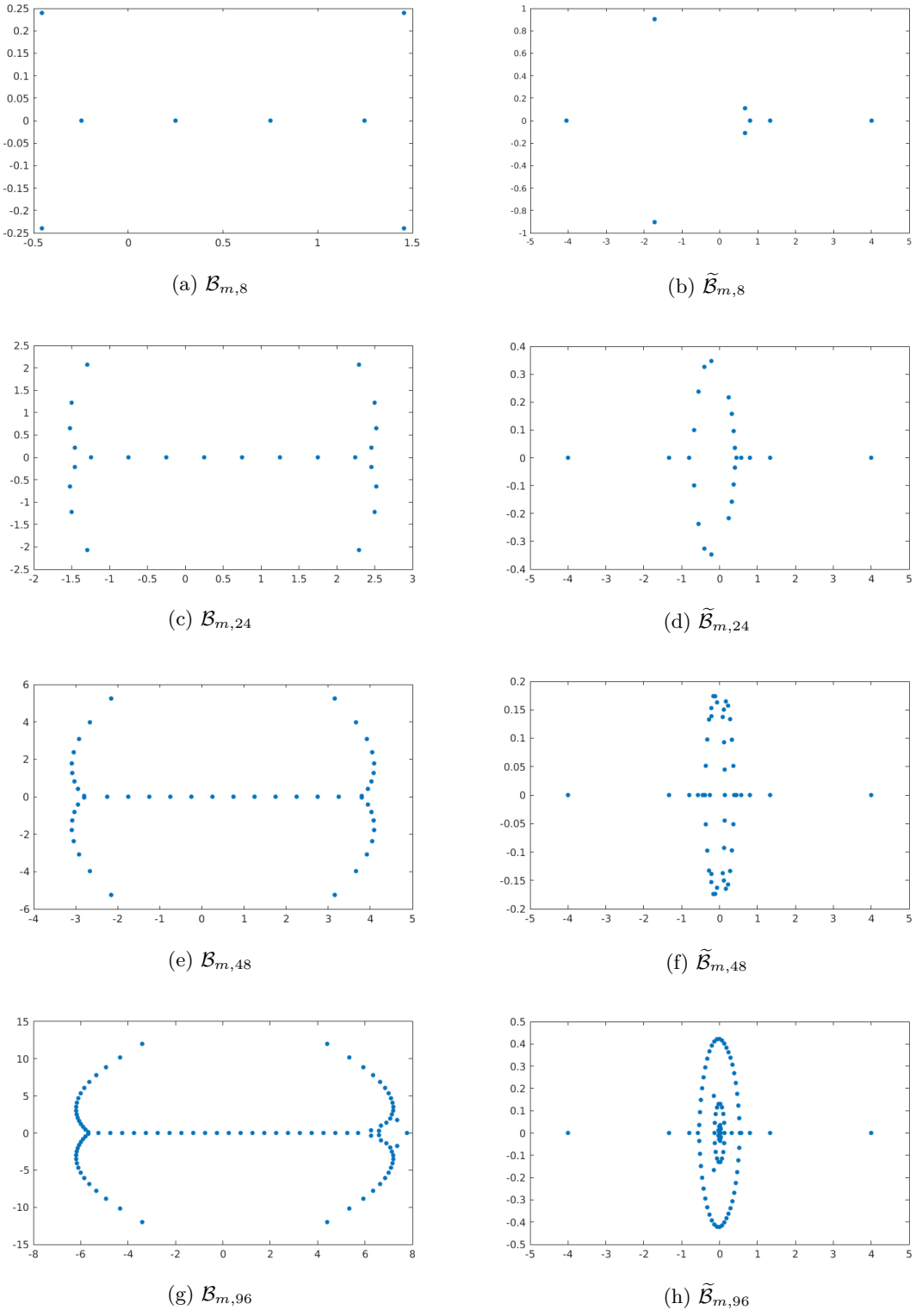
5.7.1 Big period matrix

Here we give timings for the computation of the big period matrix for hyper- superelliptic curves defined by the family of curves (5.43) and (5.44). In contrast to the comparison for general affine algebraic curves in Section 4.8, not only will we consider higher precisions here, namely $D_{10} \in \{50, 200, 500, 1000, 2000\}$, but also curves of higher genera, ranging from $g = 3$ up to $g = 805$.

Hyperelliptic examples In Table 5.1 we compare the following implementations and integration schemes for hyperelliptic curves:

- in ARB (A) using Gauss-Chebyshev integration (GC),
- in ARB (A) using double-exponential integration (DE),
- in MAGMA (M) using Gauss-Chebyshev integration (GC),
- in MAGMA (M) using using double-exponential integration (DE),
- the (old) existing MAGMA (M) implementation [93].

We will omit a comparison between these specialized algorithms and the implementations in MAPLE & SAGE for general affine algebraic curves, because they do not play in the same league. Having a quick look at the Tables in §4.8.4 and comparing the timings to Table 5.1 supports that decision.



(a) $\mathcal{B}_{m,8}$ (b) $\widetilde{\mathcal{B}}_{m,8}$

(c) $\mathcal{B}_{m,24}$ (d) $\widetilde{\mathcal{B}}_{m,24}$

(e) $\mathcal{B}_{m,48}$ (f) $\widetilde{\mathcal{B}}_{m,48}$

(g) $\mathcal{B}_{m,96}$ (h) $\widetilde{\mathcal{B}}_{m,96}$

Figure 5.10: Configurations of branch points of Bernoulli polynomials.

Genus	Curve	Algo. \ D_{10}	50	200	500	1000	2000
3	$\mathcal{B}_{2,8}$	(A)+(GC)	.005	.015	.067	0.30	1.47
		(A)+(DE)	0.06	0.39	0.69	3.76	22.3
		(M)+(GC)	0.02	0.08	0.23	0.82	3.86
		(M)+(DE)	0.11	0.55	2.50	11.0	61.2
		(M)+(old)	0.18	0.34	2.83	107	-
	$\widetilde{\mathcal{B}}_{2,8}$	(A)+(GC)	.005	.016	.075	0.33	1.75
		(A)+(DE)	0.05	0.43	0.69	5.11	41.0
		(M)+(GC)	0.02	0.11	0.41	1.56	7.73
		(M)+(DE)	0.14	0.75	3.91	17.8	102
		(M)+(old)	0.21	0.37	2.87	102	-
11	$\mathcal{B}_{2,24}$	(A)+(GC)	.037	0.14	0.72	3.67	18.6
		(A)+(DE)	0.85	5.17	6.50	35.7	205
		(M)+(GC)	0.16	0.52	2.02	7.12	31.7
		(M)+(DE)	0.78	3.69	16.8	66.4	320
		(M)+(old)	2.31	4.01	32.4	1233	-
	$\widetilde{\mathcal{B}}_{2,24}$	(A)+(GC)	.035	0.14	0.82	3.89	19.7
		(A)+(DE)	0.81	5.21	6.55	36.8	207
		(M)+(GC)	0.18	0.62	2.54	8.73	38.3
		(M)+(DE)	0.83	3.92	17.9	70.9	341
		(M)+(old)	2.86	4.64	35.7	1311	-
23	$\mathcal{B}_{2,48}$	(A)+(GC)	0.23	0.70	3.71	17.4	88.1
		(A)+(DE)	4.92	27.5	28.5	147	835
		(M)+(GC)	0.60	2.07	8.40	29.8	130
		(M)+(DE)	2.80	13.3	64.2	253	1213
	$\widetilde{\mathcal{B}}_{2,48}$	(A)+(GC)	0.24	0.79	3.84	17.6	90.2
		(A)+(DE)	5.40	31.4	29.5	150	835
		(M)+(GC)	0.73	2.54	10.2	35.3	161
		(M)+(DE)	2.90	13.8	67.8	268	1250
47	$\mathcal{B}_{2,96}$	(A)+(GC)	1.78	4.67	20.1	74.5	325
		(A)+(DE)	24.6	146	122	602	3077
		(M)+(GC)	2.61	9.21	36.4	125	504
		(M)+(DE)	11.4	55.4	269	1022	4458
	$\widetilde{\mathcal{B}}_{2,96}$	(A)+(GC)	1.67	4.54	19.7	72.5	347
		(A)+(DE)	45.0	35.0	161	646	3258
		(M)+(GC)	3.25	10.0	38.0	131	535
		(M)+(DE)	13.8	58.6	273	1019	4540

Table 5.1: Timings for hyperelliptic curves (in seconds).

Conclusions Our algorithm sets a new standard for big period matrices of hyperelliptic curves. Compared to the old MAGMA implementation, we obtain a huge speed-up which is due to integration along a spanning tree (smaller number of integrals) and using better integration schemes: in all displayed hyperelliptic examples of Table 5.1 Gauss-Chebyshev integration is by far the best option. As one would expect from the asymptotic behavior, the gap between (GC) and (DE) is growing with increasing precision. But even with double-exponential integration our algorithm still beats the old implementation in most instances. Moreover, the old implementation scales horribly with the precision and barely works for one thousand or more decimal digits, while our new algorithm shows consistent quadratic behavior as we would expect from our complexity analysis (except for very few instances using (A)+(DE)). Furthermore, we mention that our algorithm is basically unlimited in terms of the genus, while MAGMA cannot handle examples such as $\mathcal{B}_{2,48}$ or $\mathcal{B}_{2,96}$.

In defense of the MAGMA implementation we remark that it automatically computes the Abel-Jacobi map to the point at infinity (heuristically, using a fixed number of integration points and recursive subdivision of intervals) and that this is included in our timings. We cannot say how much of the time exactly is spent on that step, but it definitely skews the comparison.

Magma vs. Arb Comparing ARB to MAGMA we see that (A)+(GC) is consistently faster than (M)+(GC), although both implementations are very similar and closely follow the strategy that we developed in this chapter. On the one hand, this is due to Molin's programming skills, who did a great job implementing our algorithm. On the other hand, we believe that ARB is just faster than MAGMA in many aspects such as basic real and complex multi-precision arithmetic or finding polynomial roots. The same is true for the comparison of (A)+(DE) against (M)+(DE), although (A)+(DE) shows weird behavior in some instances which could be fixed easily.

Superelliptic examples In Table 5.2 we compare runnings times for superelliptic curves defined by polynomials (5.43) and (5.44). In particular, we compare our implementations

- in ARB (A) using double-exponential integration (DE),
- in MAGMA (M) using using double-exponential integration (DE),
- in MAGMA (M) using Gauss-Jacobi integration (GJ),
- and our MAGMA implementation (M) for general algebraic curves using (GL) integration with splitting (see Section 4.8).

For the same reason given in §5.7.1, we will not include the MAPLE & SAGE implementations in this comparison.

Genus	Curve	Algo. \ D_{10}	50	200	500	1000	2000
3	$\mathcal{B}_{3,4}$	(A)+(DE)	0.02	0.05	0.45	3.58	31.7
		(M)+(DE)	0.04	0.25	1.42	7.37	48.0
		(M)+(GJ)	0.03	0.63	5.30	37.1	291
		(M)+(GL)	0.35	2.85	19.3	117	943
7	$\widetilde{\mathcal{B}}_{4,6}$	(A)+(DE)	0.03	0.24	2.58	20.5	198
		(M)+(DE)	0.20	1.47	9.64	52.4	340
		(M)+(GJ)	0.93	17.6	174	1336	10956
		(M)+(GL)	1.03	6.25	34.3	167	1385
15	$\mathcal{B}_{7,7}$	(A)+(DE)	0.05	0.37	4.00	31.0	361
		(M)+(DE)	0.23	1.80	15.3	81.5	485
		(M)+(GJ)	0.20	3.00	30.2	196	1468
		(M)+(GL)	4.15	23.3	122	525	2648
31	$\widetilde{\mathcal{B}}_{10,8}$	(A)+(DE)	0.07	0.54	5.94	44.4	442
		(M)+(DE)	0.43	2.67	15.4	76.8	467
		(M)+(GJ)	0.97	15.0	138	969	8231
		(M)+(GL)	11.0	55.9	287	1152	5219
37	$\mathcal{B}_{8,12}$	(A)+(DE)	0.28	1.60	6.05	39.5	305
		(M)+(DE)	0.64	3.75	21.1	102	608
		(M)+(GJ)	0.55	5.49	44.8	287	2135
		(M)+(GL)	6.64	33.2	164	663	3608
77	$\mathcal{B}_{24,8}$	(A)+(DE)	0.29	1.01	5.65	33.4	225
		(M)+(DE)	0.77	3.50	16.9	75.7	423
		(M)+(GJ)	0.86	7.03	52.7	334	2417
		(M)+(GL)	54.5	262	1337	5093	-
91	$\widetilde{\mathcal{B}}_{18,12}$	(A)+(DE)	0.52	2.41	10.4	61.7	435
		(M)+(DE)	1.61	7.32	36.5	164	926
		(M)+(GJ)	2.29	24.9	215	1415	10920
		(M)+(GL)	80.6	367	1805	6734	-
184	$\mathcal{B}_{17,24}$	(A)+(DE)	4.40	13.4	46.3	218	1238
		(M)+(DE)	6.47	22.1	92.8	381	1966
		(M)+(GJ)	5.28	22.2	134	710	4710
		(M)+(GL)	139	515	2366	8703	-
529	$\widetilde{\mathcal{B}}_{24,48}$	(A)+(DE)	85.5	196	468	1564	7590
		(M)+(DE)	79.8	169	531	1827	8208
		(M)+(GJ)	79.6	222	1057	5278	34233
		(M)+(GL)	2104	5674	19967	-	-
805	$\mathcal{B}_{18,96}$	(A)+(DE)	475	892	2280	6052	23234
		(M)+(DE)	300	620	1779	5562	22282
		(M)+(GJ)	250	462	1263	4155	18790

Table 5.2: Timings for superelliptic curves (in seconds).

Conclusions From Table 5.2 we see that for precision $D_{10} = 50$, Gauss-Jacobi integration is a competitive alternative to double-exponential integration. Already for precision $D_{10} = 200$ though, (DE) wins over (GJ) due to faster initialization, as already discussed in §3.4.1 and §3.2.1. There might still be some optimization potential left for computing the Gauss-Jacobi scheme, but it will hardly become faster than (DE) for higher precisions. An exception here being the genus 805 curve defined by $\mathcal{B}_{18,96}$. Due to the huge degree $n = 96$, the cost of evaluating the integrals for each abscissa increase drastically. Hence, having less abscissas overall becomes more valuable than fast initialization. From Chapter 3, we know that Gaussian quadratures require much less integration points than double-exponential integration. The same argument explains why (GL) integration is so good for general curves: because the cost per abscissa is tremendous. However, in the superelliptic case (i.e. $m > 2$) (DE) is the superior integration method (at least for all examples that are not completely ridiculous).

Our implementation for general curves (M)+(GL) faces its obvious limitations here. In particular, it is unfit to handle large genus examples such as $\widetilde{\mathcal{B}}_{24,48}$ and $\mathcal{B}_{18,96}$ or precisions like $D_{10} = 1000, 2000$. Including these timings in our comparison here highlights just how fast and efficient our new algorithm for superelliptic curves really is.

Magma vs. Arb We recognize the same differences as in §5.7.1: ARB is generally faster than MAGMA (except for the high genus examples $\widetilde{\mathcal{B}}_{24,48}$ and $\widetilde{\mathcal{B}}_{18,96}$), although we can observe that the gap is more narrow than for hyperelliptic curves.

Distribution of computational cost Finally, we have a look at the relative computational costs in our MAGMA implementation, see Table 5.3 below for some examples. As already mentioned in §5.6.3, we see that by far the most time is spent on evaluating the integrals (usually >80%), which includes computing the m-th root as an analytic function. These steps consist entirely of basic arithmetic operations and simple transcendental functions. Initialization takes almost no time (using (GC) for $m = 2$ and (DE) for $m > 2$) and computing the roots of $f(x)$ is also pretty cheap (5% or less). Surprisingly, computing the big period matrix (§5.6.1) can take a significant portion of the time, once the genus becomes quite large. Other than that, Table 5.3 confirms that our period matrix algorithm is fairly optimized.

Genus	Curve	D_{10}	Init	Integration	Anal. cont.	Roots	Matrix mult.
11	$\mathcal{B}_{2,24}$	100	< 1	80	62	5	< 2
		1000	3	93	79	2	< 1
184	$\mathcal{B}_{17,24}$	100	<1	85	25	<1	11
		1000	< 2	97	43	<1	1
47	$\mathcal{B}_{2,48}$	100	<1	84	74	5	<1
		1000	<1	96	83	2	< 1
529	$\mathcal{B}_{24,48}$	100	<1	63	13	<1	33
		1000	<1	94	27	<1	5

Table 5.3: Contributions of individual tasks to running time (in percent).

5.7.2 Abel-Jacobi map

Computing the Abel-Jacobi map via numerical integration can be a useful tool for the identification of principal divisors on curves, which the following example will show. This example is due to Jeroen Sijsling.

