Computing the Riemann Constant Vector

Bernard Deconinck *
University of Washington
Department of Applied Mathematics
Seattle, WA 98195-3925

Matthew S. Patterson †
8506 37th Ave SW
Seattle, WA 98126

Christopher Swierczewski ‡
University of Washington
Department of Applied Mathematics
Seattle, WA 98195-3925

July 27, 2015

Abstract

The Riemann constant vector is a fundamental ingredient in the study of Riemann surfaces and their Jacobians. It is necessary to discuss the Jacobi inversion problem, for the study of the theta divisor, and for periodic solutions to integrable partial differential equations.

We present a mathematical algorithm and an implementation for computing the Riemann constant vector on a Riemann surface given by the desingularization and compactification of a complex plane algebraic curve. The source code of the implementation is provided in the Python software package ABELFUNCTIONS [33].

1 Introduction

This paper presents the next step in an ongoing research program to make effective the calculus on Riemann surfaces represented by complex plane algebraic curves. Here, "effective" means algorithms are devised and implemented in the form of black-box programs so that different relevant quantities associated with Riemann surfaces may be computed using a combination of symbolic and numerical tools in an efficient way. One of our main objectives in this program is to compute the so-called finite-genus solutions of integrable partial differential equations, such as the Korteweg-deVries (KdV), Nonlinear Schrödinger (NLS), and Kadomtsev-Petviashvili (KP) equations. These partial differential equations have been used extensively for describing a wide variety of physical phenomena ranging from water waves, nonlinear optics, and plasma physics to biological applications and cellular automata [1, 2, 13, 19]. Other applications are found in convex optimization and number theory [3, 23, 26, 27, 28].

*deconinc@uw.edu
†matthew.s.patterson@icloud.com
‡cswiercz@uw.edu

As an example, consider the finite-genus solutions $u = u(x, y, t)$ to the KP equation given by

$$u = c + 2\partial_x^2 \log \theta\Big(\boldsymbol{U}x + \boldsymbol{V}y + \boldsymbol{W}t + \boldsymbol{A}\left(P^\infty, \mathcal{D}\right) - \boldsymbol{K}(P^\infty), \Omega\Big), \tag{1}$$

where c is a constant, $\boldsymbol{U}$,$\boldsymbol{V}$, and $\boldsymbol{W}$ are vectors, and $\theta(z, \Omega)$ denotes the Riemann theta function parameterized by the Riemann matrix Ω of Riemann surface X. $\mathcal{D}$ is a divisor on X, $\boldsymbol{A}(P^\infty, \mathcal{D})$ is its Abel map with initial place P^∞, and $\boldsymbol{K}(P^\infty)$ is the Riemann constant vector at P^∞. Details on all of these components and discussion of the finite-genus solution above are found in [4, 15]. The computation of the Riemann matrix Ω is the main topic of [14]. The numerical calculation of the Riemann theta function is discussed in [10]. Future papers will present algorithms for computing the remaining quantities and efficiently computing $u = u(x, y, t)$.

Other approaches exist for computing with Riemann surfaces. Bobenko and collaborators [4, 7] compute solutions of integrable equations using a Schottky group representation for the associated surface. To our knowledge, the only paper dealing with all Riemann surfaces represented by algebraic curves is by Frauendiener, Klein, and Shramchenko who compute the homology of a Riemann surface from the monodromy of an underlying algebraic curve, following [14]. Otherwise, authors have restricted themselves to specific families of Riemann surfaces such as hyperelliptic ones [20, 21] or low genus ones [16, 30, 34]. Our aim throughout is the development of algorithms capable of dealing with arbitrary compact connected Riemann surfaces, as is required for the investigation of solutions of, for instance, the KP equation [13, 31].

In this paper we present a mathematical algorithm for computing the Riemann constant vector (RCV) and a demonstration of its implementation in ABELFUNCTIONS, an open–source Python library. The software implementation details, documentation, additional examples, and installation instructions are found on the project website [33] as well as the computational examples presented in this paper in the form of an iPython notebook.

2 Definitions and Background

In this section the required ingredients from the theory of Riemann surfaces are introduced. Details can be found in the standard references [18, 32] and the review paper [15]. A computational approach to these topics is found in [6, 11] and [12].

Let C be a complex plane algebraic curve $C = \{(\alpha, \beta) \in \mathbb{C}^2 : f(\alpha, \beta) = 0\}$ where $f \in \mathbb{C}[x, y]$ is a polynomial $f(x, y) = \sum_{k=0}^{n} \alpha_k(x) y^k$ with $\alpha_k \in \mathbb{C}[x]$. Let X be the genus g compact and connected Riemann surface obtained by desingularizing and compactifying the curve C; *i.e.*, X is a compact, connected, complex manifold of complex dimension one. Every compact and connected Riemann surface can be obtained this way [22].