Example 5.7.1. We consider a projection φ between hyperelliptic curves X (genus 3) and Y (genus 2). In particular, the curves are given by affine equations

$$\begin{aligned} X &: y^2 = x^8 + 2x^6 - 3x^4 + 7x^2 - 1, \\ Y &: y^2 = x(x^4 + 2x^3 - 3x^2 + 7x - 1)\,. \end{aligned}$$

and the projection is given as

$$\varphi : X \to Y,\ (x, y) \mapsto (x^2, xy)\,.$$

Under the pullback φ^*, the zero divisor $D = (0, 0) - P_\infty$ on Y becomes principal on X. In fact, D generates the kernel of the corresponding map on the Jacobians. This can be verified by appliying our algorithms. We compute the image of D in $\mathbb{R}^4/\mathbb{Z}^4$ under the Abel-Jacobi map on Y

$$\mathcal{A}(D) \equiv \begin{pmatrix} 1/2 & 0 & -1/2 & -1/2 \end{pmatrix} =: v\,.$$

Using algorithms of Costa et al. [21] we can use period matrices of X and Y to compute a homology representation of the corresponding map between the Jacobians

$$R = \begin{pmatrix} 0 & 0 & 1 & 2 & -1 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & -1 & 0 & 1 & -2 \end{pmatrix}$$

and get that

$$v \cdot R = \begin{pmatrix} 1 & 0 & 1 & 1 & -1 & 1 \end{pmatrix} \in \mathbb{Z}^6\,,$$

which confirms that the pullback of D is a principal divisor on X. For this computation we used 200 decimal digits precision.

Comparison of integration methods Finally, we want to give timings for the computation of the Abel-Jacobi map. As explained in §5.4.3 we can compute the Abel-Jacobi map of $D_\infty = P_\infty - P_0$ elegantly by moving the divisor such that its support contains only finite points. This is an excellent opportunity to test the correctness and the performance of our code.

For each curve in 5.1 we compute the Abel-Jacobi map of $D_\infty^{(s)}$ for all points at infinity $s = 1, \ldots, \delta = \gcd(m, n)$. In the nice case of Theorem 5.4.7, we have to compute at most $n - 1$ integrals. Otherwise, we are stuck in the case of Theorem 5.4.9 and have to compute up to $n(m + \delta\nu)$ integrals.

After having computed $\mathcal{A}(D_\infty^{(s)})$ for $s = 1, \ldots, \delta$ we can check our results by numerically verifying the relation

$$\frac{m}{\delta} \sum_{s=1}^{\delta} \mathcal{A}(D_\infty^{(s)}) \equiv 0 \mod \mathbb{Z}^{2g}\,.$$

We executed all computations twice, once using double-exponential (DE) integration and once using Gauss-Jacobi (GJ) integration, as explained in the previous sections, in order to compare the two integration schemes in this situation. In Table 5.4 you can see examples of different genera chosen such that $\delta \neq 1$ (otherwise the computation would be

trivial, see §5.4.3). The 3rd column states the number of actual integrations that were performed and therefore indicates whether we are in the case of either Theorem 5.4.7 or 5.4.9.

Genus	Curve	#Integrals	Algo. \ D_{10}	50	200	500	1000	2000
3	$\mathcal{B}_{2,8}$	7	(M)+(DE)	0.10	0.57	3.11	15.2	91.5
			(M)+(GC)	0.05	0.55	5.26	36.0	283
6	$\mathcal{B}_{5,5}$	3	(M)+(DE)	0.07	0.50	3.30	18.3	118
			(M)+(GC)	0.04	0.57	4.86	34.3	275
7	$\mathcal{B}_{4,6}$	$2 \cdot 17$	(M)+(DE)	0.90	6.89	44.6	245	1594
			(M)+(GC)	1.87	29.6	308	2215	17362
15	$\mathcal{B}_{7,7}$	5	(M)+(DE)	0.15	1.06	6.43	33.8	213
			(M)+(GC)	0.20	2.77	26.4	177	1381
17	$\mathcal{B}_{6,8}$	$2 \cdot 31$	(M)+(DE)	2.05	13.7	83.9	436	2702
			(M)+(GC)	2.30	22.5	194	1178	9295
37	$\mathcal{B}_{8,12}$	$4 \cdot 35$	(M)+(DE)	7.04	41.9	238	1164	6862
			(M)+(GC)	6.74	55.2	424	2544	18137

Table 5.4: Timings for the Abel-Jacobi map (in seconds).

Conclusion The situation is fairly similar to our comparison (§5.7.1) for period matrices of superelliptic curves. For precision $D_{10} = 50$, Gauss-Jacobi integration can be seen as solid alternative, while for higher precisions (DE) is the preferable choice. Note that the more often we can reuse each integration scheme (i.e. more integrals), the more it becomes profitable to use (GJ) integration.

5.8 Outlook

In this chapter we presented an approach based on numerical integration for arbitrary precision computation of period matrices and the Abel-Jacobi map of superelliptic curves given by $m > 1$ and squarefree $p \in \mathbb{C}[x]$.

Integration along a spanning tree and the special geometry of such curves make it possible to compute these objects to high precision performing only a few numerical integrations. The resulting algorithm scales execellently with the genus and works for several thousand digits of precision.

We briefly comment on possible generalizations and improvements here.

Multiple roots In a first step the algorithm could be extended to all complex superelliptic curves given by $m > 1$ and $p \in \mathbb{C}[x]$, where f can have multiple roots of order at most $m - 1$, say $p(x) = \prod_{k=1}^{n}(x - x_k)^{n_k}$. We want to highlight the following issues:

- The differentials are of the form
$$\frac{\prod_{k=1}^{n}(x - x_k)^{i_k}}{y^j}\mathrm{d}x$$
where the exact condition on the holomorphicity is given in [52, Theorem 3]. However, these holomorphic differentials can still be integrated using double-exponential integration as presented in §3.4.

- The local monodromies may no longer be equal or even cyclic, but they are completely (up to conjugacy) determined by the multiplicites n_k. We believe that applying the Tretkoff algorithm [88], that was already used for the homology in the general case (see Section 4.6), could be a better approach than generalizing the methods used in Section 5.3 (although this seems possible).

Although several adjustments would have to be made in the analysis and in the code, staying within the superelliptic setting promises a fast and rigorous extension of our algorithm. Moreover, this generalization would allow to perform any Moebius transform on the model of the curve and to efficiently implement the idea of §5.6.4.

Compact Riemann surfaces We also believe that the strategy employed here (numerical integration between branch points combined with information about local intersections) could be adapted for compact Riemann surfaces given by an affine equation $f(x, y) = 0$ with irreducible $f \in \mathbb{C}[x, y]$ (which was the topic of Chapter 4).

The obvious interpolation between our approach for compact Riemann surfaces and the one for superelliptic curves, would be to use numerical integration between branch points and symbolic integration (for example using Puiseux series expansions) in the neighborhood of exceptional values. This possibility has already been discussed in Section 4.11 and completely hinges on whether we have at our disposal an efficient algorithm for Puiseux series expansions with arbitrarily many terms. After discussing this possibility with Poteaux and Johansson, we highly doubt that this approach will be more efficient than numerical integration along arcs and circles.

Recognizing superelliptic curves Suppose we want to determine period matrices or the Abel-Jacobi map associated to an irreducible polynomial $f \in \mathbb{C}[x, y]$. Since our algorithms for superelliptic curves (in the sense of Definition 5.1.1) are rigorous and much faster than our algorithms for general compact Riemann surfaces, we would like to have an algorithm that checks whether there exists an affine model of the form $y^m = p(x)$ and if so, determines one. For hyperelliptic curves (i.e. $m = 2$) this is already implemented in MAGMA and we obtain a model by calling the function 'IsHyperelliptic' followed by 'SimplifiedModel'. In order to recognize superelliptic curves, one could compute the automorphism group and then check whether taking the quotient by it gives a projective line. In order to obtain a superelliptic affine model one still needs to find a suitable projective transformation; this could be done using algorithmic Kummer theory (see [19, §5.2.3] for the number field case). The existence of such a model is guaranteed by [69, Theorem IV.3.2].

Chapter 6

Applications

In the final chapter of this thesis we explore several applications of period matrices and the Abel-Jacobi map (or, more generally, integration of differential forms on Riemann surfaces) to interesting problems in number theory. We show how our work is related to or might be connected with results and algorithms of other authors in many instances. While we carried out some applications by ourselves, we simply indicate out how the others might be realized.

Structure of this chapter More precisely, we briefly introduce the Birch and Swinnerton-Dyer conjecture for algebraic curves defined over the rational numbers in Section 6.1 and explain how big period matrices of the associated Riemann surface are related to the real period of the Jacobian. In the following Section 6.2 we show how big period matrices can be used as numerical approximation to compute endomorphism rings of Jacobians and give an example how one can use MAGMA to verify that a Jacobian has complex multiplication by a certain CM field. As already mentioned several times, one of the most important applications is the Riemann Theta functions, which is actually defined in terms of an element of the Siegel upper half-space, e.g. by a small period matrix. Its definition and possible applications are discussed in Section 6.3. The final application deals with computing certain regulators that are related to the second K-group of algebraic curves and can be found in Section 6.4. We describe how the algorithms presented in Chapter 4 can be modified to numerically integrate a certain meromorphic differential. Picking up the strategy of de Jeu, Dokchitser & Zagier [29] we compute numerical approximations to such regulators (up to rational multiples) for many hyperelliptic and non-hyperelliptic curves, confirming old and providing new evidence for the Beilinson conjecture.

6.1 The Birch and Swinnerton-Dyer conjecture

Let $J = \mathrm{Jac}(C/\mathbb{Q})$ be the Jacobian of a smooth, projective, geometrically irreducible curve C defined over $\mathbb{Q}$. For such abelian varieties the Birch & Swinnerton-Dyer (BSD) conjecture can be stated as

Conjecture 6.1.1 (Birch and Swinnerton-Dyer)**.** Denote the analytic rank of J by $r = \mathrm{ord}_{s=1} L(J,s)$. Then, the following equalities are true:

$$\begin{aligned} &(1) \quad r = \mathrm{rk}(J/\mathbb{Q})\,, \\ &(2) \quad \lim_{s\to 1} \frac{L(J,s)}{(s-1)^r} = \frac{\Omega_J \cdot R_J \cdot \prod_p c_p \cdot \#Ш(J)}{(\#J(\mathbb{Q})_{\mathrm{tors}})^2}\,. \end{aligned} \tag{6.1}$$

The BSD conjecture for abelian varieties is open in general. It is proven only for few examples and certain modular abelian varieties. Thus, it is of great interest to provide

numerical evidence for the conjecture. For a precise definition of all the six quantities appearing in the second statements we refer to [34] or [46, p. 462]. A definition of the L-function of algebraic curves can be found in §6.4.1.

In this section we want to explain how the computation of period matrices of Riemann surfaces, which is the main topic of this thesis (see Chapters 4 and 5) is related to the BSD conjecture. Of interest to us is the quantity Ω_J, which appears in the numerator of the right hand side of the second equation (6.1), and is called the *real period of* J.

Let $\Omega \in \mathbb{C}^{g\times 2g}$ be a big period matrix obtained by integrating a $\mathbb{Q}$-basis of holomorphic differentials $\bar{\omega} = (\omega_1, \ldots, w_g)$ of C along a basis $(\gamma_1, \ldots, \gamma_{2g})$ of the integral homology group $H_1(C, \mathbb{Z})$. Now, the columns of the matrix $\mathrm{Tr}_{\mathbb{C}/\mathbb{R}}(\Omega) = \Omega + \bar{\Omega} \in \mathbb{R}^{g\times 2g}$ generate a lattice Λ in $\mathbb{R}^g$. The value of the integral $\int_{J(\mathbb{R})} |w_1 \wedge \cdots \wedge w_g|$ can then be computed as the volume of a fundamental domain for Λ.

In the case where $\bar{\omega}$ is also a basis for the integral 1-forms on the Jacobian J we already found the real period of J, i.e. $\int_{J(\mathbb{R})} |w_1 \wedge \ldots \wedge w_g| = \Omega_J$. Otherwise, the real period can be obtained via a change of basis, i.e. by expressing the integral 1-forms on J in terms of the basis $\bar{\omega}$.

This procedure is described explicitly in [34, Section 3.5] in the case of genus 2 curves and has been proved rigorously and generalized by Raymond van Bommel [89] to arbitrary hyperelliptic curves of higher genus. Of course, our period matrix algorithms are not limited to curves of such small genera and could be used to provide high precision big period matrices for the computation of the real period in the future.

6.2 Endomorphism rings

Let C be a smooth, projective, geometrically irreducible curve of genus g defined over a number field K with Jacobian J and let $\Omega \in \mathbb{C}^{g\times 2g}$ be a big period matrix such that $J(\mathbb{C}) \cong \mathbb{C}^g/\Lambda$ where $\Lambda = \Omega\mathbb{Z}^{2g}$. Using methods of van Wamelen which are described in [93] and implemented in MAGMA we can compute a (putative) basis for the endomorphism ring $\mathrm{End}(J(\mathbb{C}))$, that is, matrices $R_1, \ldots, R_d \in \mathbb{Z}^{2g\times 2g}$ that form a $\mathbb{Z}$-basis of $\mathrm{End}(J(\overline{K}))$. Given a big period matrix Ω (with sufficiently high precision) as input, the function EndmorphismRing(Ω) uses lattice methods to compute such a basis, as well as matrices $M_i \in \mathbb{C}^{g\times g}$ such that we have the equality $M_i\Omega = \Omega R_i$, $i = 1, \ldots, d$.

Originally, van Wamelen implemented these algorithms for Jacobians of hyperelliptic curves, but many of them solely rely on period matrices as input and can thus be used for period matrices associated to general nice algebraic curves.

Example 6.2.1. For example, we can use the MAGMA functionality to confirm that the Jacobian of a curve has complex multiplication. Consider an affine model for the smooth plane quartic X_{19} of [51, Case 19], e.g. setting $z = 1$ we can write $X_{19} : f = 0$ with

$$f = -7x^4 - 2x^3y + 16x^3 + 7x^2y^2 - 6x^2y - 8x^2 + 10xy^3 + 14xy^2 \tag{6.2}$$
$$+ 2xy - 15x + y^4 + 10y^3 + 13y^2 + 17y + 14 \in \mathbb{Q}[x, y]\,. \tag{6.3}$$

The authors of [51] constructed this curve such that the endomorphism ring $\mathrm{End}(J(\overline{\mathbb{Q}}))$ is the maximal order of the number field

$$K = \mathbb{Q}[t]/(t^6 - 2t^5 + 102t^4 - 160t^3 + 5845t^2 - 206t + 140932)\,.$$

Indeed, we can verify this by computing a small period matrix τ for X_{19} to reasonable precision, say 300 decimal digits. The MAGMA command EndomorphismRing(τ) then

returns a matrix algebra of degree 6 over $\mathbb{Z}$ with 3 generators

$$R_1 = \begin{pmatrix} 1 & -2 & 1 & 1 & 3 & 0 \\ 1 & 0 & 5 & 5 & 8 & 4 \\ 2 & 0 & -1 & 2 & 2 & 0 \\ 4 & -1 & 2 & 4 & 1 & 2 \\ -1 & -3 & 1 & 0 & 1 & 0 \\ -2 & 0 & -2 & -3 & -4 & -3 \end{pmatrix}, \quad R_2 = \begin{pmatrix} 4 & -2 & -2 & 0 & -2 & -2 \\ 2 & 0 & 1 & 2 & 0 & 2 \\ 4 & 0 & -5 & 2 & -2 & 0 \\ 0 & 0 & 4 & 4 & 2 & 4 \\ 0 & 0 & 1 & -2 & 0 & 0 \\ -4 & -1 & 0 & -2 & 1 & -5 \end{pmatrix},$$

$$R_3 = \begin{pmatrix} 3 & -2 & 0 & 2 & 0 & 0 \\ -1 & 1 & 3 & 2 & 3 & 3 \\ 2 & 3 & 3 & 2 & 1 & 9 \\ -4 & 2 & 3 & 1 & 3 & 2 \\ 2 & -3 & -1 & 0 & -1 & -3 \\ -1 & -2 & -5 & -2 & -2 & -8 \end{pmatrix}.$$

Afterwards, we use the function MinimalPolynomial(R_i) to compute the minimal polynomials of the matrices R_1, R_2, R_3, which we find to generate the number field

$$L = \mathbb{Q}[t]/(t^6 - 2t^5 + 16t^4 + 12t^3 - 46t^2 + 224t + 451)\,.$$

As expected, calling IsIsomorphic(L, K) returns 'true', thus verifying that the Jacobian corresponding to τ does have K as CM field.

6.2.1 Interface with the LMFDB

Having a fast and reliable algorithm to compute period matrices to high precision is a valuable input for the methods developed by Costa et al. [21] to compute endormorphism rings of Jacobians rigorously. In their work they revisit the strategy of van Wamelen [92] and enhance its practical performance. Moreover, their methods apply to curves of arbitrary genus.

During a meeting aimed at expanding the 'L-functions and modular forms database' [58] to include genus 3 curves, we synced our period matrix algorithms with their routines. The MAGMA implementation for period matrices of superelliptic curves, as described in Section 5.2, was incorporated in their framework to successfully compute the endomorphism rings of Jacobians of 67.879 hyperelliptic curves of genus 3, and confirm those of the 66.158 genus 2 curves that are currently in the database. For these applications big period matrices were computed to 300 decimal digits precision.

Moreover, we tailored an extra version of our period matrix algorithm for general algebraic curves, as described in Section 4.8, for the special case of plane smooth quartics (non-hyperelliptic genus 3 curves) simply to gain some further speed-up. This version was used to compute endormorphism rings of Jacobians of 82.244 plane smooth quartics that are going to appear in the LMFDB. The working precision here was 100 decimal digits and at the time when the computation was carried out our algorithm took on average ≈ 10 seconds for each period matrix.

Let us mention here that among these 216.281 curves there was not a single 'bad example' which our algorithms could not handle and that other period matrix algorithms would have been too slow or unstable to carry out this huge computation.

6.2.2 Isogenies between Jacobians

In fact, the case of endomorphism rings is just a special case of more general maps between Jacobians. Let C_1 and C_2 be smooth, projective, geometrically irreducible curves of genus g defined over a number field K with Jacobians $J_1 = \mathrm{Jac}(C_1/K)$ and $J_2 = \mathrm{Jac}(C_2/K)$. An *isogeny* between J_1 and J_2 is a surjective morphism $J_1 \to J_2$ with finite kernel.