Given X of genus g we choose a canonical basis of cycles $\{a_1, \ldots, a_g, b_1, \ldots, b_g\}$ for the first homology group as well as a normalized basis of Abelian differentials of the first kind $\{\omega_1, \ldots, \omega_g\}$. We define the Jacobian $J(X)$ of X using these two ingredients. The *period*

matrix $[I \;\; \Omega] \in \mathbb{C}^{g\times 2g}$ of X is constructed by

$$\oint_{a_j} \omega_i = \delta_{ij}, \qquad \oint_{b_j} \omega_i = \Omega_{ij}. \tag{2}$$

The matrix $\Omega \in \mathbb{C}^{g\times g}$ is a *Riemann matrix*: a symmetric complex $g \times g$ matrix with positive definite imaginary part. The Jacobian is given by the quotient space $J(X) = \mathbb{C}^g / \Lambda$ where $\Lambda = \mathbb{Z}^g + \Omega\, \mathbb{Z}^g$ is the *period lattice*.

Algorithms for computing a canonical basis of cycles, a basis of Abelian differentials of the first kind, and Riemann matrices are given in [35], [24, 36], and [14], respectively. The basis of differentials $\{\tilde{\omega}_1, \ldots, \tilde{\omega}_g\}$ returned by the algorithm in [24, 36] is not necessarily normalized. If a normalized basis is desired one can be determined by computing the period matrix $[A \;\; B] \in \mathbb{C}^{g\times 2g}$ defined by

$$\oint_{a_j} \tilde{\omega}_i = A_{ij}, \qquad \oint_{b_j} \tilde{\omega}_i = B_{ij}. \tag{3}$$

The associated Riemann matrix is constructed by $\Omega = A^{-1}B$ and a normalized basis of Abelian differentials is determined by $\boldsymbol{\omega} = A^{-1}\tilde{\boldsymbol{\omega}}$ with $\tilde{\boldsymbol{\omega}} = [\tilde{\omega}_1, \ldots, \tilde{\omega}_g]^T$.

Example 1. Let X be the Riemann surface obtained via desingularization and compactification of the non-hyperelliptic, genus 4 curve

$$C : f(x, y) = x^2y^3 - x^4 + 1 = 0. \tag{4}$$

We use this Riemann surface as an example throughout this paper. First we compute a basis, not necessarily normalized, for the space of Abelian differentials of the first kind.

In [1]:
```
from sympy.abc import x,y
from abelfunctions import (RiemannSurface, RiemannTheta, Jacobian,
                           AbelMap, RiemannConstantVector, puiseux)

f = x**2*y**3 - x**4 + 1
X = RiemannSurface(f,x,y)
g = X.genus()

omega = X.holomorphic_differentials()
print 'differentials:'
for omegai in omega:
    print omegai
print 'genus:', g
```

Out[1]:
```
differentials:
1/(3*x**2*y**2)
x/(3*x**2*y**2)
x*y/(3*x**2*y**2)
x**2/(3*x**2*y**2)
genus: 4
```

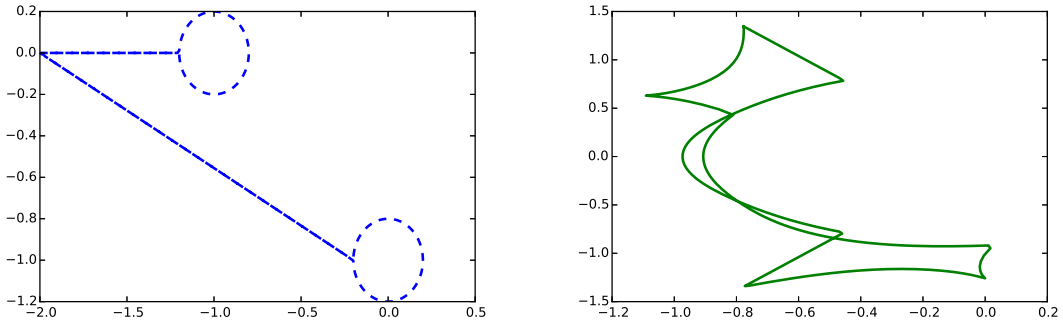
Next, we compute a canonical basis of cycles on X. We plot the projection of the cycle a_1 in the complex x- and y-planes, respectively.

In [2]:

```
a = X.a_cycles()
b = X.b_cycles()

# use 256 points to plot the x- and y-parts of the path
a[0].plot_x(256, color='blue', linewidth=2, linestyle='dashed')
a[0].plot_y(256, color='green', linewidth=2)
```

Out[2]:



Finally, we compute the associated Riemann matrix by numerically integrating the non-normalized basis of holomorphic differentials around the a- and b-cycles and computing $\Omega = A^{-1}B$. By default, ABELFUNCTIONS numerically integrates differentials along Riemann surface paths using a tolerance of 10^{-8}. We verify that Ω is symmetric and has positive definite imaginary part by computing $||\Omega - \Omega^T||_2$ and the eigenvalues of $\mathrm{Im}(\Omega)$.