In particular, over the complex numbers, the Jacobians can be represented by period matrices Ω_1, Ω_2 such that

$$J_i(\mathbb{C}) \cong \mathbb{C}^g/\Lambda_i \quad \text{where } \Lambda_i = \Omega_i\mathbb{Z}^{2g}, \text{ for } i = 1, 2.$$

In this case, isogenies between the (analytic) Jacobians (*analytic homomorphisms*) can be approximated with MAGMA, using our period matrices as numerical approximations. As an example, we consider the non-split Cartan modular curve $X_{ns}(13)$ which is defined by an affine equation

$$(-y-z)x^3 + (2y^2+yz)x^2 + (-y^3+zy^2-2z^2y+z^3)x + (2z^2y^2-3z^3y) = 0$$

and which is isomorphic to the split Cartan modular curve $X_s(13)$ (those are two of the main results of Baran's paper [5]). In a different paper [4], Banwait and Cremona show that the modular curve $X_{S_4}(13)$ is a genus 3 curve, whose canonical embedding in $\mathbb{P}^2(\mathbb{Q})$ has the following model:

$$4x^3y-3x^2y^2+3xy^3-x^3z+16x^2yz-11xy^2z+5y^3z+3x^2z^2+9xyz^2+y^2z^2+xz^3+2yz^3 = 0\,.$$

Further, by [4, Proposition 9.1], the Jacobians $J_s(13)$ and $J_{S_4}(13)$ of the modular curves $X_s(13)$ and $X_{S_4}(13)$ are $\mathbb{Q}$-isogenous. This implies, of course, that the corresponding analytic Jacobians are isogenous as well.

We can check this now: computing respective small period matrices $\tau_s(13)$ and $\tau_{S_4}(13)$ to high precision (here we used 500 decimal digits) and then applying the MAGMA function 'AnalyticHomomorphisms($\tau_s(13)$,$\tau_{S_4}(13)$)' yields 3 matrices in $\mathbb{Z}^{6\times 6}$

$$M_1 = \begin{pmatrix} 1 & 0 & -3 & 1 & -2 & 0 \\ -2 & -2 & 2 & 1 & -1 & 0 \\ 0 & -1 & -1 & 2 & -1 & -1 \\ -1 & -2 & 0 & 0 & -1 & -2 \\ 0 & 2 & -1 & 0 & -1 & 2 \\ -2 & 1 & 1 & 3 & -1 & 2 \end{pmatrix}, \quad M_2 = \begin{pmatrix} 1 & 0 & 0 & -2 & 1 & -1 \\ -2 & 2 & 3 & 4 & 1 & 3 \\ 0 & -3 & -1 & -1 & -1 & -3 \\ -1 & 1 & 2 & 3 & 1 & 2 \\ 0 & 1 & 1 & -2 & 1 & 1 \\ -1 & -3 & 0 & -2 & -2 & -2 \end{pmatrix},$$

$$M_3 = \begin{pmatrix} 0 & 1 & 3 & -2 & 4 & 0 \\ -1 & -3 & -2 & -1 & -3 & -3 \\ 0 & 1 & 2 & 1 & 2 & 1 \\ 0 & -2 & -1 & 0 & -2 & -1 \\ 0 & 2 & 3 & -1 & 3 & 2 \\ -1 & 3 & 5 & 2 & 4 & 4 \end{pmatrix}.$$

such that, for every M_i there exists a matrix $N_i \in \mathbb{C}^{g\times g}$ with

$$N_i(\tau_s(13), \mathrm{id}) = (\tau_{S_4}(13), \mathrm{id})M_i \quad \text{for } i = 1, 2, 3.$$

By [4, Remark 9.2] the Jacobians J_1 and J_2 are non-isomorphic. Applying the MAGMA function 'IsIsomorphicSmallPeriodMatrices' to the small period matrices $\tau_s(13)$ and $\tau_{S_4}(13)$ returns false, which confirms this result.

6.3 Riemann Theta functions

One of the most important applications of period matrices of Riemann surfaces is the computation of the so-called *Riemann Theta function*. It is a complex-valued function, dependent on two variables z and τ, that is of central interest in mathematics and physics. It has numerous applications in the areas of number theory, the theory of abelian functions and integrable partial differential equations. The following definitions and more background on Theta functions can be found in [68, Chapter 2].

Definition 6.3.1. Let τ be an element of the Siegel upper half-space $\mathcal{H}_g$ and $z \in \mathbb{C}^g$. Then the Riemann Theta function is defined as

$$\Theta(z,\tau) = \sum_{v\in\mathbb{Z}^g} \exp(\pi i v^T \tau v) \exp(2\pi i v^T z)\,.$$

The Theta function with characteristic $[a;b]$ for $a,b \in \frac{1}{2}\mathbb{Z}^g/\mathbb{Z}^g$ is defined as

$$\Theta_{[a;b]}(z,\tau) = \sum_{v\in\mathbb{Z}^g} \exp(\pi i (v+a)^T \tau (v+a)) \exp(2\pi i (v+a)^T (z+b))\,.$$

The values of $\Theta_{[a;b]}(z,\tau)$ at $z = 0 \in \mathbb{C}^g$ are called *Theta-constants*.

Theta functions are implemented in MAGMA and only need a small period matrix as input.

Example 6.3.2. We compute a small period matrix τ for the smooth plane quartic X_{19} from Example 6.2.1

$$\tau = \begin{pmatrix} 0.33087 + 0.42366\cdot I & -0.24629 + 0.23665\cdot I & 0.21438 - 0.29331\cdot I \\ -0.24629 + 0.23665\cdot I & 0.81041 + 0.60848\cdot I & 0.12651 - 0.31965\cdot I \\ 0.21438 - 0.29331\cdot I & 0.12651 - 0.31965\cdot I & 0.69733 + 0.59126\cdot I \end{pmatrix} \in \mathcal{H}_3\,.$$

Using the MAGMA function 'Theta(char,z,τ)' we can compute the *fundamental Theta constants*, which are the 2^g Theta-constants with characteristic $[0,i]$ where $i = 2(b_0 + 2b_1 + \cdots + 2^{g-1}b_{g-1})$, to arbitrary precision. We list these values with 10 digits of precision here

$$\begin{aligned}
\Theta_{[0,0]}(0,\tau) &= 1.251511913 - 0.0316079011\cdot I,\\
\Theta_{[0,1]}(0,\tau) &= 0.253213390 + 0.4663903113\cdot I,\\
\Theta_{[0,2]}(0,\tau) &= 0.676272699 + 0.478508172\cdot I,\\
\Theta_{[0,3]}(0,\tau) &= 0.869771995 + 0.180164185\cdot I,\\
\Theta_{[0,4]}(0,\tau) &= 1.015740610 + 0.675681015\cdot I,\\
\Theta_{[0,5]}(0,\tau) &= 0.440680993 - 0.514531310\cdot I,\\
\Theta_{[0,6]}(0,\tau) &= 2.168688576 + 0.668208329\cdot I,\\
\Theta_{[0,7]}(0,\tau) &= 1.317923705 - 2.014015850\cdot I\,.
\end{aligned}$$

These values could, for example, be used to determine the Dixmier-Ohno invariants [51, Section 2.2] of smooth plane quartics.

In his PhD-thesis [55] Hugo Labrande presents an algorithm that computes the Riemann Theta function and the small period matrix by inverting a certain function using Newton-iteration. It is quasi-linear in the precision and makes it possible to determine these objects with hundreds of thousands digits of precision. Labrande focuses on genus 1 and 2 in his thesis and generalizes his algorithm to genus 3 in [51, Section 2.1]. However, his algorithm is inherently exponential in the genus and thus not well-suited for curves of higher genus. It would be interesting to investigate whether this can be combined with our implementation.

6.3.1 Reduced small period matrix

For a given curve our algorithms compute small period matrices τ in the Siegel upper half-space $\mathcal{H}_g$ which are arbitrary in the sense that they depend on the choice of a symplectic basis made during the algorithm.

For applications like the computation of Theta functions it is useful to have a small period matrix τ in the Siegel fundamental domain $\mathcal{F}_g \subset \mathcal{H}_g$ (see [51, §1.3]). The convergence of the Theta function in greatly improved when $\tau \in \mathcal{F}_g$.

We did not implement any such reduction. The authors of [51] give a theoretical sketch of an algorithm (Algorithm 1.9) that achieves this reduction step, as well as two practical versions (Algorithms 1.12 and 1.14) which work in any genus and have been implemented for $g \leq 3$. Combining this reduction algorithm with our period matrix implementation would certainly be reasonable.

6.3.2 Computing canonical heights

Another interesting application of Theta functions in number theory is the computation of the canonical height of abelian varieties. This can be used to compute the regulator R_J (6.1) appearing in the Birch and Swinnerton-Dyer conjecture. In the end of Section 4 of his paper [67], Steffen Müller explains how one can compute the intersection multiplicities at archimedean places using the Theta function with characteristic $a = (1/2, \ldots, 1/2), b = (g/2, (g-1)/2, \ldots, 1, 1/2)$. Another option is to compute these directly using integration on $C(\mathbb{C})$, see [56, Section 13.7].

6.4 Numerical verification of Beilinson's conjecture

In this section we will apply our methods of integrating differential forms on Riemann surfaces to numerically verify Beilinson's conjecture for the second K-group of curves. In order to formulate the conjecture we will briefly introduce L-functions and the necessary objects from K-theory, closely following Sections 2 and 3 of [29]. We slightly modify their strategy and use our algorithm to compute regulators. Then, we compare our results with the numerical data given in [29, Section 10].

Moreover, we will pick up a superelliptic example that originates from the PhD-thesis of Vinzenz Busch [14], who generalized the construction of [29] to superelliptic curves. Applying our strategy there gives us insight into why these curves are not suitable for checking Beilinson's conjecture.

Lastly, we consider the work of H. Liu and R. de Jeu [23] who construct integral elements for a very general family of curves that includes hyperelliptic as well as non-hyperelliptic examples. Using our algorithms, we provide new numerical evidence for Beilinson's conjecture for several examples of both categories. In particular, we are the first to compute such regulators (up to rational multiples) numerically for non-hyperelliptic examples.

6.4.1 L-functions of algebraic curves

Let C be a non-singular, projective, geometrically irreducible curve of genus g defined over $\mathbb{Q}$. The Dirichlet series which is defined for $\mathrm{Re}(s) > \frac{3}{2}$ by the Euler product

$$L(C, s) = \prod_{p \text{ prime}} L_p(C, s)$$

is called *L-function* of C. The Euler factors $L_p(C, s)$ carry and encode arithmetic infomration about the curve C. For the primes of good reduction the Euler factor $L_p(C, s)$ is defined by

$$L_p(C, s) = \exp\left(\sum_{n=1}^{\infty} (p^n + 1 - \#C(\mathbb{F}_{p^n})) \frac{p^{-ns}}{n}\right),$$

for other primes the definition is more involved and we refer the reader to [46, p.461]. Moreover, we can associate to a curve $C/\mathbb{Q}$ an integer N that is called the *conductor*, see [59] for its definition, which is an important arithmetic invariant of the curve.

What is really important for us here, is the functional equation appearing in the Hasse-Weil conjecture.

Conjecture 6.4.1 (Hasse-Weil)**.** The function

$$L^*(C,s) = \frac{N^{s/2}}{(2\pi)^{gs}}\Gamma(s)^g L(C,s) \tag{6.4}$$

extends to an entire function of s and satisfies the functional equation

$$L^*(C,s) = wL^*(C,2-s) \quad \text{with } w \in \{\pm 1\}. \tag{6.5}$$

If the conjecture holds, then we have $L^{(0)}(C,0) = \cdots = L^{(g-1)}(C,0) = 0$ and

$$\frac{L^{(g)}(C,0)}{g!} = \lim_{s\to 0}\frac{L(C,s)}{s^g} = L^*(C,0) = wL^*(C,2) = \frac{wN}{(2\pi)^{2g}}L(C,2) \neq 0, \tag{6.6}$$

since the Gamma-function has a pole of order one with residue 1 at $s = 0$.

In order to check the Beilinson Conjecture 6.4.2 one needs to know the special value $L^*(C,0)$ up to a rational multiple. If we assume (6.5) then, by (6.6), for this it suffices to know the value $L(C,2)$ and the genus g. For certain hyperelliptic curves this value can be computed numerically in MAGMA just from an affine equation for the curve. MAGMA uses the algorithms presented in [28], which also compute the quantities N,w and the 'bad' Euler factors $L_p(C,s)$ under the assumption that the functional equation (6.5) holds.

For general nice algebraic curves this is a very hard computational problem, especially if the conductor N and the bad Euler factors are unknown. However, a lot of work has been done for hyperelliptic curves, see for example [44] and [50], and there is active research going on to improve the computation of L-series for smooth plane quartics by Andrew V. Sutherland and David Harvey.

6.4.2 K-theory and Beilinson's conjecture

We want to mention that the description of several K-groups we are going to use are the very explicit definitions of [29, Section 3] that are in our situation equivalent to the much more complicated, general definitions that can be found in [76]. For a field F one can define the second K-group of F as

$$K_2(F) = (F^* \otimes F^*)/\langle a\otimes(1-a) \mid a \in F\setminus\{0,1,\}\rangle,$$

a free abelian group where the class of an element $a\otimes b \in (F^*\otimes F^*)$ in $K_2(F)$ will be denoted $\{a,b\}$.

Let C be a non-singular, projective, geometrically irreducible curve of genus g defined over $\mathbb{Q}$ and denote by $F = \mathbb{Q}(C)$ its function field. We can then describe a certain 'tame' quotient group $K_2^T(C)$ of $K_2(C)$ as

$$K_2^T(C) = \ker\left(K_2(L) \xrightarrow{T} \bigoplus_{x\in C(\overline{\mathbb{Q}})} \overline{\mathbb{Q}}^*\right),$$

where the x-component of the map T is the *tame symbol* at x which is defined by

$$T_x : \{a,b\} \mapsto (-1)^{\mathrm{ord}_x(a)\,\mathrm{ord}_x(b)}\frac{a^{\mathrm{ord}_x(b)}}{b^{\mathrm{ord}_x(a)}}(x).$$

We obtain a map from $F^* \otimes_{\mathbb{Z}} F^*$ to the group of almost everywhere defined 1-forms on the Riemann surface $X = \mathcal{C}(\mathbb{C})$ by assigning to an element $a \otimes b$ the differential form

$$\eta(a,b) = \log|a|\,\mathrm{d}(\arg b) - \log|b|\,\mathrm{d}(\arg a). \tag{6.7}$$

It can be shown [14, Lemma 4.3.2] that this is a well defined, closed, smooth (real-analytic) 1-form outside the finite set of zeros and poles of a and b, denoted by $\mathcal{S} \subset X$. Moreover, integrating these differentials along cycles on $X \setminus \mathcal{S}$ gives us a pairing

$$\langle \cdot, \cdot \rangle : H_1(X,\mathbb{Z}) \times K_2^T(C) \to \mathbb{R},\ \langle \gamma, \{a,b\} \rangle := \frac{1}{2\pi} \int_\gamma \eta(a,b)\,. \tag{6.8}$$

The homology group can be split up into two subgroups, each of rank g, namely

$$H_1(X,\mathbb{Z}) \cong H_1(X,\mathbb{Z})^+ + H_1(X,\mathbb{Z})^-$$

where $H_1(X,\mathbb{Z})^+$ contains the cycles that are invariant under complex conjugation on X and $H_1(X,\mathbb{Z})^-$ the anti-invariant ones respectively. For $\gamma \in H_1(X,\mathbb{Z})^+$, $\langle \gamma, \cdot \rangle$ vanishes identically on $K_2^T(\mathcal{C})$, so we restrict the pairing (6.8) to

$$\langle \cdot, \cdot \rangle : H_1(X,\mathbb{Z})^- \times K_2^T(C)/\text{torsion} \to \mathbb{R}\,, \tag{6.9}$$

which is then called the *regulator pairing*. In order to formulate Beilinson's conjecture we have to restrict this pairing even further by imposing some integrality condition on the elements of $K_2^T(C)$.

Let $\mathcal{C}$ be a regular proper model of C over $\mathbb{Z}$. For such a model we define

$$K_2^T(\mathcal{C}) = \ker\left(K_2^T(C) \xrightarrow{T} \bigoplus_{p, \mathcal{D} \subset \mathcal{C}_p} \mathbb{F}_p(\mathcal{D})^* \right),$$

where $\mathcal{C}_p$ denotes the fiber of $\mathcal{C}$ over the prime p and $\mathcal{D}$ runs over the irreducible components of the curve $\mathcal{C}_p$ with $\mathbb{F}_p(\mathcal{D})$ denoting the function field of $\mathcal{D}$. The corresponding tame symbol T is given by the discrete valuation that $\mathcal{D}$ induces on F. Finally, we define

$$K_2(C,\mathbb{Z}) = K_2^T(\mathcal{C})/\text{torsion}$$

which is a subgroup of $K_2^T(C)/\text{torsion}$ and is, by [23, Proposition 4.1], independent of the choice of the regular, proper model $\mathcal{C}$. Restricting the pairing (6.9) to

$$< \cdot, \cdot > : H_1(X,\mathbb{Z})^- \times K_2(C,\mathbb{Z}) \to \mathbb{R}\,, \tag{6.10}$$

we can formulate Beilinson's conjecture as in [29, Conjecture 3.11].

Conjecture 6.4.2 (Beilinson's conjecture)**.** Let C be a non-singular, projective, geometrically irreducible curve of genus g defined over $\mathbb{Q}$ and $X = C(\mathbb{C})$ the associated Riemann surface. Then,

(i) the group $K_2(C,\mathbb{Z})$ is a free abelian group of rank g and the pairing (6.10) is non-degenerate;

(ii) let R denote the absolute value of the determinant of this pairing with respect to $\mathbb{Z}$-bases of $H_1(X,\mathbb{Z})^-$ and $K_2(C,\mathbb{Z})$ and let $L^*(C,0)$ be defined as in (6.4), then $L^*(C,0)/R \in \mathbb{Q}^*$.

6.4.3 Strategy for checking Belinson's conjecture

In order to check Conjecture 6.4.2, the authors of [29] used the following strategy:

(1) Try to construct a set of g integral elements $M = \{M_k \mid k = 1, \dots, g\} \subset K_2(C, \mathbb{Z})$.

(2) Compute the absolute value of the determinant of the pairing (6.10) with respect to a basis of $H_1(X, \mathbb{Z})^-$ and M; we call this R_M.

(3) If R_M is non-zero, then $\mathrm{rk}(K_2(C, \mathbb{Z})) \geq g$ and we can try to recognize $L^*(C, 0)/R_M$ as a rational number.

(4) If we have more than g integral elements, we can also check that the resulting maps $\langle \cdot, M_k \rangle \to \mathbb{R}$ are linearly dependent over $\mathbb{Z}$, thus providing numerical evidence for $\mathrm{rk}(K_2(C, \mathbb{Z})) \leq g$.

In order to find non-trivial relations with integer coefficients between the elements in step (4), we use the MAGMA function 'LinearRelation'.

Since we want to use as much of our period matrix algorithm as possible, we modify step (2) of this strategy as follows:

Instead of computing the pairing for a basis of $H_1(X, \mathbb{Z})^-$, we integrate the differentials defined by the elements in M along a full homology basis $\gamma_1, \dots, \gamma_{2g}$ of $H_1(X, \mathbb{Z})$, resulting in a matrix

$$P_M = \big(\langle \gamma_l, M_k \rangle\big)_{\substack{1 \leq k \leq g, \\ 1 \leq l \leq 2g}} \in \mathbb{R}^{g \times 2g}\,. \tag{6.11}$$

In fact, we will be integrating along a canonical homology basis as in Definition 2.7.3. If $\mathrm{rk}(P_M) = g$ we denote by R_M the minimal absolute value of all non-zero minors of maximal rank of P_M and, provided we know the value $L^*(C, 0)$, continue with step (3). Otherwise, we have $\mathrm{rk}(P_M) < g$ which indicates that the elements in M are linearly dependent or (unlikely) that the Conjecture is wrong.

The reason why this works is that the anti-invariant part $H_1(X, \mathbb{Z})^-$ is isomorphic to the quotient $H_1(X, \mathbb{Z})/H_1(X, \mathbb{Z})^+$ via the map

$$H_1(X, \mathbb{Z}) \to H_1(X, \mathbb{Z})^-,\ \gamma \mapsto \gamma - c^*(\gamma)\,,$$

where c^* denotes complex conjugation on X. As Busch shows in [14, Lemma 5.10.1] the images of $\gamma_1, \dots, \gamma_{2g}$ under this map generate a finite index subgroup H of $H_1(X, \mathbb{Z})^-$. For any $\gamma \in H_1(X, \mathbb{Z})$ and $\{a, b\} \in K_2^T(C)/\text{torsion}$ we have that

$$2\langle \gamma, \{a, b\} \rangle = \langle \gamma, \{a, b\} \rangle - \langle c^*(\gamma), \{a, b\} \rangle = \langle \gamma - c^*(\gamma), \{a, b\} \rangle\,.$$

Therefore, a rational non-zero multiple of R_M corresponds to integration along a basis of $H_1(X, \mathbb{Z})^-$ and our modification of step 2 does not conflict with our goal of checking Beilinson's conjecture. Another strong numerical check on the correctness of our computations is that all non-zero minors of maximal rank ought to be rational multiples of each other.

6.4.4 Numerical integration

Here we want to explain how to actually compute the pairing (6.8). Assume that our curve is given by an affine model $f(x, y) = 0$ with $f \in \mathbb{Q}[x, y]$ and we are given an element $\{a, b\} \in K_2(F)$. The task is to integrate $\eta(a, b)$ along the vector of lifts $\tilde{\gamma} = \{(x, y(x)) \mid x \in \gamma([-1, 1])\}$ obtained from analytic continuation (see Section 4.5) of a smooth path $\gamma : [-1, 1] \to \mathbb{C}$ in the x-plane that avoids the set $\mathcal{S}$ (zeros and poles of a and b) as well as the exceptional values $\mathcal{L}(\varphi_x)$. Analogously to the integration of holomorphic differentials (see Section 4.7) we choose a suitable integration scheme and compute the corresponding parameters. The only actual difference is the evaluation of the differential $\eta(a, b)$, which we will now describe.

Parametrization of $\eta(a,b)$ Recall, that we want to evaluate the vector of integrals

$$\int_\gamma \eta(a,b) = \int_\gamma \log|a|\,\mathrm{d}(\arg b) - \log|b|\,\mathrm{d}(\arg a)\,.$$

First we want to write the differential $\mathrm{d}\arg$ in a more tractable way. For a complex number $z = re^{i\varphi} \in \mathbb{C}\setminus\{0\}$ we have that

$$\frac{\mathrm{d}z}{z} = \mathrm{d}(\log z) = \mathrm{d}(\log r) + \mathrm{d}i\varphi \quad \Rightarrow \quad \mathrm{d}(\arg z) = \operatorname{Im}\left(\frac{\mathrm{d}z}{z}\right).$$

Since we want to parametrize $\mathrm{d}x$, we can write $\mathrm{d}y = -\frac{\partial_x f}{\partial_y f}\mathrm{d}x$. Moreover, we have that $a(x,y) \in F^*$ and therefore

$$\begin{aligned}\mathrm{d}a(x,y) &= \partial_x a(x,y)\mathrm{d}x + \partial_y a(x,y)\mathrm{d}y\\ &= \left(\partial_x a - \frac{\partial_x f}{\partial_y f}\partial_y a\right)(x,y)\mathrm{d}x,\end{aligned}$$

and analogously for $b(x,y)$. Finally, parametrizing $(x,y) = (\gamma(t), y(\gamma(t)))$, the pairing can be computed as

$$\langle\tilde{\gamma}, \{a,b\}\rangle = \frac{1}{2\pi}\int_{-1}^{1} \log|a(t)|\operatorname{Im}\left(\frac{\mathrm{d}b(t))}{b(t)}\right) - \log|b(t)|\operatorname{Im}\left(\frac{\mathrm{d}a(t))}{a(t)}\right),$$

where

$$\begin{aligned}a(t) &= a(\gamma(t), y(\gamma(t)),\\ \mathrm{d}a(t) &= \left(\partial_x a - \frac{\partial_x f}{\partial_y f}\partial_y a\right)(\gamma(t), y(\gamma(t)))\,\gamma'(t)\,\mathrm{d}t,\end{aligned}$$

and $b(t), \mathrm{d}b(t)$ are defined analogously. This formula has already been used by [29] and [14] without justification in slightly different variations.

Computing a homology basis As already indicated above, to obtain a homology basis we use the algorithms from Sections 4.3 and 4.6 that were already applied for computing period matrices. The only modification: we need to add the x-coordinates of all (finite) zeros and poles of the differential $\eta(a,b)$ to the set $\mathcal{L}$ of exceptional values, which are the points that have to be avoided for numerical integration. Note that adding extra points will not affect the monodromy representation (and therefore the homology basis).

6.4.5 Examples of de Jeu, Dokchitser & Zagier

In [29] the authors construct enough ($\geq g$) elements for several families of hyperelliptic curves. In particular, their Construction 6.11 yields hyperelliptic curves of genus g of the form

$$y^2 + h(x)y + x^d = 0 \tag{6.12}$$

with either

$$\begin{aligned}&h(x) = c_g x^g + \ldots, c_1 x + c_0 \in \mathbb{Z}[x] \text{ and } d = 2g+1 \text{ or}\\ &h(x) = 2x^{g+1} + c_g x^g + \cdots + c_1 x + c_0 \in \mathbb{Z}[x], c_0 \neq 0 \text{ and } d = 2g+2,\end{aligned} \tag{6.13}$$

such that $t(x) = -x^d + h(x)^2/4$ is separable. Note that any curve defined this way is isomorphic to the curve defined by

$$y^2 = t(x)\,.$$

The polynomial $t(x)$ is called the 2-torsion polynomial because its roots give rise to the 2-torsion points on the Jacobian of the curve.

Theorem 6.4.3. [29, Theorem 8.1.]
Let $C/\mathbb{Q}$ be defined by (6.12) *and* (6.13). *Let $m(x) \in \mathbb{Z}[x]$ be a non-constant factor, irreducible over $\mathbb{Z}$, of the* 2*-torsion polynomial $t(x) = -x^d + h(x)^2/4$. Then*

$$\left\{\frac{y^2}{x^3}, \frac{m(x)}{m(0)}\right\} \in K_2^T(C)$$

and, denoting by $\tilde{M}$ its class modulo torsion, we have

(1) if $m(0) = \pm 1$ then $\tilde{M} \in K_2(C, \mathbb{Z})$,

(2) if there is a prime dividing $m(0)$ but not every c_i then $n\tilde{M} \notin K_2(C, \mathbb{Z})$ for any $n \neq 0$.

Moreover, if $c_0 = \pm 1$ then $\{-g(0)y, -x\} \in K_2^T(C)$. Let $\mathbb{M}$ denote its class modulo torsion, then

$$\begin{cases} \mathbb{M} \in K_2(C, \mathbb{Z}), & \text{if } d = 2g+1, \\ 2\mathbb{M} \in K_2(C, \mathbb{Z}), & \text{if } d = 2g+2. \end{cases}$$

We tested our algorithm against the results of [29] in many of their examples, following our modified strategy as explained in §6.4.3. For all examples the integral elements in M where chosen exactly as in [29], in particular $\#M \geq g$. For every example the matrix $P_M \in \mathbb{R}^{\#M \times 2g}$ that we computed had rank g; for some examples the corresponding regulator R_M is given in the tables below. Moreover, all g-minors of P_M (including R_M) could be recognized as rational multiples of $L^*(C, 0)$ (values taken from the tables of loc. cit.). These observations confirm the correctness of our integration routines and, of course, the results of [29].

Example 6.4.4. [29, Example 10.1] The first family that the authors of [29] consider is a one-parameter family of curves of genus 2

$$C_b : y^2 + ((4b-3)x^2 + (5b-1)x + b)y + x^5 = 0\,, \quad b \in \mathbb{Z} \setminus \{0\}.$$

The 2-torsion polynomial $t(x)$ factors over $\mathbb{Z}$ (up to a constant) as

$$m_1(x)m_2(x)m_3(x) = (x-1)(4x-1)(x^3 - (4b^2 - 6b + 1)x^2 + (5b^2 - 2b)x - b^2).$$

In particular, there is a typo in [29] in the linear term of $m_3(x)$: they write $(5b^2 - b)$. From Theorem 6.4.3 we obtain the following integral elements

$$M = \begin{cases} \{M_1, M_2\}, & \text{if } b \in \mathbb{Z} \setminus \{-1, 0, 1\}, \\ \{M_1, M_2, M_3, \mathbb{M}\}, & \text{if } b = \pm 1. \end{cases}$$

Table 6.1 shows the values for R_M (18 digits precision) that have been computed as explained in 6.4.3 for different values of b. We found that $\mathrm{rk}(P_M) = 2$ and the ratio $L^*(C, 0)/R_M$ could be recognized as a rational number for all values of b; the values for $L^*(C, 0)$ have been taken from [29, Table 3]. It is worth mentioning that in every case, except $b = \pm 1$, we computed exactly the same regulator as [29], which they denote by $R(\Lambda)$, i.e. $R_M = R(\Lambda)$. Moreover, we were able to numerically confirm the relations

$$\begin{aligned} 41M_1 + 56M_2 - 44M_3 = 0 \quad &\text{for } b = -1, \\ 4M_1 - 26M_2 + 29M_3 = 0 \quad &\text{for } b = 1. \end{aligned}$$

b	Conductor	R_M	$L^*(C,0)/R_M$
-10	$2^2 \cdot 3 \cdot 5^2 \cdot 7 \cdot 11^2 \cdot 1031$	77.130 134 889 287 138 999	$2^2 \cdot 3 \cdot 5^2 \cdot 7$
-9	$3^3 \cdot 23 \cdot 8461$	73.537 647 438 966 012 020	$2^2 \cdot 3^3$
-8	$2^2 \cdot 3 \cdot 61 \cdot 6113$	69.634 990 149 626 965 070	$3 \cdot 23/2$
-7	$3 \cdot 7^2 \cdot 53 \cdot 4243$	65.355 609 002 659 819 789	$13 \cdot 19$
-6	$2^2 \cdot 3^3 \cdot 5 \cdot 2797$	60.607 437 952 449 742 454	$2 \cdot 3^2$
-5	$3 \cdot 5^2 \cdot 37 \cdot 1721$	55.257 255 220 919 193 247	29
-4	$2^2 \cdot 3 \cdot 29 \cdot 31^2$	49.100 257 337 811 396 017	2^2
-3	$3^3 \cdot 7 \cdot 463$	41.793 138 137 745 545 385	1
-2	$2^2 \cdot 3 \cdot 13 \cdot 172$	32.676 060 511 451 741 457	$1/2$
-1	$3 \cdot 5 \cdot 37$	0.684 936 949 938 389 325	$1/3$
1	$3 \cdot 11^2$	0.649 248 173 594 350 399	$1/5$
2	$2^2 \cdot 3 \cdot 13 \cdot 19$	19.031 931 751 296 953 262	$1/(2^2 \cdot 5)$
3	$3^3 \cdot 47$	33.576 646 858 439 342 020	$1/(2 \cdot 3^3)$
4	$2^2 \cdot 3 \cdot 5 \cdot 7 \cdot 223$	42.908 137 312 458 286 852	1
5	$3 \cdot 5^2 \cdot 43 \cdot 569$	50.179 539 249 499 433 184	2^4
6	$2^2 \cdot 3^3 \cdot 17^2 \cdot 67$	56.265 553 014 245 968 261	$2^2 \cdot 7$
7	$3 \cdot 7^2 \cdot 59 \cdot 1987$	61.544 348 463 785 563 610	$2^2 \cdot 5^2$
8	$2^2 \cdot 3 \cdot 67 \cdot 3167$	66.227 862 684 148 830 809	3^3
9	$3^3 \cdot 5 \cdot 4733$	70.450 208 338 642 626 931	2^2
10	$2^2 \cdot 3 \cdot 5^2 \cdot 23 \cdot 83 \cdot 293$	74.302 797 336 615 458 561	$2^7 \cdot 3^2$

Table 6.1: R_M for the curves C_b with 18 digits precision.

Remark 6.4.5. It was straightforward to check many more of the examples of [29]. We briefly summarize the results here.

1. Example 10.5 (loc. cit.): For the genus 2 curves given by
$$C_b : y^2 + (2x^3 - 4bx^2 - x + b)y + x^6 = 0\,, \quad b \in \mathbb{Z}_{>0},$$
we obtained exactly the same regulators for all examples in Table 4 ($b = 1, \ldots, 14$), i.e. $R_M = R(\Lambda)$ and confirmed the relation for $b = 1$.

2. Example 10.6 (loc. cit.): For the genus 3 curves given by
$$C_{a,b} : y^2 + ((4b-3)x^3 - (4a+5b-1)x^2 + (5a+b)x - a)y + x^7 = 0\,, \quad b \in \mathbb{Z}_{>0}$$
our regulators were different for all examples in Table 5 (mostly by a factor $1/3$ or $1/6$), we confirmed the relation for $a, b = 1, 3$.

3. Example 10.7 (loc. cit.): For the sporadic examples of type
$$y^2 + h(x)y + x^d = 0$$
we obtain the same regulators for all examples in Table 6 ($d = 10, g = 4$), except a factor 2 for the last curve; for all examples in Table 7 ($d = 12, g = 5$) we computed $R_M = 2R(\Lambda)$.

4. Example 10.8 (loc. cit.): we checked quite a few examples of the Tables 8,9,10,11 which contain curves of genera 2,3 and 4, given by equations of the form
$$y^2 + \Big(2x^{g+1} \pm \prod_{j=1}^{g}(\alpha_j x + 1)\Big)y + x^{2g+2} = 0, \quad \text{with } v_j \in \mathbb{Z}, \text{ distinct.}$$
As for previous examples, it regularly happens that our regulator is either the same or differs by a factor $2^{\pm 1}$.

6.4.6 Example of Busch

In his PhD-thesis [14] Vincenz Busch generalizes the construction of [29] to obtain elements of $K_2(C,\mathbb{Z})$ from hyperelliptic to superelliptic curves. Busch shows that for almost all superelliptic curves his construction will not yield enough elements to check Beilinson's conjecture. Yet he manages to find a family of superelliptic genus 3 curves for which there is hope that the conjecture can be checked. We state a corrected and modified version of his Theorem 5.5.1. as

Lemma 6.4.6. *For $a,b \in \mathbb{Z}\setminus\{0\}$ such that $a \neq \pm b$ and*

$$\begin{aligned} c_0 &= ab(a^2+ab+b^2),\\ c_1 &= -(a+b)(a^2+b^2)/c_0 \end{aligned}$$

the superelliptic curve of genus 3 defined by the affine equation

$$C_{a,b} : y^3 = c_0^3x^4 + (c_0c_1x+1)^3 \tag{6.14}$$

has 3 elements in $K_2^T(C)$ for which a priori no relations are known.