In [3]:

```
# for brevity, we only print the first four significant digits
import numpy
numpy.set_printoptions(precision=4, suppress=True)

tau = X.period_matrix()
A = tau[:g,:g]
B = tau[:g,g:]
Omega = X.riemann_matrix()  # returns A**(-1)*B

print A
print B
print Omega
```

Out[3]:

```
[[ 0.2800+1.045j   0.2800-0.485j  -1.8100+1.045j   0.0000-0.j    ]
 [ 0.6625-1.1475j  0.6625+0.3825j -0.6625-1.1475j -0.0000+1.53j  ]
 [-0.8347+0.4819j -0.8347+0.4819j  0.8347-1.4457j  0.0000+1.9276j]
 [-1.0450+0.28j   -1.0450+1.81j   -0.4850+0.28j    0.0000+0.j    ]]
```

```
[[-0.2800+0.485j   0.2800-1.045j   0.0000-2.09j    0.7650-1.325j ]
 [ 0.6625+0.3825j -0.6625+0.3825j  0.0000-0.765j   0.0000-1.53j  ]
 [-0.8347+0.4819j  0.8347-1.4457j  0.0000-0.9638j -1.6694-0.9638j]
 [ 1.0450-1.81j   -1.0450-0.28j   -0.0000-0.56j    0.7650-1.325j ]]

[[ 0.3934+0.795j  -0.7541-0.3691j -0.4426-0.0284j  0.2049+0.2697j]
 [-0.7541-0.3691j  0.2787+0.8518j  0.0984+0.1988j -0.4344-0.1562j]
 [-0.4426-0.0284j  0.0984+0.1988j -0.3770+0.6815j -0.9180+0.4543j]
 [ 0.2049+0.2697j -0.4344-0.1562j -0.9180+0.4543j -1.2787+0.8802j]]
```

In [4]:

```
symmetric_error = numpy.linalg.norm(Omega - Omega.T)
imag_part_evals = numpy.linalg.eigvals(Omega.imag)

print 'error:', symmetric_error
print 'evals:', imag_part_evals
```

Out[4]:

```
error: 3.54420889595e-10
evals: [ 1.4038  1.1654  0.4294  0.21  ]
```

2.1 Places and Divisors

Definition 2. *Given a place $P \in X$, a local representation of the Riemann surface centered at P is given using a Puiseux series*

$$P = \begin{cases} x_P(t) = \alpha + \lambda t^e, \\ y_P(t) = \sum_{k=0}^{\infty} \beta_k t^{n_k}, \end{cases} \tag{5}$$

where $\alpha, \lambda, \beta_k \in \mathbb{C}$, and $e, n_k \in \mathbb{Z}$ [5].

Places lie "above" the curve C in the sense that evaluating $P = (x_P(t), y_P(t))$ at $t = 0$ maps the place P onto a point (α, β) of the curve.

Let $R(f, \partial_y f)(x)$ be the resultant of $f(x, y)$ and $\partial_y f(x, y)$ with respect to y [22]. The roots $\alpha \in \mathbb{C}$ of R correspond to the *discriminant points* of C, which consist of the branch points and singular points of the curve. A place is called *discriminant* if it lies above a discriminant point of C. Otherwise, it is *regular*. An algorithm for computing Puiseux series expansions is found in [17] and [29].

Example 3. Continuing from Example 1, there is one place on X lying above the discriminant point $x = 0$:

$$P = \begin{cases} x_P(t) &= -t^3, \\ y_P(t) &= -t^{-2} + O\left(t^2\right). \end{cases} \tag{6}$$

Let α_1, α_2, and α_3 denote the roots of the polynomial $f(2,y) = 4y^3 - 15$. There are three regular places P_1,P_2, and P_3 lying above $x = 2$:

$$P_i = \begin{cases} x_{P_i}(t) & = 2 + t, \\ y_{P_i}(t) & = \alpha_i + \frac{17}{45}\alpha_i t + O\left(t^2\right), \end{cases} \tag{7}$$

We confirm the form of these places computationally.

In [5]:
```
places_above_zero = X(0)
print places_above_zero
```

Out[5]:
```
[(-t**3, -1/t**2 + O(t**2))]
```

By default, ABELFUNCTIONS does not determine the Puiseux series expansions of regular places since each regular place above a given $x = \alpha$ maps to a unique point $(\alpha, \beta) \in C$. We can request these series expansions using the `puiseux()` function.

In [6]:
```
print 'Places:'
places_above_two = X(2)
for P in places_above_two:
    print P

print 'Puiseux:'
series_at_two = puiseux(f,x,y,2)
for p in series_at_two:
    print p
```

Out[6]:
```
Places:
(2, RootOf(4*_y**3 - 15, 0))
(2, RootOf(4*_y**3 - 15, 1))
(2, RootOf(4*_y**3 - 15, 2))

Puiseux:
(t + 2, RootOf(4*_y**3 - 15, 0) + 17*t*RootOf(4*_y**3 - 15, 0)/45 + O(t**2))
(t + 2, RootOf(4*_y**3 - 15, 1) + 17*t*RootOf(4*_y**3 - 15, 1)/45 + O(t**2))
(t + 2, RootOf(4*_y**3 - 15, 2) + 17*t*RootOf(4*_y**3 - 15, 2)/45 + O(t**2))
```

Related to places is the notion of a divisor [22, 32].