Proof. Under the transformation $y \mapsto y/c_0$ equation (6.14) becomes

$$y^3 = x^4 + (c_1x + c_0^{-1})^3\,. \tag{6.15}$$

Denoting $z = -\frac{1}{2}(a+b) + i\sqrt{\frac{1}{2}(a^2+b^2+\frac{1}{2}(a+b)^2)}$ we have that $\|z\|^2 = a^2+ab+b^2$ and therefore

$$\begin{aligned} c_0 &= ab\|z\|^2 = abz\bar{z},\\ c_1 &= -\frac{(a+b)(\|z\|^2-ab)}{ab\|z\|^2} = -(a^{-1}+b^{-1}+z^{-1}+\bar{z}^{-1}) \end{aligned}$$

and equation (6.15) becomes

$$y^3 = x^4 + (-(a^{-1}+b^{-1}+z^{-1}+\bar{z}^{-1})x + (abz\bar{z})^{-1})^3$$

which is exactly the form of [14, Proposition 5.3.2] and [14, Construction 5.2.3] yields 6 elements of $K_2^T(C)$, of which at most 3 are linearly independent by [14, Remark 5.3.3]. The 3-torsion polynomial then factors over $\mathbb{C}$ as

$$t(x) = c_0^3(x-a^{-3})(x-b^{-3})(x-z^{-3}))(x-\bar{z}^{-3})$$

which gives us, by assumption, 3 irreducible factors over $\mathbb{Z}$

$$t(x) = m_1(x)m_2(x)m_3(x) = (-a^3x+1)(-b^3x+1)((z\bar{z})^3x^2 - 2\,\mathrm{Re}(z^3)x+1).$$

The corresponding elements in $K_2^T(C)$ are given by

$$M_k = \left\{ \frac{(y-(c_0c_1x+1))^3}{c_0^3x^4}, m_k(x) \right\}, \quad k=1,2,3.$$

□

In the original version, Busch's Theorem 5.5.1 also includes the integrality of the elements M_k, but we leave this issue aside for now. Busch's main result is that he computes the value $L^*(C,0)$ and the regulator for the curve $C_{1,3}$. Due to problems with numerical integration he obtains the corresponding quotient with only one decimal digit of precision

which does not suffice to recognize it as a rational number. According to Busch, for the curves $C_{a,b}$ numerical integration is hard because the branch points are located very close to each other. We suspected that this did indeed cause his attempt of checking Beilinson's conjecture to fail.

So we computed our regulator R_M for the elements $M = \{M_1, M_2, M_3\}$ from Lemma 6.4.6 for several of the curves $C_{a,b}$, using the strategy from §6.4.3. The resulting values are shown in Table 6.2. In fact, Busch was right in the regard that computing the period matrix P_M for the curves $C_{a,b}$ is hard, though far from impossible, because the branch points tend to cluster around zero. Our period matrix algorithm automatically switched to a combination of double-exponential integration and Gauss-Legendre quadrature to compute these integrals in reasonable time, even for hundreds of digits.

a, b	R_M	a, b	R_M
1,2	31.044 543 474 288 968 388	-1,2	25.348 438 072 231 322 538
1,3	40.665 955 143 700 335 257	-1,3	36.363 774 364 414 504 998
1,4	47.169 072 205 286 050 227	-1,4	43.700 392 657 710 022 477
1,5	52.188 949 156 473 532 024	-1,5	49.301 309 655 260 796 512
2,3	23.001 343 705 040 798 238	-2,3	16.104 204 858 224 340 977
2,5	36.455 988 164 949 251 694	-2,5	31.572 969 951 204 028 232
3,4	18.940 928 025 238 370 226	-3,4	11.756 398 702 903 917 549
3,5	26.168 494 808 977 806 759	-3,5	19.701 329 232 920 098 621
4,5	16.355 307 853 195 068 869	-4,5	9.234 466 974 524 185 496

Table 6.2: R_M for the curves $\mathcal{C}_{a,b}$ with 18 digits precision.

The values of Table 6.2 strongly indicate that the elements M_1, M_2 and M_3 from Lemma 6.4.6 are indeed linearly independent. The next step would be to check the ratio of R_M by $L^*(C_{a,b}, 0)$. Unfortunately, computing special values of the L-function of genus 3 curves without knowledge of the conductor, the root number and the bad Euler factors is currently not possible. In his thesis, Busch put a lot of effort into calculating $L^*(C, 0)$ for his main example, namely the curve

$$C_{1,3} : y^3 = 39^3x^4 + (-40x + 1)^3.$$

In his Chapter 3 he manages to compute the value $L(C_{1,3}, 2)$ with 20 digits of precision

$$L(C_{1,3}, 2) = 1.0432103453059890940\,.$$

Experimentally he also computes the conductor $N = 2^6 \cdot 3^5 \cdot 13^2 \cdot 17^2$ and the root number $w = 1$. Thus, using (6.6) he obtains

$$L^*(C_{1,3}, 0) = \frac{N}{(2\pi)^6} L(C_{1,3}, 2) \approx 12878.446259004075748\,.$$

We computed the ratio of this value by the corresponding regulator R_M from Table 6.2

$$L^*(C_{1,3}, 0)/R_M \approx \frac{12878.446259004075748}{40.665955143700335257} \approx 316.68864566185181670\,,$$

which MAGMA could not recognize as a rational number (with denominator smaller than 10^{100000}). Having another look at the proof of his Theorem 5.5.1 we discovered that there is a mistake in the application of his Theorem 5.4.1 which is used to show the integrality of the elements in M. The numerical results strongly suggest that the elements are not all integral. Moreover, we found it impossible to construct a different example with at least g integral elements. Hence we conclude that there seem to be no interesting examples of superelliptic curves (that are not hyperelliptic) for which Beilinson's conjecture can be checked numerically using the construction of [29].

6.4.7 Examples of Liu & de Jeu

The authors of [23] construct families of algebraic curves defined over number fields of arbitrary genus with g elements in $K_2(C, \mathcal{O}_K)$ (same definition as for $K = \mathbb{Q}$). They show that, for a subclass of their curves, these elements are independent by a limit calculation of the regulator. Moreover, they determine which of these curves are hyperelliptic and which are not. In particular, they obtain two families of hyperelliptic curves. One contains the family of [29, Example 10.8.] as a special case, the other gives entirely different curves. First we will briefly present their construction, including affine models for the curves and formulas for elements of $K_2^T(C)$. Afterwards, we will check Beilinson's conjecture numerically using our strategy for several curves defined over $\mathbb{Q}$. In §6.4.7 we cover hyperelliptic curves of genus $2, 3$ and 4 and in §6.4.7, we calculate our regulator R_M for some non-hyperelliptic curves of higher genera, including $4, 5, 6, 8, 9, 10$ and 12.

Construction

The authors of [23] start by considering affine curves of the form $f(x, y) = 0$, defined over a number field K, where

$$f(x,y) = \lambda \prod_{i=1}^{N} \prod_{j=1}^{N_i} L_{i,j} - 1 \in K[x,y], \quad \lambda \neq 0 \tag{6.16}$$

with $N \geq 2$, $N_i \geq 1$ for $i = 1, \ldots, N$ and $L_{i,j} = a_i x + b_i y + c_{i,j}$ is a non-constant polynomial such that the lines defined by $L_{i,j} = 0$ are distinct and non-parallel for $i \neq j$.

In the following C will always denote the normalization of the projective closure in $\mathbb{P}^2$ of the affine curve defined by $f(x, y) = 0$. In [23, Lemma 2.2] the authors define two families of elements of $K_2^T(C)$, namely

$$\begin{aligned} R_{i,j,k,l,m,n} &= \left\{ \frac{L_{i,j}}{L_i, k}, \frac{L_{l,m}}{L_{l,n}} \right\}, \quad i \neq l, \\ T_{i,j,k,l,m,n} &= \left\{ \frac{[i,m]}{[k,m]} \frac{L_{k,l}}{L_i, j}, \frac{[i,k]}{[m,k]} \frac{L_{m,n}}{L_{i,j}} \right\}, \quad i,k,l \text{ distinct} \end{aligned} \tag{6.17}$$

where $[i, k] = a_i b_k - a_k b_i$. In the next step they show that these elements satisfy several relations and obtain a generating set:

Proposition 6.4.7. [23, Proposition 2.2] *Let V be the subgroup of $K_2^T(C)$ generated by all the elements $R_{i,j,k,l,m,n}$ and $T_{i,j,k,l,m,n}$. Then V is already generated by the following elements:*

$$\begin{aligned} &R_{1,1,j,2,1,m}, \quad 1 < j \leq N_1, 1 < m \leq N_2; \\ &T_{1,1,k,l,m,n}, \quad 2 \leq k < m \leq N, 1 \leq l \leq N_k, 1 \leq n \leq N_m; \\ &T_{1,j,2,1,m,n}, \quad 2 \leq j \leq N_1, 3 \leq m \leq N, 1 \leq n \leq N_m \,. \end{aligned}$$

In the following, denote by M the set consisting of these elements.

By their Remark 3.3, the set M contains exactly

$$(N_1 - 1)(N_2 - 1) + \sum_{2 \leq k < m \leq N} N_k N_m + (N_1 - 1) \sum_{m=3}^{N} N_m \tag{6.18}$$

elements, which is exactly the genus of C if the affine curve $C_f : f = 0$ given by (6.16) is non-singular.

For the integrality of the elements (6.17) they restrict to the cases where $N = 2$ or 3. Hence, from here on we only consider curves defined by polynomials of the form

$$f(x,y) = \lambda \prod_{i=1}^{N_1}(x+\alpha_i) \prod_{j=1}^{N_2}(y+\beta_j) \prod_{k=1}^{N_3}(y-x+\gamma_k) - 1 \in K[x,y], \quad \lambda \neq 0 \tag{6.19}$$

where $N_1 \geq N_2 \geq N_3$ and $\alpha_i \neq \alpha_j, \beta_i \neq \beta_j, \gamma_i \neq \gamma_j$ for $i \neq j$. Moreover, we will assume $f(x,y)$ to be defined over the rationals, i.e. $K = \mathbb{Q}$. Thus, we get

Theorem 6.4.8. [23, Theorem 4.2.] *With the notation above, assume that $\alpha_i, \beta_j, \gamma_k$ and $\lambda \neq 0$ are integers. Then the elements given by* (6.17) *are integral, i.e. in $K_2^T(C, \mathbb{Z})$.*

They also provide a simple criterion to check whether C is hyperelliptic or not.

Proposition 6.4.9. [23, Proposition 5.1] *Suppose the affine curve defined by $f(x,y) = 0$ with $f(x,y)$ as in* (6.19) *is non-singular and that C has positive genus. Then C is hyperelliptic if and only if either $N_2 = N_3 = 1$, or $N_2 = 2$ and $N_3 = 0$.*

Hyperelliptic examples

In this section we want to numerically verify Beilinson's conjecture for some hyperelliptic examples. Since we want to use MAGMA for the computation of the L-series, we limit ourselves to examples where this is possible up to a reasonable precision. Note that the numerical integration is far from being the bottleneck here, neither for precision nor for the complexity of the curves (genus, size of coefficients, configuration of exceptional values).

By Proposition 6.4.9 it makes sense to distinguish two cases here. Firstly, if $N_2 = 2$ and $N_3 = 0$, then by a transformation we can find an equation for the curve

$$y\left(y + 2x^{g+1} + \lambda \prod_{i=1}^{g}(\alpha_i x + 1)\right) + x^{2g+2} = 0 \tag{6.20}$$

such that the corresponding elements from (6.17) become

$$M_i = \left\{-\frac{x^{g+1}}{y}, \alpha_i x + 1\right\} \quad i = 1, \ldots, g.$$

For $\lambda = \pm 1$, these curves and elements correspond to those considered in [29] such that $2M_i$ is the element for the factor α_i from Example 10.8. of loc. cit. As explained in Remark 6.4.5 we checked many of the examples listed in Table 8 of [29] and were able to confirm their results. In Table 6.3 we give new numerical evidence for Beilinson's conjecture for genus 2 curves that are not isomorphic to the ones appearing in [29, Table 8].

Secondly, if $N_2 = N_3 = 1$, we have an affine model

$$y\left(y + 2x^{g+2} + \lambda \prod_{i=1}^{g}(\alpha_i x + 1)\right) + x^{2g+4} = 0 \tag{6.21}$$

with corresponding elements

$$M_i = \left\{-\frac{x^{g+2}}{y}, \alpha_i x + 1\right\} \quad i = 1, \ldots, g.$$

For curves given by (6.21) we can easily verify that an example is 'new' (i.e. different from the curves of Example 10.8. of [29]) by checking that they do have exactly g rational branch

points (as opposed to $\geq g+1$ branch points) which is the case for all curves listed in the tables below. Thus, we give new numerical evidence for Conjecture 6.4.2 for hyperelliptic curves with genus 2 in Table 6.4, with genus 3 in Table 6.5 and with genus 4 in Table 6.6.

α_1,α_2	λ	Conductor	R_M	$L^*(C,0)/R_M$
-9,3	-2	$2^8 \cdot 3^4 \cdot 17 \cdot 53$	40.157 243 446 266 171 897	$2^5 \cdot 3^2$
	2	$2^8 \cdot 3^6 \cdot 347$	40.445 929 991 236 018 821	$2^5 \cdot 3 \cdot 11$
-8,2	-2	$2^6 \cdot 6 \cdot 5077$	30.586 152 267 121 164 246	2^5
	2	$2^5 \cdot 5 \cdot 7669$	31.347 959 201 373 203 205	$2^2 \cdot 5$
-4,5	2	$2^8 \cdot 3 \cdot 5 \cdot 13 \cdot 607$	34.076 500 221 997 100 065	$2^4 \cdot 3 \cdot 13$
	3	$2^4 \cdot 3^5 \cdot 5^2 \cdot 239$	39.027 663 260 076 810 053	$2^5 \cdot 3^2$
-3,-7	-3	$2^4 \cdot 3^5 \cdot 7 \cdot 1259$	29.940 531 111 407 665 940	$2^6 \cdot 3^2$
	3	$2^3 \cdot 3^5 \cdot 7 \cdot 37^2$	30.081 874 629 038 140 150	$2^4 \cdot 3 \cdot 7$
-2,2	10	$2^7 \cdot 5^4 \cdot 11 \cdot 13$	25.263 978 498 214 499 023	$2^3 \cdot 3 \cdot 11$
	12	$2^8 \cdot 3^6 \cdot 11 \cdot 23$	27.151 528 096 204 943 380	$2^7 \cdot 3^2$
-1,-5	-6	$2^8 \cdot 3^6 \cdot 5 \cdot 43$	19.541 844 289 094 918 749	$2^4 \cdot 3 \cdot 29$
	6	$2^8 \cdot 3^6 \cdot 5 \cdot 151$	21.392 136 921 430 867 515	$2^6 \cdot 3^2 \cdot 7$
-1,1	11	$2^3 \cdot 11^4 \cdot 13$	8.431 383 875 986 209 411	$2^2 \cdot 3 \cdot 7$
	13	$2^3 \cdot 13^4 \cdot 61$	9.723 109 461 555 451 065	$2^4 \cdot 3 \cdot 13$
2,1	-10	$2^9 \cdot 5^4$	7.550 149 267 298 211 191	$2^3 \cdot 3$
	-5	$2^3 \cdot 5^4 \cdot 83$	4.100 100 061 421 810 874	$2^4 \cdot 3$
2,7	-4	$2^7 \cdot 5 \cdot 7 \cdot 5521$	30.711 357 501 792 163 869	$2^4 \cdot 31$
	4	$2^7 \cdot 5^2 \cdot 7 \cdot 13^2$	30.241 242 163 364 995 702	$2^3 \cdot 3^2$
2,18	-2	$2^5 \cdot 3 \cdot 23 \cdot 4019$	45.366 598 887 078 149 119	2^7
	2	$2^6 \cdot 3 \cdot 19 \cdot 3863$	44.833 727 768 331 836 379	$2^6 \cdot 3$
3,4	-7	$2^4 \cdot 3 \cdot 7^4 \cdot 13^2$	16.942 378 596 607 902 527	$2^6 \cdot 3^2$
	7	$2^3 \cdot 3 \cdot 7^4 \cdot 659$	18.720 829 954 345 696 641	$2^8 \cdot 3$
5,6	3	$2^3 \cdot 3^5 \cdot 5^2 \cdot 251$	18.491 491 884 374 525 635	$2^5 \cdot 3^2$
	9	$2^3 \cdot 3^5 \cdot 5 \cdot 11 \cdot 239$	28.945 455 299 754 826 992	$2^7 \cdot 3$
1,8	-32	$2^7 \cdot 7 \cdot 213637$	52.204 279 460 991 961 715	$2^6 \cdot 76$
	-16	$2^6 \cdot 7 \cdot 56629$	42.425 014 230 077 161 533	$2^6 \cdot 5$
	-8	$2^6 \cdot 7^2 \cdot 2251$	33.698 572 880 074 345 953	$2^4 \cdot 7$
	-4	$2^6 \cdot 7 \cdot 4729$	26.073 066 596 592 118 065	$2^2 \cdot 11$
	-2	$2^8 \cdot 7 \cdot 1567$	19.593 835 688 524 596 875	$2^4 \cdot 5$
	2	$2^8 \cdot 7 \cdot 53$	14.813 899 994 225 298 034	2^2
	4	$2^5 \cdot 7 \cdot 1489$	24.048 110 493 694 347 697	2^3
	8	$2^6 \cdot 7 \cdot 9277$	32.627 369 278 809 858 915	2^6
	16	$2^5 \cdot 7 \cdot 43669$	41.848 416 409 554 195 162	2^7
	32	$2^8 \cdot 7 \cdot 29 \cdot 6473$	51.894 099 795 786 275 799	$2 \cdot 31 \cdot 59$
2,5	-5	$2^3 \cdot 3 \cdot 5^4 \cdot 59 \cdot 103$	24.612 611 743 927 750 340	$2^3 \cdot 3 \cdot 73$
	-4	$2^7 \cdot 3 \cdot 5^2 \cdot 197$	22.449 805 715 259 941 143	$2^4 \cdot 3$
	-3	$2^3 \cdot 3^5 \cdot 5 \cdot 751$	19.811 784 000 363 512 956	$2^6 \cdot 3$
	-2	$2^9 \cdot 3 \cdot 5 \cdot 127$	16.378 118 628 517 015 462	2^5
	2	$2^9 \cdot 3 \cdot 5 \cdot 71$	15.673 548 258 496 979 659	$2^3 \cdot 3$
	3	$2^3 \cdot 3^5 \cdot 5 \cdot 17 \cdot 31$	19.321 762 494 007 430 165	$2^3 \cdot 3 \cdot 5$
	4	$2^7 \cdot 3 \cdot 5 \cdot 761$	22.067 850 806 501 735 971	$2^3 \cdot 5$
	5	$2^3 \cdot 3 \cdot 5^4 \cdot 4957$	24.297 248 716 017 873 448	$2^5 \cdot 3^2 \cdot 5$

Table 6.3: Genus 2 curves defined by (6.20)

α_1, α_2	λ	Conductor	R_M	$L^*(C,0)/R_M$
$-1,2$	1	$3 \cdot 269$	2.778 805 941 907 237 297	$1/(2^3)$
$-2,3$	1	$5^2 \cdot 37$	16.262 221 873 282 600 595	$1/(2^3 \cdot 5)$
$-3,4$	1	$7 \cdot 109 \cdot 601$	31.203 152 023 145 321 601	$(2 \cdot 3)$
$-4,5$	1	$3 \cdot 594917$	44.050 864 811 285 204 483	$2 \cdot 3^3$
$-5,6$	1	$11 \cdot 3148477$	55.630 934 775 860 378 384	$2^2 \cdot 3 \cdot 23$
-6,7	1	$13 \cdot 17 \cdot 173099$	66.206 752 514 913 897 792	$2^4 \cdot 3^2 \cdot 7$
$-7,8$	1	$3 \cdot 5^2 \cdot 312841$	75.954 497 726 215 376 272	$2^4 \cdot 3^2$
$-9,10$	1	$19 \cdot 11887 \cdot 22051$	93.463 460 330 404 716 393	$2^4 \cdot 3^2 \cdot 157$
1,2	-5	$3^2 \cdot 5^4 \cdot 151$	5.424 369 542 333 873 857	$2^4 \cdot 5$
	-4	$2^5 \cdot 331$	4.707 981 109 579 949 333	1
	-3	$3^4 \cdot 19 \cdot 67$	3.920 459 199 544 282 782	$2^2 \cdot 3$
	-2	$2^5 \cdot 3^2 \cdot 67$	3.024 883 445 410 549 335	2^2
	-1	1123	1.933 194 467 367 533 231	$1/(2 \cdot 2)$
	1	$3^2 \cdot 101$	0.830 472 324 499 748 701	$1/2$
	2	$2^5 \cdot 389$	1.405 589 369 283 136 033	2^2
	3	$3^4 \cdot 631$	1.924 265 642 550 111 809	$2^3 \cdot 3$
	4	$2^5 \cdot 3^2 \cdot 13$	2.420 474 088 986 136 627	1
	5	$5^4 \cdot 17^2$	2.915 450 534 350 779 409	$2^3 \cdot 3$
1,4	-10	$2^5 \cdot 3^2 \cdot 5^4 \cdot 31 \cdot 61$	24.197 918 196 346 743 127	$2^7 \cdot 3 \cdot 23$
	-9	$3^3 \cdot 5^2 \cdot 197$	22.724 984 300 107 038 678	$(2^3)/3$
	-8	$2^7 \cdot 3 \cdot 43$	20.777 308 313 347 182 417	$1/2$
	-7	$3^2 \cdot 7^4 \cdot 1487$	18.840 899 436 962 049 611	$2^5 \cdot 3^3$
	-6	$2^5 \cdot 3^4 \cdot 47 \cdot 79$	17.074 796 494 018 855 926	$2^5 \cdot 3^2$
	-5	$3 \cdot 5^4 \cdot 9239$	15.299 879 777 289 147 441	$2^5 \cdot 13$
	-4	$2^5 \cdot 3^2 \cdot 5^2 \cdot 11$	13.441 694 552 825 870 861	2^2
	-3	$3^4 \cdot 97^2$	11.421 169 469 803 934 212	$2^2 \cdot 3^2$
	-2	$2^5 \cdot 3 \cdot 11 \cdot 353$	9.108 602 249 868 633 012	$2^3 \cdot 3$
	-1	$3^2 \cdot 1657$	6.190 412 208 778 898 565	1
	1	$3 \cdot 5^2 \cdot 163$	9.821 378 711 800 766 370	$1/2$
	2	$2^5 \cdot 3^2 \cdot 1721$	13.796 856 334 623 253 452	$2^3 \cdot 3$
	3	$3^4 \cdot 17729$	16.584 411 580 419 584 465	$2 \cdot 3 \cdot 7$
	4	$2^5 \cdot 3 \cdot 6571$	18.786 984 856 046 478 365	2^4
	5	$3^2 \cdot 5^4 \cdot 31 \cdot 43$	20.628 726 874 616 307 953	$2^6 \cdot 3$
	6	$2^5 \cdot 3^4 \cdot 5^2 \cdot 937$	22.221 606 845 384 142 055	$2^8 \cdot 7$
	7	$3 \cdot 7^4 \cdot 11 \cdot 5351$	23.630 749 093 514 787 832	$2^3 \cdot 3 \cdot 349$
	8	$2^7 \cdot 3^2 \cdot 3001$	24.897 812 118 312 038 230	$2^3 \cdot 11$
	9	$3^3 \cdot 11 \cdot 47 \cdot 167$	26.051 253 302 992 279 159	$3 \cdot 13$
	10	$2^5 \cdot 3 \cdot 5^4 \cdot 109 \cdot 467$	27.111 474 274 050 589 443	$2^5 \cdot 2029$
2,3	1	$5^2 \cdot 37$	2.439 333 280 992 390 089	$1/(2 \cdot 3)$
	3	$3^4 \cdot 569$	12.320 990 222 323 949 951	2
	5	$5^4 \cdot 17 \cdot 223$	16.916 677 554 622 824 304	2^6
	7	$7^4 \cdot 8741$	20.061 504 085 183 831 489	$2^5 \cdot 3 \cdot 5$
	9	$3^3 \cdot 17 \cdot 907$	22.522 079 759 749 421 493	3^2
	11	$5^2 \cdot 11^4 \cdot 953$	24.563 568 310 486 920 703	$2^9 \cdot 3 \cdot 5$
	13	$13^4 \cdot 29 \cdot 1171$	26.317 643 154 739 125 665	$2^5 \cdot 3 \cdot 5 \cdot 31$
	15	$3^4 \cdot 5^4 \cdot 45821$	27.860 721 411 144 107 821	$2^8 \cdot 157$

Table 6.4: Genus 2 curves defined by (6.21).

α_1, α_2,α_3	λ	Conductor	R_M	$L^*(C,0)/R_M$
-2,-1,4	1	$2^8 \cdot 3^2 \cdot 18371$	59.046 005 050 527 712 972	$2^5/5$
-1,1,2	1	$2^3 \cdot 3 \cdot 13441$	4.284 779 466 389 835 916	$2^2/5$
-1,1,5	-1	$2^7 \cdot 3 \cdot 79397$	27.669 699 094 992 761 987	$(2^3 \cdot 3^2)/5$
-1,1,6	1	$2^3 \cdot 5 \cdot 7 \cdot 744353$	31.090 391 952 751 320 926	2^6
-1,1,7	-1	$2^7 \cdot 3 \cdot 1072219$	40.602 032 884 481 091 737	$2^3 \cdot 13$
	1	$2^7 \cdot 3 \cdot 20509$	38.572 616 158 231 928 566	$(2^2 \cdot 3)/5$
-1,2,5	1	$2^3 \cdot 3^2 \cdot 348031$	71.582 436 419 327 843 261	$(2^3 \cdot 3)/5$
-1,3,4	1	$2^3 \cdot 5 \cdot 26053$	59.850 802 326 344 913 245	$1/5$
1,2,3	1	$2^3 \cdot 18233$	4.228 635 246 968 233 915	$2/5$
1,3,4	-1	$2^3 \cdot 3 \cdot 27919$	29.479 450 738 393 877 437	$2^2/(3 \cdot 5)$
	1	$2^3 \cdot 3 \cdot 53407$	44.395 593 452 934 245 913	$2/5$
2,3,4	-2	$2^{10} \cdot 20549$	68.321 025 364 121 294 593	2^2
	1	$2^7 \cdot 1823$	14.864 122 760 206 815 555	$2/(3 \cdot 5)$
	2	$2^{10} \cdot 7603$	63.163 189 588 039 386 444	2
3,4,5	1	$2^3 \cdot 3 \cdot 47431$	69.422 587 877 081 519 967	$1/(3 \cdot 5)$
	2	2^{11}	119.496 359 208 818 343 264	$2^2 \cdot 3^2$
4,5,6	-1	$2^7 \cdot 105503$	116.337 042 385 238 654 068	$(2 \cdot 3)/5$

Table 6.5: Genus 3 curves defined by (6.21).

α_1,α_2,α_3,α_4	λ	Conductor	R_M	$L^*(C,0)/R_M$
-3,-2,-1,1	-1	$2^6 \cdot 3 \cdot 548489$	36.695 124 546 980 647 767	2/3
	1	$2^6 \cdot 3 \cdot 31 \cdot 3911$	49.552 479 563 630 303 195	$1/3^2$
-1,1,2,4	1	$2^8 \cdot 3^2 \cdot 5 \cdot 100297$	90.187 009 232 834 664 343	2^2
1,2,3,4	1	$2^8 \cdot 3 \cdot 137849$	94.258 408 935 953 995 643	1/3

Table 6.6: Genus 4 curves defined by (6.21).

Non-hyperelliptic examples

Finally, we want to calculate our regulator R_M for some non-hyperelliptic curves defined over the rationals. Assume that the affine curve $f(x,y) = 0$ given by a polynomial of the form (6.19) is non-singular. Then, by [23, Proposition 5.1.], C is non-hyperelliptic unless $N_2 = N_3 = 1$ or $N_2 = 2$, $N_3 = 0$. Even if the affine model is singular, we can check for this property with the MAGMA function IsGeometricallyHyperelliptic.

In the following we considered non-hyperelliptic examples with $N_3 = 0$, i.e. curves defined via equation (6.19) by integers $\alpha_1, \ldots, \alpha_{N_1}$, $\beta_1, \ldots, \beta_{N_2}$ such that $N_1 \geq N_2 \geq 3$ and $0 \neq \lambda \in \mathbb{Z}$. By Proposition 6.4.7 their construction yields exactly $(N_1-1)(N_2-1) \geq g$ elements of R-type (and none of T-type) in this case. All the lines $L_{i,j}$ are of the form $x+\alpha_i$ or $y+\beta_j$. It is easy to see that three of such lines will never meet in an affine point. Therefore, [23, Assumption 6.1.] will always hold true for such curves.

We computed the regulator R_M for many non-hyperelliptic examples of genus $4, 6, 8, 9, 10$ and 12. We also found a genus 5 curve whose affine model has singularities over an ex-

tension field, presented in Example 6.4.10. We are not aware that anyone has computed regulators for finite index subgroups of $K_2(C,\mathbb{Z})$ for non-hyperelliptic curves before, so we presume it is worth listing some of these values. We computed the regulator R_M for many more curves than listed in the tables below. For all examples that we considered the matrix P_M had rank g, thus providing numerical evidence for $\mathrm{rk}(K_2^T(C,\mathbb{Z})) \geq g$.

Note that with our algorithm it is possible to compute these regulators to hundreds of digits of precision for curves defined by polynomials (6.19), even for high genus. If anyone managed to calculate the L-series for these curves, then the second part of Beilinson's conjecture could be checked using the values for R_M that are listed in the tables below.

Example 6.4.10. We look at the curve defined by $[\alpha_i] = [\beta_j] = [-1,0,1,2]$ (i.e. $N = 2$, $N_1 = N_2 = 4$) and $\lambda = 1$ with equation

$$\begin{aligned} &x^4y^4 + 2x^4y^3 - x^4y^2 - 2x^4y + 2x^3y^4 + 4x^3y^3 - 2x^3y^2 \\ -&4x^3y - x^2y^4 - 2x^2y^3 + x^2y^2 + 2x^2y - 2xy^4 - 4xy^3 + \\ &2xy^2 + 4xy - 1 = 0\,. \end{aligned}$$

The affine curve has 4 singularities over the splitting field, causing the genus of C to be only 5 (instead of 9). Nonetheless, by (6.18) and Theorem 6.4.8, we have 9 integral elements $M = \{M_1, \ldots, M_9\}$ that have no a priori reason to be linearly dependent. Our strategy yields a real (9×10)-matrix P_M that has rank 5 with regulator

$$R_M = 0.615\ 705\ 316\ 185\ 336\ 649\ 171\ 137\ 603\ 021.$$

Moreover, for some 6-subsets of M we could find non-trivial integer relations between these elements, e.g.

$$\begin{aligned} 4M_1 + 4M_2 - 4M_3 - M_4 - M_5 + M_6 &= 0, \\ -5M_4 - 5M_5 + 5M_6 + 4M_7 + 4M_8 - 4M_9 &= 0\,. \end{aligned}$$

We can interpret this as quite strong numerical evidence for $\mathrm{rk}(K_2^T(C,\mathbb{Z})) = g$. We found this relations by applying the MAGMA function LinearRelation to the entries of the columns of P_M. This function uses the LLL-algorithm to find small integer relations among complex numbers. Presumably there exist non-trivial integer relations for each 6-subset of M, but the coefficients are simply too large to be found by LinearRelation.

$[\alpha_i], [\beta_j]$	λ	R_M
[-4,-2,0],[-4,0,2]	1	456.043 453 241 054 052 627 045 933 832 051
	2	686.614 524 761 928 801 940 835 676 020 924
[-3,-2,0],[-3,0,2]	1	162.166 019 922 041 952 049 801 549 762 265
	2	274.232 839 790 298 730 680 423 444 638 905
[-2,-1,3],[-3,0,1]	1	177.385 266 412 147 500 204 260 528 294 340
	2	297.006 844 888 622 010 599 783 751 382 951
	4	469.882 837 274 022 892 240 597 753 938 858
	8	709.411 676 766 294 538 656 490 193 638 722
	16	1030.070 664 892 505 319 732 787 353 063 345
[-1,0,1],[-4,-1,0]	1	14.105 020 302 991 836 538 780 051 244 741
	2	34.239 334 425 022 841 309 967 231 686 303
	3	59.363 162 473 450 196 675 182 887 969 578
	4	84.281 001 244 876 601 137 489 759 265 808
	5	105.829 488 765 660 987 721 148 627 610 684
[-1,0,1],[-3,0,3]	1	109.145 596 573 853 184 332 035 572 949 048
	2	194.654 149 851 681 309 337 281 397 066 682
	3	263.022 945 589 933 684 258 257 573 404 295
	4	321.279 593 575 536 520 747 136 708 550 448
	5	372.657 827 585 399 389 368 061 099 542 381
[-1,0,1],[-2,0,2]	1	27.568 933 663 287 109 898 947 413 928 443
	2	65.510 174 737 954 604 127 379 595 103 916
	3	97.943 840 814 314 839 597 101 327 618 481
	4	126.878 531 440 319 005 109 856 423 267 449
	5	153.216 783 264 459 658 208 641 451 871 831
[-1,1,2],[-1,1,2]	1	22.967 554 622 887 647 714 671 626 562 318
	2	53.796 479 624 141 630 928 818 555 990 170
	3	92.091 559 754 146 752 632 772 632 732 685
	4	123.990 119 702 584 843 261 070 766 578 114
	5	152.197 660 265 409 904 192 521 423 760 204
[1,2,3],[4,5,6]	1	1.803 127 938 237 073 335 476 302 671 407
	2	2.634 355 647 799 092 544 726 667 458 482
	3	4.967 144 449 687 612 637 211 190 308 940
	4	7.865 090 162 994 230 867 483 085 059 968
	5	11.378 328 271 642 185 488 723 527 021 752
[2,3,4],[-5,-6,7]	1	159.868 104 615 551 235 777 722 637 422 493
	3	394.750 000 897 801 638 146 120 227 034 355
	9	785.982 345 617 705 281 073 198 940 706 635
	27	1 397.164 947 531 995 768 140 847 597 188 591
	81	2 296.488 344 976 166 236 258 434 729 029 851

Table 6.7: Non-hyperelliptic genus 4 curves defined by (6.19) .