Definition 4. *A divisor $\mathcal{D}$ on the Riemann surface X is a finite formal linear combination of places P_i with multiplicities n_i:*

$$\mathcal{D} = \sum_i n_i P_i. \tag{8}$$

The sum

$$\deg \mathcal{D} = \sum_i n_i \tag{9}$$

is called the degree of $\mathcal{D}$.

The set of all divisors on a Riemann surface forms an Abelian group $\mathrm{Div}(X)$ under addition. A divisor with all $n_i \geq 0$ is called *positive* or *effective*. A *valuation divisor*, important for the calculation of the Riemann constant vector, is obtained by examining the root and pole structure of a meromorphic one-form on X.

Definition 5. *Let Ω^1_X denote the set of meromorphic one-forms on X and let $\nu \in \Omega^1_X$ have m zeros of multiplicity p_j at the places P_j and n poles of multiplicity q_j at the places Q_j. Then*

$$(\nu)_{val} = \sum_{i=1}^{m} p_i P_i - \sum_{j=1}^{n} q_j Q_j \tag{10}$$

is called the valuation divisor of ν. A divisor $\mathcal{C} \in Div(X)$ is called canonical if $\mathcal{C} = (\nu)_{val}$ for some $\nu \in \Omega^1_X$.

All canonical divisors have the same degree and, by the Riemann–Roch Theorem, this degree is $\deg \mathcal{C} = 2g-2$ [32]. Note that every Abelian differential of the first kind is trivially meromorphic, since it is a holomorphic one-form, and its valuation divisor has $q_j = 0$ for all j. Consequently, every such differential has exactly $2g-2$ zeros including multiplicities.

Example 6. We use the places computed in Example 3 to construct divisors on the Riemann surface X.

In [7]:
```
P = places_above_zero[0]
Q = places_above_two[0]
D = 3*P + Q
print 'Divisor:', D
print 'Degree:', D.degree
```

Out[7]:
```
Divisor: 3(-t**3, -1/t**2 + O(t**2)) + (2, RootOf(4*_y**3 - 15, 0))
Degree: 4
```

2.2 The Abel Map

Definition 7. *Let $P \in X$ be a fixed place. The Abel Map $\boldsymbol{A} : X \to J(X)$ is defined by*

$$\boldsymbol{A}(P,Q) = \big(A_1(P,Q), \ldots, A_g(P,Q)\big), \tag{11}$$

where

$$A_j(P,Q) = \int_P^Q \omega_j, \tag{12}$$

and the path chosen from P to Q is the same for each A_j. The Abel map is written in vector form as

$$\boldsymbol{A}(P,Q) = \int_P^Q \boldsymbol{\omega}. \tag{13}$$

The definition of the Abel map can be extended to divisors: let $\mathcal{D} = \sum_i n_i P_i$. We define

$$\boldsymbol{A}(P, \mathcal{D}) = \sum_i n_i \boldsymbol{A}(P, P_i). \tag{14}$$

The Abel Map is independent of the path γ from P to Q chosen on X for if γ and η are two such paths then their difference is a linear combination of homology basis cycles. The integral of $\boldsymbol{\omega}$ along this closed path is a lattice element and therefore is congruent to zero in $J(X)$.

An algorithm for computing the Abel map is described in [12]. The implementation in ABELFUNCTIONS is based on this algorithm.

Example 8. ABELFUNCTIONS selects a regular "base place" P_0 from which to construct all paths on X. When given a single argument `AbelMap()` returns $\boldsymbol{A}(P_0, P)$.

In [8]:
```
J = Jacobian(X)    # reduces vectors modulo lattice ZZ^g + Omega ZZ^g
z1 = AbelMap(P)    # Abel map from P0 to P
z2 = AbelMap(Q)    # Abel map from P0 to Q
z3 = AbelMap(P,Q)  # Abel map from P to Q
print z1
print z2
print z3

# numerically verify that A(P,Q) = A(P0,Q) - A(P0,P)
print numpy.linalg.norm( J((z2-z1) - z3) )
```

Out[8]:
```
[-0.5261+0.0864j  0.0669+0.6392j -0.7495+1.1037j -1.5030+1.0356j]
[-0.3875+0.1157j -0.0290+0.4437j -0.4532+0.7774j -0.9721+0.6732j]
[ 0.1468-0.0985j  0.8467+0.6989j  0.0996+1.0083j -1.1003+0.8159j]
3.80631643473e-16
```

The Abel map accepts divisors as well.