$[\alpha_i], [\beta_i]$	λ	R_M
[-1,0,1,2],[-1,0,1]	1	13.494 821 399 236 645 020 489 062 860 619
	2	68.723 773 726 907 208 463 526 327 207 324
	3	203.392 887 640 995 088 513 731 611 976 856
	6	885.329 912 371 348 213 445 938 011 240 912
	12	2 544.140 948 799 804 312 508 214 947 394 976
	24	6 038.220 951 448 499 268 873 866 088 509 977
[-2,-1,0,2],[-1,1,2]	1	858.678 576 717 381 620 780 180 541 806 104
	2	2 769.376 586 171 122 869 829 573 857 541 085
	3	4 828.364 321 545 602 727 415 702 870 369 030
[-1,0,1,2],[-1,0,2]	1	152.534 884 806 105 013 742 133 860 910 004
	2	771.706 642 266 164 757 306 003 664 647 763
	3	1 544.643 314 765 958 640 512 577 158 422 883
[1,2,3,-4],[-8,5,6]	1	96 340.683 768 001 185 229 006 616 338 045 410
	2	159 956.021 088 862 400 873 220 851 361 999 658
	3	210 532.800 621 481 026 081 485 806 475 436 788
[-4,-2,0,2],[-2,0,3]	1	52 742.285 990 980 794 564 518 687 461 651 358
	2	88 978.921 771 010 165 226 366 765 673 628 608
	3	118 386.410 453 223 882 490 044 668 886 436 354

Table 6.8: Non-hyperelliptic genus 6 curves defined by (6.19) .

Genus	$[\alpha_i], [\beta_j]$	λ	R_M
8	[-2,-1,0,1,2],[-1,0,1]	1	420.552 706 742 028 906 935 252 086 427 897
8	[-2,-1,0,1,2],[-1,1,2]	1	124 62.203 079 931 265 848 605 791 030 383 288
8	[-1,0,1,2,3],[-1,0,2]	1	6 231.101 539 965 632 924 302 895 515 191 644
9	[-3,-1,0,1],[-3,-1,0,1]	-1	293 779.893 543 194 598 332 827 005 807 430 454
9	[-1,1,2,3],[-1,0,1,2]	1	36 576.138 789 587 471 029 181 088 460 501 378
9	[-2,-1,0,1],[-2,-1,1,2]	1	103 649.865 291 009 800 031 652 721 292 896 122
10	[-2,-1,0,1,2,3],[1,2,3]	1	213 154.270 820 449 438 832 553 362 048 251 339
10	[-2,-1,0,1,2,3],[-2,-1,1]	-1	2 043 703.158 618 027 854 414 594 268 129 904 226
10	[-2,-1,0,1,2,4],[-3,-1,0]	1	6 205 197.565 327 708 112 933 975 003 989 959 505
12	[-1,0,1,2,3],[-1,0,1,2]	1	1 203 434.305 816 573 775 831 991 074 232 924 397
12	[-2,-1,0,1,2],[-2,-1,1,2]	1	37 363 335.725 710 046 191 310 784 290 901 896 391

Table 6.9: Non-hyperelliptic curves defined by (6.19) with genus > 6 .

Appendices

Appendix A

Miscellaneous

A.1 Closest point on an ellipse

For an ellipse ε_r parametrized by $r > 0$ we use

$$\varepsilon_r = \{\, r\cos(t) + i\sqrt{r^2-1}\sin(t) \;\mid\; 0 \le t < \pi \,\}$$

to compute a point on the ellipse that realizes the distance to a point $x \notin \varepsilon_r$ see Algorithm A.1.1 below. In particular, we apply Newton's method (§1.3) to the function

$$s(t) = \cos(t)\sin(t) - r\,\mathrm{Re}(x)\sin(t) + \sqrt{r^2-1}\,\mathrm{Im}(x)\cos(t)\,.$$

Since the ellipse is convex, the solution is unique on the quadrant containing x and Newton's method converges quadratically.

Algorithm A.1.1 (Closest point on ellipse)**.** For $x \in \mathbb{C}$ outside of ε_r, approximate $z \in \varepsilon_r$ up to an error $\delta > 0$ such that $|x - z| = \mathrm{dist}(x, \varepsilon_r)$.

(1) Set $s_r \leftarrow \mathrm{sgn}(\mathrm{Re}(x))$, $s_i \leftarrow \mathrm{sgn}(\mathrm{Im}(x))$, $r' \leftarrow \sqrt{r^2-1}$.

(2) If $s_i = 0$, then return $s_r \cdot r$.

(3) If $s_r = 0$, then return $s_i \cdot ir'$.

(4) Set $\tilde{x} \leftarrow |\mathrm{Re}(x)| + i\,|\mathrm{Im}(x)|$.

(5) Set $\tilde{t} \leftarrow \cos^{-1}(\tilde{x})$.

(6) Repeat

  (6.1) Newton-step: Set $t \leftarrow \tilde{t}$ and $\tilde{t} \leftarrow s(t)/s'(t)$.

(6) until $\left|t - \tilde{t}\right| < \delta$.

(7) $\tilde{x} \leftarrow r\cos(\tilde{t}) + ir'\sin(\tilde{t})$.

(8) Return $s_r \cdot \mathrm{Re}(\tilde{x}) + s_i \cdot i\,\mathrm{Im}(\tilde{x})$.

A.2 Source code for the Tretkoff algorithm

```
////////////////////////////////////////////////////////////////
/// ***** Magma implementation of the Tretkoff algorithm ***** ///
/// ************ Code written by Christian Neurohr *********** ///
////////////////////////////////////////////////////////////////

// Define suitable edge class for edges: RSEdge
declare type RSEdge;
declare attributes RSEdge: StPt, EndPt, Branch, Level, Terminated, Position, PQ;

// Constructor
function RS_Edge( Edge : Branch := [Edge[1],Edge[2]], Level := 0, Terminated :=
    false )
// Creates Edge
   e := New(RSEdge);
   e`StPt := Edge[1];
   e`EndPt := Edge[2];
   e`Branch := Branch;
   e`Level := Level;
   e`Terminated := Terminated;
   if e`StPt lt e`EndPt then
      e`PQ := true;
   else
      e`PQ := false;
   end if;
   return e;
end function;

// Printing
intrinsic Print( e::RSEdge )
{ Print Edge }
   if e`Terminated then
      print "Edge from",e`StPt,"to",e`EndPt,"on Level",e`Level,"and Terminated
          Branch",e`Branch;
   else
      print "Edge from",e`StPt,"to",e`EndPt,"on Level",e`Level,"and
          Branch",e`Branch;
   end if;
end intrinsic;

// Define "=" on RSEdges
intrinsic 'eq'(Edge1::RSEdge, Edge2::RSEdge) -> BoolElt
{ Returns true if Edge1 = Edge2, false otherwise }
   if (Edge1`StPt ne Edge2`StPt) or (Edge1`EndPt ne Edge2`EndPt) then
      return false;
   else
      return true;
   end if;
end intrinsic;

function RS_ReverseEdge(Edge)
// Returns the edge with swapped start- and end point
   return RS_Edge(<Edge`EndPt,Edge`StPt>);
end function;
// End RSEdge
```

```
function RS_CycleToPerm(Tau,m)
// Creates a permutation element from the indexed set Tau
   Perm := [1..m];
   t := #Tau;
   for j in [1..t] do
      Perm[Tau[j]] := Tau[j mod t + 1];
   end for;
   return Sym(m)!Perm;
end function;

// Functions for sorting

function RS_SortTermEdges(EdgesOnPartLevel, StOrEnd)
// Sorts the terminated edges in step 3 and 5
// If StOrEnd = -1 sort by increasing start point; If StOrEnd = 1 sort by
    increasing end point
   assert StOrEnd in [-1,1];
   function RS_CompareEdges( Edge1, Edge2 )
      if StOrEnd eq -1 then
         e1 := Edge1`StPt; e2 := Edge2`StPt;
      else
         e1 := Edge1`EndPt; e2 := Edge2`EndPt;
      end if;
      if e1 lt e2 then
         return -1;
      elif e1 eq e2 then
         if Edge1`Position lt Edge2`Position then
            return -1;
         elif Edge1`Position eq Edge2`Position then
            return 0;
         else
            return 1;
         end if;
      else
         return 1;
      end if;
   end function;
   return Sort(EdgesOnPartLevel,RS_CompareEdges);
end function;

function RS_SortByBranch( EdgesOnLevel, TerminatedEdges )
// Returs the list of terminated edges sorted with RS_CompareBranches
   function RS_CompareBranches(Edge1,Edge2)
      // Edge1 < Edge2 iff Edge1 appears earlier while going counter-clockwise
          through the Tretkoff tree
      // Edge2 > Edge2 iff Edge2 ---- "" ----
      if Edge1`Level eq Edge2`Level then
         if Edge1`Position lt Edge2`Position then
            return -1;
         elif Edge1`Position eq Edge2`Position then
            return 0;
         else
            return 1;
         end if;
      elif Edge1`Level lt Edge2`Level then
         PredEdge2 :=
             RS_Edge(<Edge2`Branch[Edge1`Level],Edge2`Branch[Edge1`Level+1]>);
```

```
        Ind := Position(EdgesOnLevel[Edge1`Level],PredEdge2);
        return RS_CompareBranches(Edge1,EdgesOnLevel[Edge1`Level][Ind]);
      else
        return (-1)*RS_CompareBranches(Edge2,Edge1);
      end if;
   end function;
   return Sort(TerminatedEdges,RS_CompareBranches);
end function;

// Start Tretkoff algorithm

declare verbose Tretkoff,2; // Printing

intrinsic RS_Tretkoff( LocalMonodromy::SeqEnum[GrpPermElt] : Genus := -1 ) ->
    SeqEnum[SeqEnum[RngIntElt]], Mtrx
{ Compute a basis of the first homology group of a connected compact Riemann
    surface from a monodromy representation }

   d := #LocalMonodromy;
   if d eq 0 then
      error Error("Input cannot be an empty list.");
   else
      Sym := Universe(LocalMonodromy);
      m := Degree(Sym);
   end if;

   // Checking connectivity of the Riemann surface
   S := PermutationGroup< m | [ s : s in LocalMonodromy ] >;
   assert IsTransitive(S);

   // Decompose the local monodromies into disjoint cycles
   RamificationPoints := []; RamificationIndices := [];
   for j in [1..d] do
      CycleDecomp := CycleDecomposition(LocalMonodromy[j]);
      for Tau in CycleDecomp do
         if #Tau gt 1 then
            Append(~RamificationPoints,<j,Tau,RS_CycleToPerm(Tau,m)>);
            Append(~RamificationIndices,#Tau-1);
         end if;
      end for;
   end for;

   vprint Tretkoff,1 : "Ramification points:",RamificationPoints;
   vprint Tretkoff,1 : "Ramification indices:",RamificationIndices;

   // Checking Riemann-Hurwitz formula
   GenusViaMonodromy := 1/2 * &+RamificationIndices + 1 - m;
   if Genus ne -1 then
      g := Genus;
      assert g eq GenusViaMonodromy;
   else
      Ok, g := IsCoercible(Integers(),GenusViaMonodromy);
      if not Ok or g lt 0 then
         error Error("Riemann-Hurwitz formula violated.");
      end if;
   end if;

   // Dealing with genus 0 curves
```

```
if g eq 0 then
   error Error("Genus of the Riemann surface is zero; homology group is
       trivial.");
else
   vprint Tretkoff,1 : "Genus of the Riemann surface: ",g;
end if;

// Initializing Tretkoff tree
// Vertices 1,...,t correspond to ramification points
// Vertices t+1,...,t+m correspond to sheets
t := #RamificationPoints;
Vertices := {t+1};
EdgesOnLevel := [[]];
TerminatedEdges := [];

// Condition to terminate the algorithm
AllBranchesTerminated := false;

// Step 1:
// Initializing graph on level 1
Level := 1;
vprint Tretkoff, 1: "Initializing graph-level",Level;
for j in [1..t] do
   if 1 in RamificationPoints[j][2] then
      e := RS_Edge(<t+1,j>: Level := Level, Branch := [t+1,j]);
      Include(~Vertices,j);
      Append(~EdgesOnLevel[Level],e);
   end if;
end for;

repeat
   // Step 2: Continuing open branches to sheets
   Level +:= 1;
   vprint Tretkoff, 1: "Constructing graph-level",Level;
   s := 0;
   EdgesOnLevel[Level] := [];
   for Edge in EdgesOnLevel[Level-1] do
      if not Edge`Terminated then
         Perm := RamificationPoints[Edge`EndPt][3];
         l := Edge`StPt - t;
         while Perm ne Id(Sym) do
            k := t + l^Perm;
            if not k in Edge`Branch then
               NewEdge := RS_Edge(<Edge`EndPt,k> : Level := Level, Branch :=
                   Append(Edge`Branch,k), Terminated := false);
               s +:= 1;
               NewEdge`Position := s;
               Append(~EdgesOnLevel[Level],NewEdge);
            end if;
            Perm *:= RamificationPoints[Edge`EndPt][3];
         end while;
      end if;
   end for;

   // Step 3: Terminate branches at sheets
   vprint Tretkoff, 1: "Terminate branches at graph-level",Level;
   SortedEdgesOnLevel := RS_SortTermEdges(EdgesOnLevel[Level],-1);
   for j in [1..#SortedEdgesOnLevel] do
```

```
      Edge := SortedEdgesOnLevel[j];
      if Edge‘EndPt in Vertices then
         Edge‘Terminated := true;
         Append(~TerminatedEdges,Edge);
      else
         Include(~Vertices,Edge‘EndPt);
      end if;
   end for;

   // Step 4:
   // Continuing open branches to ramification points
   Level +:= 1;
   vprint Tretkoff, 1: "Constructing graph-level",Level;
   s := 0;
   EdgesOnLevel[Level] := [];
   for Edge in EdgesOnLevel[Level-1] do
      if not Edge‘Terminated then
         l := Edge‘EndPt - t;
         k := Edge‘StPt mod t + 1;
         for j in [1..t] do
            if l in RamificationPoints[k][2] and not k in Edge‘Branch then
               NewEdge := RS_Edge(<Edge‘EndPt,k> : Level := Level, Branch :=
                   Append(Edge‘Branch,k), Terminated := false);
               s +:= 1;
               NewEdge‘Position := s;
               Append(~EdgesOnLevel[Level],NewEdge);
            end if;
            k := k mod t + 1;
         end for;
      end if;
   end for;

   // Step 5:
   // Terminate branches at ramification points
   vprint Tretkoff, 1: "Terminate branches at graph-level",Level;
   SortedEdgesOnLevel := RS_SortTermEdges(EdgesOnLevel[Level],1);
   for j in [1..#SortedEdgesOnLevel] do
      Edge := SortedEdgesOnLevel[j];
      if Edge‘EndPt in Vertices then
         Edge‘Terminated := true;
         Append(~TerminatedEdges,Edge);
      else
         Include(~Vertices,Edge‘EndPt);
      end if;
   end for;

   // Step 6:
   // Check whether all branches are terminated
   AllBranchesTerminated := true;
   for Edge in EdgesOnLevel[Level] do
      if not Edge‘Terminated then
         AllBranchesTerminated := false;
         break;
      end if;
   end for;
   vprint Tretkoff, 1: "All branches terminated?",AllBranchesTerminated;
until AllBranchesTerminated;
```

```
// Cut off empty entries in list
if #EdgesOnLevel[Level] eq 0 then
   Prune(~EdgesOnLevel);
end if;

// Checking correct number of terminated edges
CTE := 4*g + 2*m - 2;
CTE2 := Round(CTE/2);
assert CTE eq #TerminatedEdges;

vprint Tretkoff,2 : "EdgesOnLevel:",EdgesOnLevel;
vprint Tretkoff,2 : "#TerminatedEdges:",#TerminatedEdges;
vprint Tretkoff,2 : "4*Genus + 2*N - 2: ",CTE;

// Order terminated edges according to RS_SortByBranch
OrdTermEdges := RS_SortByBranch(EdgesOnLevel,TerminatedEdges);

// Need to go through ordered terminated edges clockwise (instead of
    counter-clockwise)
Reverse(~OrdTermEdges);
vprint Tretkoff,2 : "Ordered terminated edges:",OrdTermEdges;

// Divide terminated edges into lists P (even level) and Q (odd level)
    depending on their level
vprint Tretkoff,1 : "Labeling terminated edges ... ";
P := []; QQ := []; Q := []; l := 1;
for k in [1..CTE] do
   Edge := OrdTermEdges[k];
   Edge`Position := [k];
   if Edge`PQ then
      Append(~P,Edge);
      Append(~Edge`Position,l);
      l +:= 1;
   else
      Append(~QQ,Edge);
   end if;
end for;
vprint Tretkoff,2 : "P:", P;
for k in [1..CTE2] do
   Edge := QQ[k];
   l := Position(P,RS_ReverseEdge(Edge));
   Append(~Edge`Position,l);
   Q[l] := Edge;
end for;
vprint Tretkoff,2 : "Q:", Q;

// Piece together the 2g+m-1 cycles
vprint Tretkoff,1 : "Constructing cycles ... ";
Cycles := [];
for j in [1..CTE2] do
   Cycle := P[j]`Branch cat Reverse(Prune(Prune(Q[j]`Branch)));
   k := 1;
   while k le #Cycle-1 do
      Cycle[k] -:= t;
      Cycle[k+1] := RamificationPoints[Cycle[k+1]][1];
      k +:= 2;
   end while;
   Cycle[k] -:= t;
```

```
      Append(~Cycles,Cycle);
   end for;
   vprint Tretkoff,2 : "Cycles: ", Cycles;

   // Compute the intersection pairing of the cycles and store them in a matrix K
   vprint Tretkoff,1 : "Computing intersection matrix ... ";
   K := ZeroMatrix(Integers(),CTE2,CTE2);
   for j in [1..CTE2] do
      k := P[j]‘Position[1] mod CTE + 1;
      while true do
         NextEdge := OrdTermEdges[k];
         if NextEdge‘PQ then
            K[NextEdge‘Position[2]][j] +:= 1;
         else
            if NextEdge‘Position[2] eq j then
               break;
            else
               K[NextEdge‘Position[2]][j] -:= 1;
            end if;
         end if;
         k := k mod CTE + 1;
      end while;
   end for;

   // Checking rank of intersection matrix
   rk := Rank(K); assert rk eq 2*g;
   vprint Tretkoff,2 : "Intersection matrix:",K;
   vprint Tretkoff,2 : "Rank of intersection matrix: ",rk;

   // Return cycles and intersection matrix
   return Cycles, K;
end intrinsic;
```

Appendix B

Declaration

Hereby, I declare, that the presented dissertation with the title

Efficient integration on Riemann surfaces & applications

is my own work and, that I have not used other sources than the sources stated in the text or in the bibliography. The dissertation has, neither as a whole, nor in part, been submitted for assessment in a doctoral procedure at another university.