In [9]:
```
w = AbelMap(D)
print w

# verify that w = 3*z1 + z2 mod period lattice
z = J(3*z1 + z2)
print numpy.linalg.norm(w-z)
```

Out[9]:
```
[ 0.0670-0.1361j  0.9421+0.7429j -0.4887+0.7663j -1.5057+0.6992j]
1.57107346e-15
```

2.3 The Riemann Constant Vector

Definition 9. *Let X be a genus g Riemann surface with associated Riemann matrix Ω. The Riemann constant vector $\boldsymbol{K} : X \to J(X)$ is defined as*

$$\boldsymbol{K}(P) = \big(K_1(P), \ldots, K_g(P)\big), \tag{15}$$

where

$$K_j(P) = \frac{1+\Omega_{jj}}{2} - \sum_{k\neq j}^{g} \oint_{a_k} \omega_k(Q) A_j(P,Q). \tag{16}$$

Once we know the value of $\boldsymbol{K}(P_0)$ the value of $\boldsymbol{K}(P)$ is determined using only a shift by the Abel map:

Theorem 10. *Let P_0, P be places on a genus g Riemann surface X. Then*

$$\boldsymbol{K}(P) = \boldsymbol{K}(P_0) + (g-1)\,\boldsymbol{A}(P_0, P). \tag{17}$$

Proof. Let Q be an arbitrary place on X. By the definition of the Abel map, $\boldsymbol{A}(P,Q) = \boldsymbol{A}(P,P_0) + \boldsymbol{A}(P_0,Q)$. Using this identity in the definition of the RCV we obtain

$$\begin{aligned} K_j(P) &= \frac{1+\Omega_{jj}}{2} - \sum_{k\neq j}^{g} \oint_{a_k} \omega_k(Q) A_j(P,Q)\,\mathrm{d}Q \\ &= \frac{1+\Omega_{jj}}{2} - \sum_{k\neq j}^{g} \oint_{a_k} \omega_k(Q)\Big(A_j(P,P_0) + A_j(P_0,Q)\Big)\,\mathrm{d}Q \\ &= \frac{1+\Omega_{jj}}{2} - \sum_{k\neq j}^{g} \oint_{a_k} \omega_k(Q) A_j(P,P_0)\,\mathrm{d}Q - \sum_{k\neq j}^{g} \oint_{a_k} \omega_k(Q) A_j(P_0,Q)\,\mathrm{d}Q. \end{aligned} \tag{18}$$

The jth-component of the Abel map appearing in the first sum has no dependence on the variable of integration Q nor on the summation index k. Therefore,

$$\begin{aligned} K_j(P) &= \frac{1+\Omega_{jj}}{2} - A_j(P,P_0) \sum_{k\neq j}^{g} \oint_{a_k} \omega_k(Q)\,\mathrm{d}Q - \sum_{k\neq j}^{g} \oint_{a_k} \omega_k(Q) A_j(P_0,Q)\,\mathrm{d}Q \\ &= \frac{1+\Omega_{jj}}{2} - A_j(P,P_0)(g-1) - \sum_{k\neq j}^{g} \oint_{a_k} \omega_k(Q) A_j(P_0,Q)\,\mathrm{d}Q \\ &= K_j(P_0) + (g-1) A_j(P_0,P). \end{aligned} \tag{19}$$

□

The primary computational benefit to using the result of Theorem 10 is that most of the work in evaluating $\boldsymbol{K}$ comes from evaluating it at a fixed place P_0. Once this is done, we

only need the Abel map to determine $\boldsymbol{K}$ for all other places $P \in X$. In ABELFUNCTIONS a fixed place of the Riemann surface is automatically chosen.

The inspiration behind the algorithm for computing the RCV described in the following section comes from the following two theorems. Theorem 11 characterizes a certain class of divisors in terms of the RCV, a proof of which is found in [18].

Theorem 11. *Let $\mathcal{C}$ be a divisor on a genus g Riemann surface X of degree $2g-2$. Then $\mathcal{C}$ is a canonical divisor if and only if*

$$2\,\boldsymbol{K}(P) \equiv -\,\boldsymbol{A}(P,\mathcal{C}). \tag{20}$$

Theorem 13 establishes a connection between the Riemann theta function and the RCV, a proof of which is also found in [18].

Definition 12. *The Riemann theta function $\theta : J(X) \times \mathfrak{h}_{\mathrm{g}} \to \mathbb{C}$ is defined by*

$$\theta(z,\Omega) = \sum_{n\in\mathbb{Z}^g} e^{2\pi i\left(\frac{1}{2}n\cdot\Omega n + n\cdot z\right)}. \tag{21}$$

This series converges absolutely and uniformly on compact sets in $J(X)\times\mathfrak{h}_{\mathrm{g}}$ where $\mathfrak{h}_{\mathrm{g}}$ is the space of all Riemann matrices.

Theorem 13. *Let Ω be the Riemann matrix associated with the Riemann surface X and $P_0 \in X$ an arbitrary place. Then a vector $\boldsymbol{W} \in J(X)$ satisfies*

$$\theta(\boldsymbol{W},\Omega) = 0, \tag{22}$$

if and only if there exists a divisor $\mathcal{D} = P_1 + \cdots P_{g-1}$ such that

$$\boldsymbol{W} = \boldsymbol{A}(P_0,\mathcal{D}) + \boldsymbol{K}(P_0). \tag{23}$$

Note that $\mathcal{D}$ may contain a place of multiplicity greater than one. The primary requirement of $\mathcal{D}$ is that it is of degree $g-1$ and is effective. The set $\Theta := \{\boldsymbol{W} \in J(X) : \theta(\boldsymbol{W},\Omega) = 0\}$ is known as the *theta divisor* of the Riemann surface X. It is a $(g-1)$ complex–dimensional subvariety of $J(X)$. Theorem 13 states that $\Theta = \boldsymbol{A}\left(P_0, SX^{g-1}\right) + \boldsymbol{K}(P_0)$ where SX^{g-1} is the $(g-1)$–fold symmetric product of the Riemann surface.

3 Computing the Riemann Constant Vector

In this section we present an algorithm for computing the Riemann constant vector as well as a demonstration of its implementation in ABELFUNCTIONS. First we present an overview of and the motivation behind the algorithm. We describe the two primary components of the algorithm later in this section.

Theorem 11 suggests an approach to computing the RCV provided we can compute a canonical divisor of the Riemann surface. However, even with such a divisor the theorem only makes a statement about the value of $2\,\boldsymbol{K}(P_0)$. That is, one would like to say

$$\boldsymbol{K}(P_0) \equiv -\tfrac{1}{2}\,\boldsymbol{A}(P_0, \mathcal{C}), \tag{24}$$

but division is not unique in this equivalence class. In general, there are 2^{2g} *half-lattice vectors* $\boldsymbol{h} \in \frac{1}{2}\Lambda$ such that $\boldsymbol{K}(P_0) \equiv \boldsymbol{h} - \frac{1}{2}\,\boldsymbol{A}(P_0, \mathcal{C})$. Therefore, a second objective is to find an appropriate half-lattice vector.

Algorithm 1 `riemann_constant_vector`

Input: Riemann surface X given by the desingularization and compactification of complex plane algebraic curve $C : f(x, y) = 0$
Input: place $P \in X$
Output: Riemann constant vector $\boldsymbol{K}(P)$

1: compute the Riemann matrix Ω
2: $\mathcal{C} \leftarrow$ `canonical_divisor()` ▷ Algorithm 2
3: $\boldsymbol{h} \leftarrow$ `half_lattice_vector()` ▷ Algorithm 3
4: $\boldsymbol{K}_0 \leftarrow \boldsymbol{h} - \frac{1}{2}\,\boldsymbol{A}(P_0, \mathcal{C}) \bmod \Lambda$
5: $\boldsymbol{K} \leftarrow \boldsymbol{K}_0 + (g-1)\,\boldsymbol{A}(P_0, P)$
6: **return** $\boldsymbol{K}$

The rest of this section presents in detail the subroutines `canonical_divisor` and `half_lattice_vector` which provide the necessary remaining ingredients for the above algorithm.

3.1 Computing a Canonical Divisor

Determining the zeros and poles of a meromorphic one-form

$$\nu = \frac{p(x, y)}{q(x, y)}\,\mathrm{d}x \tag{25}$$

is not as straightforward as finding the roots of the polynomials p and q. One challenge comes from analyzing the local behavior of $\mathrm{d}x$. Furthermore, it may occur that the numerator and denominator have the same order of vanishing at some place $P \in X$ in which case P is neither a root nor pole of ν.

Let $P \in X$ be a place with Puiseux series representation $(x_P(t), y_P(t))$ where t is a local parameter as given in Definition 2. A necessary condition for P to be a root or pole of ν is that

$$p\big(x_P(t), y_P(t)\big)\Big|_{t=0} = 0, \qquad q\big(x_P(t), y_P(t)\big)\Big|_{t=0} = 0, \qquad \text{or} \qquad \frac{\mathrm{d}x_P}{\mathrm{d}t}\big(0\big) = 0. \tag{26}$$

In particular, to determine if $P \in (\nu)_{\text{val}}$ we substitute the Puiseux series representation into ν and expand as a Laurent series in t:

$$\begin{aligned}\nu\Big|_P &= \frac{p\big(x_P(t), y_P(t)\big)}{q\big(x_P(t), y_P(t)\big)}\,\mathrm{d}x_P(t) \\ &= \frac{p(t)}{q(t)} x_P'(t)\,\mathrm{d}t \\ &= \Big(ct^{\text{val}(\nu,P)} + \cdots\Big)\,\mathrm{d}t, \end{aligned} \tag{27}$$

where $\text{val}(\nu, P)$ is the leading order behavior of ν at P. This gives us a test for determining if P is a member of the set of places appearing in $(\nu)_{\text{val}}$: if $\text{val}(\nu, P) < 0$ then P is a pole, if $\text{val}(\nu, P) > 0$ then P is a zero, otherwise P does not appear in the valuation divisor of ν. The multiplicity of the zero or pole is equal to $|\text{val}(\nu, P)|$.