I confirm that I am aware of the guidelines of good scientific practice of the Carl von Ossietzky University Oldenburg and that I observed them. Furthermore, I declare that I have not availed myself of any commercial placement or consulting services in connection with my doctoral procedure.

Oldenburg, 22.05.2018

Christian Neurohr

Bibliography

[1] Milton Abramowitz and Irene A. Stegun. *Handbook of mathematical functions with formulas, graphs, and mathematical tables*, volume 55 of *National Bureau of Standards Applied Mathematics Series*. For sale by the Superintendent of Documents, U.S. Government Printing Office, Washington, D.C., 1964.

[2] David H Bailey. Tanh-sinh high-precision quadrature. *Online paper* `http://www.davidhbailey.com/dhbpapers/dhb-tanh-sinh.pdf`, 2006.

[3] David H Bailey, Karthik Jeyabalan, and Xiaoye S Li. A comparison of three high-precision quadrature schemes. *Experimental Mathematics*, 14(3):317–329, 2005.

[4] Barinder Singh Banwait, John Cremona, et al. Tetrahedral elliptic curves and the local-to-global principle for isogenies. *Algebra and Number Theory*, 8(5):1201–29, 2014.

[5] Burcu Baran. An exceptional isomorphism between modular curves of level 13. *Journal of Number Theory*, 145:273–300, 2014.

[6] Leo Bluestein. A linear filtering approach to the computation of discrete Fourier transform. *IEEE Transactions on Audio and Electroacoustics*, 18(4):451–455, 1970.

[7] Alexander I Bobenko. Introduction to compact Riemann surfaces. In *Computational Approach to Riemann Surfaces*, pages 3–64. Springer, 2011.

[8] Ignace Bogaert. Iteration-free computation of Gauss-Legendre quadrature nodes and weights. *SIAM Journal on Scientific Computing*, 36(3):A1008–A1026, 2014.

[9] Wieb Bosma, John Cannon, and Catherine Playoust. The MAGMA algebra system. I. The user language. *J. Symbolic Comput.*, 24(3-4):235–265, 1997. Computational algebra and number theory (London, 1993).

[10] Jean-Benoît Bost. Introduction to compact Riemann surfaces, Jacobians, and abelian varieties. In *From number theory to physics (Les Houches, 1989)*, pages 64–211. Springer, Berlin, 1992.

[11] Jean-Benoît Bost and Jean-François Mestre. Moyenne arithmético-géométrique et périodes des courbes de genre 1 et 2. *Gaz. Math.*, (38):36–64, 1988.

[12] Richard P. Brent and Paul Zimmermann. *Modern computer arithmetic*, volume 18 of *Cambridge Monographs on Applied and Computational Mathematics*. Cambridge University Press, Cambridge, 2011.

[13] Egbert Brieskorn and Horst Knörrer. *Plane algebraic curves*. Modern Birkhäuser Classics. Birkhäuser/Springer Basel AG, Basel, 1986. Translated from the German original by John Stillwell, [2012] reprint of the 1986 edition.

[14] Vincenz Busch. *Beilinson's Conjectures for Superelliptic Curves*. PhD thesis, Universität Hamburg, 2015.

[15] M. M. Chawla and M. K. Jain. Error estimates for Gauss quadrature formulas for analytic functions. *Math. Comp.*, 22:82–90, 1968.

[16] Alexey Chernov and Christoph Schwab. Exponential convergence of Gauss-Jacobi quadratures for singular integrals over simplices in arbitrary dimension. *SIAM Journal on Numerical Analysis*, 50(3):1433–1455, 2012.

[17] David V Chudnovsky and Gregory V Chudnovsky. On expansion of algebraic functions in power and Puiseux series, I. *Journal of Complexity*, 2(4):271–294, 1986.

[18] David V Chudnovsky and Gregory V Chudnovsky. On expansion of algebraic functions in power and Puiseux series, II. *Journal of Complexity*, 3(1):1–25, 1987.

[19] Henri Cohen. *Advanced topics in computational number theory*, volume 193 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, 2000.

[20] James W Cooley and John W Tukey. An algorithm for the machine calculation of complex Fourier series. *Mathematics of computation*, 19(90):297–301, 1965.

[21] Edgar Costa, Nicolas Mascot, Jeroen Sijsling, and John Voight. Rigorous computation of the endomorphism ring of a Jacobian. *Preprint arXiv:1705.09248*, 2017.

[22] John E. Cremona and Thotsaphon Thongjunthug. The complex AGM, periods of elliptic curves over $\mathbb{C}$ and complex elliptic logarithms. *J. Number Theory*, 133(8):2813–2841, 2013.

[23] Rob de Jeu and Hang Liu. On K_2 of certain families of curves. *Preprint arXiv:1402.4822*, 2014.

[24] B. Deconinck and M. Patterson. Computing the Abel map. *Physica D*, 237:3214–3232, 2008.

[25] Bernard Deconinck, Matthias Heil, Alexander Bobenko, Mark Van Hoeij, and Marcus Schmies. Computing Riemann Theta functions. *Mathematics of Computation*, 73(247):1417–1442, 2004.

[26] Bernard Deconinck, Matthew S Patterson, and Christopher Swierczewski. Computing the Riemann constant vector. *Preprint* `https://depts.washington.edu/bdecon/papers/pdfs/rcv.pdf`, 2015.

[27] Bernard Deconinck and Mark van Hoeij. Computing Riemann matrices of algebraic curves. *Phys. D*, 152/153:28–46, 2001. Advances in nonlinear mathematics and science.

[28] Tim Dokchitser. Computing special values of motivic L-functions. *Experimental Mathematics*, 13(2):137–149, 2004.

[29] Tim Dokchitser, Rob De Jeu, and Don Zagier. Numerical verification of Beilinson's conjecture for K_2 of hyperelliptic curves. *Compositio Mathematica*, 142(2):339–373, 2006.

[30] Pierre Duhamel and Martin Vetterli. Fast Fourier transforms: a tutorial review and a state of the art. *Signal processing*, 19(4):259–299, 1990.

[31] Dominique Duval. Rational Puiseux expansions. *Compositio mathematica*, 70(2):119–154, 1989.

[32] J Eilbeck, Keno Eilers, and V Enolski. Periods of second kind differentials of (n,s)-curves. *Transactions of the Moscow Mathematical Society*, 74:245–260, 2013.

[33] Hershel M Farkas and Irwin Kra. Riemann surfaces. In *Riemann surfaces*, pages 9–31. Springer, 1992.

[34] E Flynn, Franck Leprévost, Edward Schaefer, William Stein, Michael Stoll, and Joseph Wetherell. Empirical evidence for the Birch and Swinnerton-Dyer conjectures for modular Jacobians of genus 2 curves. *Mathematics of Computation*, 70(236):1675–1697, 2001.

[35] Jörg Frauendiener and Christian Klein. Algebraic curves and Riemann surfaces in MATLAB . In *Computational approach to Riemann surfaces*, pages 125–162. Springer, 2011.

[36] Jörg Frauendiener and Christian Klein. Computational approach to hyperelliptic Riemann surfaces. *Letters in Mathematical Physics*, 105(3):379–400, 2015.

[37] Jörg Frauendiener and Christian Klein. Computational approach to compact Riemann surfaces. *Nonlinearity*, 30(1):138–172, 2017.

[38] Georg Frobenius. Theorie der linearen formen mit ganzen coefficienten. *Journal für die reine und angewandte Mathematik*, 86:146–208, 1879.

[39] Andreas Glaser, Xiangtao Liu, and Vladimir Rokhlin. A fast algorithm for the calculation of the roots of special functions. *SIAM Journal on Scientific Computing*, 29(4):1420–1438, 2007.

[40] Irving John Good. The interaction algorithm and practical Fourier analysis. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 361–372, 1958.

[41] Xavier Gourdon. *Combinatoire, algorithmique et géométrie des polynômes*. PhD thesis, 1997.

[42] Nicholas Hale and Alex Townsend. Fast and accurate computation of Gauss-Legendre and Gauss-Jacobi quadrature nodes and weights. *SIAM Journal on Scientific Computing*, 35(2):A652–A674, 2013.

[43] Robin Hartshorne. *Algebraic geometry*. Springer-Verlag, New York-Heidelberg, 1977. Graduate Texts in Mathematics, No. 52.

[44] David Harvey, Maike Massierer, and Andrew V Sutherland. Computing L-series of geometrically hyperelliptic curves of genus three. *LMS Journal of Computation and Mathematics*, 19(A):220–234, 2016.

[45] Florian Hess. Computing Riemann-Roch spaces in algebraic function fields and related topics. *Journal of Symbolic Computation*, 33(4):425–445, 2002.

[46] Marc Hindry and Joseph H. Silverman. *Diophantine geometry*, volume 201 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, 2000. An introduction.

[47] F. Johansson. Arb: a C library for ball arithmetic. *ACM Communications in Computer Algebra*, 47(4):166–169, 2013.

[48] F. Johansson. Numerical integration in arbitrary-precision ball arithmetic. *Preprint arXiv:1802.07942*, February 2018.

[49] F. Johansson and M. Mezzarobba. Fast and rigorous arbitrary-precision computation of Gauss-Legendre quadrature nodes and weights. *Preprint arXiv:1802.03948*, February 2018.

[50] Kiran S Kedlaya and Andrew V Sutherland. Computing L-series of hyperelliptic curves. In *International Algorithmic Number Theory Symposium*, pages 312–326. Springer, 2008.

[51] Pinar Kilicer, Hugo Labrande, Reynald Lercier, Christophe Ritzenthaler, Jeroen Sijsling, and Marco Streng. Plane quartics over $\mathbb{Q}$ with complex multiplication. *Preprint arXiv:1701.06489*, 2017.

[52] Ja Kyung Koo. On holomorphic differentials of some algebraic function field of one variable over $\mathbb{C}$. *Bulletin of the Australian Mathematical Society*, 43(3):399–405, 1991.

[53] Stefan Kranich. An epsilon-delta bound for plane algebraic curves and its use for certified homotopy continuation of systems of plane algebraic curves. *Preprint arXiv:1505.03432*, 2015.

[54] HT Kung and Joseph Frederick Traub. All algebraic functions can be computed fast. *Journal of the ACM (JACM)*, 25(2):245–260, 1978.

[55] Hugo Labrande. *Explicit computation of the Abel-Jacobi map and its inverse*. Theses, Université de Lorraine ; University of Calgary, November 2016.

[56] Serge Lang. *Fundamentals of Diophantine geometry*. Springer Science & Business Media, 2013.

[57] Qing Liu. *Algebraic geometry and arithmetic curves*, volume 6 of *Oxford Graduate Texts in Mathematics*. Oxford University Press, Oxford, 2002. Translated from the French by Reinie Erné, Oxford Science Publications.

[58] The LMFDB Collaboration. The L-functions and modular forms database. `http://www.lmfdb.org`, 2013. [Online; accessed 16 September 2013].

[59] Paul Lockhart, Michael Rosen, and Joseph H Silverman. An upper bound for the conductor of an abelian variety. *J. Algebraic Geom*, 2(4):569–601, 1993.

[60] Maplesoft. Maplesoft, a division of Waterloo Maple inc., Waterloo, Ontario. `http://www.maplesoft.com`, 2017.

[61] Nicolas Mascot. Computing modular Galois representations. *Rend. Circ. Mat. Palermo (2)*, 62(3):451–476, 2013.

[62] John M McNamee and Victor Pan. *Numerical methods for roots of polynomials*, volume 16. Newnes, 2013.

[63] Rick Miranda. *Algebraic Curves and Riemann Surfaces (Graduate Studies in Mathematics, Vol 5)*. American Mathematical Society, 4 1995.

[64] Pascal Molin. *Intégration numérique et calculs de fonctions L*. PhD thesis, Université de Bordeaux I, 2010.

[65] Pascal Molin and Christian Neurohr. Computing period matrices and the abel-jacobi map of superelliptic curves. *Mathematics of Computation*, Dec 2017.

[66] Pascal Molin and Christian Neurohr. hcperiods: Arb and Magma packages for periods of superelliptic curves. `http://doi.org/10.5281/zenodo.1098275`, July 2017.

[67] Steffen Müller. Computing canonical heights using arithmetic intersection theory. *Mathematics of Computation*, 83(285):311–336, 2014.

[68] David Mumford. Tata lectures on Theta. I, volume 28 of progress in mathematics, 1983.

[69] Jürgen Neukirch. *Algebraische Zahlentheorie*. Springer-Verlag, Berlin, 1992.

[70] Harald Niederreiter and Chaoping Xing. *Algebraic geometry in coding theory and cryptography*. Princeton University Press, 2009.

[71] Victor Y Pan. Univariate polynomials: nearly optimal algorithms for factorization and rootfinding. In *Proceedings of the 2001 international symposium on Symbolic and algebraic computation*, pages 253–267. ACM, 2001.

[72] Miodrag Petkovic. *Point estimation of root finding methods*. Springer, 2008.

[73] Adrien Poteaux. Computing monodromy groups defined by plane algebraic curves. In *Proceedings of the 2007 international workshop on Symbolic-numeric computation*, pages 36–45. ACM, 2007.

[74] Adrien Poteaux. *Calcul de développements de Puiseux et application au calcul du groupe de monodromie d'une courbe algébrique plane*. PhD thesis, Université de Limoges, 2008.

[75] William H Press, Saul A Teukolsky, William T Vetterling, and Brian P Flannery. *Numerical recipes in C*, volume 2. Cambridge university press Cambridge, 1996.

[76] Daniel Quillen. Finite generation of the groups K_i of rings of algebraic integers. *Higher K-Theories*, pages 179–198, 1973.

[77] Th Rivlin. Chebyshev polynomials. from approximation theory to algebra and number theory. 1990. *Pure Appl. Math.(NY)*, 1990.

[78] Arnold Schönhage. The fundamental theorem of algebra in terms of computational complexity. *Manuscript. Univ. of Tübingen, Germany*, 1982.

[79] Jean-Pierre Serre. Géométrie algébrique et géométrie analytique. In *Annales de l'institut Fourier*, volume 6, pages 1–42. Association des Annales de l'Institut Fourier, 1956.

[80] Henning Stichtenoth. *Algebraic function fields and codes*, volume 254 of *Graduate Texts in Mathematics*. Springer-Verlag, Berlin, second edition, 2009.

[81] Michael Stoll. *Lecture notes: Algebraische Kurven*. Mathematisches Institut der Universität Düsseldorf, 2001/2002.

[82] C. Swierczewski et al. abelfunctions: A library for computing with abelian functions, Riemann surfaces, and algebraic curves. `http://github.com/abelfunctions/abelfunctions`, 2017.

[83] Hidetosi Takahasi and Masatake Mori. Double exponential formulas for numerical integration. *Publications of the Research Institute for Mathematical Sciences*, 9(3):721–741, 1974.

[84] The SAGE Developers. Sagemath, the Sage Mathematics Software System (Version 8.0). `http://www.sagemath.org`, 2018.

[85] Christopher Towse. Weierstrass points on cyclic covers of the projective line. *Transactions of the American Mathematical Society*, 348(8):3355–3378, 1996.

[86] Lloyd N Trefethen. *Spectral methods in* MATLAB . SIAM, 2000.

[87] Lloyd N Trefethen. Is Gauss quadrature better than Clenshaw-Curtis? *SIAM review*, 50(1):67–87, 2008.

[88] C.L. Tretkoff and M.D. Tretkoff. Combinatorial group theory, Riemann surfaces and differential equations. *Contemporary Mathematics*, 33:467–517, 1984.

[89] Raymond van Bommel. Numerical verification of the Birch and Swinnerton-Dyer conjecture for hyperelliptic curves of higher genus over $\mathbb{Q}$ up to squares. *Preprint arXiv:1711.10409*, 2017.

[90] Paul Van Wamelen. Equations for the Jacobian of a hyperelliptic curve. *Transactions of the American Mathematical Society*, 350(8):3083–3106, 1998.

[91] Paul Van Wamelen. Examples of genus two CM curves defined over the rationals. *Mathematics of Computation of the American Mathematical Society*, 68(225):307–320, 1999.

[92] Paul Van Wamelen. Proving that a genus 2 curve has complex multiplication. *Mathematics of Computation of the American Mathematical Society*, 68(228):1663–1677, 1999.

[93] Paul van Wamelen. Computing with the analytic Jacobian of a genus 2 curve. In *Discovering mathematics with Magma*, volume 19 of *Algorithms Comput. Math.*, pages 117–135. Springer, Berlin, 2006.

[94] Joachim Von Zur Gathen and Jürgen Gerhard. *Modern computer algebra*. Cambridge university press, 2013.

[95] Jörg Waldvogel. Fast construction of the Fejér and Clenshaw-Curtis quadrature rules. *BIT Numerical Mathematics*, 46(1):195–202, 2006.

[96] Robert John Walker. *Algebraic curves*, volume 642. Princeton University Press Princeton, 1950.