With the above membership test what remains is to construct a set of places $\mathcal{P}_\nu$ guaranteed to contain the places appearing in $(\nu)_{\text{val}}$. We obtain the valuation divisor by applying the membership test to each $P \in \mathcal{P}_\nu$. Consider the resultant $R(f, p)(x)$ of f and p with respect to y. By definition, the roots of R are the points $\alpha \in \mathbb{C}$ such that

$$f(\alpha, y) = 0 \quad \text{and} \quad p(\alpha, y) = 0 \tag{28}$$

have simultaneous solutions. Therefore, for a place P to be a zero of p it must be the case that the x-projection of P, $x_P(0)$, is a root of the resultant R. Similarly, for P to be a zero of q its x-projection $x_P(0)$ must be a root of the resultant $R(f, q)(x)$. We also need to include the places P which cause $\mathrm{d}x$ to vanish. This occurs when $(\,\mathrm{d}x_P/\,\mathrm{d}t)(0) = x_P'(0) = 0$. That is, when P lies above a branch point of f.

Define the sets

$$\begin{aligned} \mathcal{X}_\nu^{(1)} &= \{\alpha \in \mathbb{C} \mid R(f, p)(\alpha) = 0\}, \\ \mathcal{X}_\nu^{(2)} &= \{\alpha \in \mathbb{C} \mid R(f, q)(\alpha) = 0\}, \\ \text{and} \quad \mathcal{X}_\nu^{(3)} &= \{\alpha \in \mathbb{C} \mid \alpha \text{ is a branch point of } f\}. \end{aligned} \tag{29}$$

Since the representation of the one-form in (25) only captures its affine behavior it is necessary to examine its behavior at all places lying above $x = \infty$. Define

$$\mathcal{X}_\nu = \mathcal{X}_\nu^{(1)} \cup \mathcal{X}_\nu^{(2)} \cup \mathcal{X}_\nu^{(3)} \cup \{\infty\}. \tag{30}$$

This consists of all x-points above which there may be a place P where ν vanishes. That is, the only places we need to check are those with x-projections in $\mathcal{X}_\nu$. Therefore, the set

$$\mathcal{P}_\nu = \{P \in X \mid x_P(0) \in \mathcal{X}_\nu\}, \tag{31}$$

is guaranteed to contain the places appearing in $(\nu)_{\text{val}}$. The Puiseux algorithm is well-designed to compute this set.

This procedure is simplified when computing the valuation divisor of an Abelian differential of the first kind. Recall that every such differential is trivially a meromorphic one-form and therefore can be used to compute a canonical divisor. We could use one of the normalized basis elements $\{\omega_1, \ldots, \omega_g\}$ to obtain a canonical divisor on X but it is preferred to use the non-normalized differentials $\{\tilde{\omega}_1, \ldots, \tilde{\omega}_g\}$ returned by ABELFUNCTIONS. This is done for several performance-related reasons:

- We already compute these differentials for the purposes of determining the period matrix of X as well as in defining the Abel map.
- Fewer resolvent sets need to be determined. The denominator of every Abelian differential of the first kind is $\partial_y f(x, y)$ [8, 25] so one can compute the resolvent set of f with $\partial_y f$ once and use the results for any given basis element $\tilde{\omega} = \tilde{\omega}_i$. This particular resolvent set consists of the discriminant points of f and is already used in the period matrix calculations.
- The non-normalized differentials usually have simple, often monomial, numerators making the set $\mathcal{X}_{\tilde{\omega}}^{(1)}$ easier to compute and have smaller cardinality.
- The set of P such that $x_P(0)$ is a branch point of f is contained in the set of discriminant points of f. Therefore, the computation of the set $\mathcal{X}_{\tilde{\omega}}^{(3)}$ is a redundant calculation and is omitted.
- In general, the valuation divisors of Abelian differentials of the first kind consist of fewer distinct places. The degree of every canonical divisor is $2g - 2$. Therefore, there must always be $2g - 2$ more zeros than poles, counting multiplicities. Since Abelian differentials of the first kind have no poles, no negative degree places appear in the valuation divisor thus minimizing the total number of places to check.
- Algorithm 2 iteratively checks each $P \in \mathcal{P}_{\tilde{\omega}}$ for membership in the set of places in $\mathcal{C} = (\tilde{\omega})_{\text{val}}$. By using Abelian differentials of the first kind we can terminate this procedure the moment $\deg \mathcal{C}$ reaches $2g - 2$ since each $P \in \mathcal{P}_{\tilde{\omega}}$ contributes a non-negative amount to the degree. For this reason, we distinguish between the set of $\mathcal{X}_{\tilde{\omega}}$ and corresponding places $\mathcal{P}_{\tilde{\omega}}$ in order to avoid unnecessarily computing Puiseux series expansions.

An algorithm for computing the valuation divisor of an Abelian differential of the first kind is given below.

Algorithm 2 `canonical_divisor` - canonical divisor of a Riemann surface

Input: Riemann surface X given by the desingularization and compactification of complex plane algebraic curve $C : f(x, y) = 0$

Input: an Abelian differential of the first kind $\tilde{\omega} = p(x, y)/\partial_y f(x, y)\,\mathrm{d}x$ on X

Output: canonical divisor $\mathcal{C} = (\tilde{\omega})_{\mathrm{val}}$

$\mathcal{C} \leftarrow$ zero divisor
$\mathcal{X}_{\tilde{\omega}}^{(1)} \leftarrow$ roots of resolvent $R(f, p)(x) = 0$
$\mathcal{X}_{\tilde{\omega}}^{(2)} \leftarrow$ discriminant points of f
$\mathcal{X}_{\tilde{\omega}} \leftarrow \mathcal{X}_{\tilde{\omega}}^{(1)} \cup \mathcal{X}_{\tilde{\omega}}^{(2)} \cup \{\infty\}$
for $\alpha \in \mathcal{X}_{\tilde{\omega}}$ **do**
    $\mathcal{P}_{\tilde{\omega}}^{\alpha} \leftarrow \{P \in X \mid x_P(0) = \alpha\}$
    for $P \in \mathcal{P}_{\tilde{\omega}}^{\alpha}$ **do**
        $n \leftarrow \mathrm{val}\,(\tilde{\omega}, P)$
        $\mathcal{C} \leftarrow \mathcal{C} + nP$
        if $\deg \mathcal{C} = 2g - 2$ **then**
            return $\mathcal{C}$
        end if
    end for
end for
raise error("Not enough places found.")

Some notes about the algorithm:

- Only the leading order behavior of the Puiseux series of each place $P \in \mathcal{P}_{\omega}^{\alpha}$ determining $\mathrm{val}(\tilde{\omega}, P)$ is needed implying that computing only the "singular part" of these expansions using the method of Duval is sufficient [17].

- Since any Abelian differential of the first kind is sufficient for computing a canonical divisor we can choose the basis element $\tilde{\omega}_i$ with lowest total degree numerator $p = p(x, y)$ to reduce the number of places to check and amount of symbolic arithmetic to perform.

- Algorithm 2 terminates once the target degree is met and will spend no further effort computing places and valuations. Since the numerators of $\tilde{\omega}_i$ are often monomial, a significant gain in efficiency is observed when $\mathcal{X}_{\tilde{\omega}}$ is ordered such that places over $x \in \{0, \infty\}$ are checked first. For testing purposes, the algorithm can be modified to verify that $\mathrm{val}(\tilde{\omega}, P) = 0$ for all remaining $P \in \mathcal{P}_{\tilde{\omega}}$ after the degree requirement is met.

- If the main loop in Algorithm 2 terminates before the requisite degree is achieved then an error is reported. This is included as an additional check for the algorithm. Yet another test is to verify that $\deg(\tilde{\omega}_i)_{\mathrm{val}} = 2g - 2$ for all basis elements $\tilde{\omega}_i$. Because this is an expensive calculation it is not performed by default. Instead it is relegated to the ABELFUNCTIONS test suite.

Example 14. For each non-normalized Abelian differential of the first kind from Example 1,

$$\tilde{\omega}_1 = \frac{\mathrm{d}x}{3x^2y^2}, \quad \tilde{\omega}_2 = \frac{x\,\mathrm{d}x}{3x^2y^2}, \quad \tilde{\omega}_3 = \frac{xy\,\mathrm{d}x}{3x^2y^2}, \quad \tilde{\omega}_4 = \frac{x^2\,\mathrm{d}x}{3x^2y^2}, \tag{32}$$

we compute its corresponding canonical divisor. First, we verify that

$$(\tilde{\omega}_1)_{\mathrm{val}} = 6P_{x=\infty} \qquad \text{where} \qquad P_{x=\infty} = \left(t^{-3}, t^{-2} + O\left(t^2\right)\right). \tag{33}$$

That is, the valuation divisor consists of the single place, $P_{x=\infty}$, of multiplicity six.

```
In [10]: C0 = omega[0].valuation_divisor()
         for place,multiplicity in C0:
             print multiplicity, place
         print 'Degree:', C0.degree
```

```
Out[10]: 6 (t**(-3), t**(-2) + O(t**2))
         Degree: 6
```

On the other hand, $(\tilde{\omega}_2)_{\mathrm{val}}$ consists of two distinct places each of multiplicity three:

$$(\tilde{\omega}_2)_{\mathrm{val}} = 3P_{x=\infty} + 3P_{x=0} \qquad \text{where} \qquad P_{x=0} = \left(-t^3, -t^{-2} + O(t^2)\right). \tag{34}$$

```
In [11]: C1 = omega[1].valuation_divisor()
         for place,multiplicity in C1:
             print multiplicity, place
         print 'Degree:', C1.degree
```

```
Out[11]: 3 (t**(-3), t**(-2) + O(t**2))
         3 (-t**3, -1/t**2 + O(t**2))
         Degree: 6
```

For the canonical divisor obtained from $(\tilde{\omega}_3)_{\mathrm{val}}$ we have

$$\mathcal{X}_{\tilde{\omega}_3} = \{0, \infty, 1, -1, i, -i\}, \tag{35}$$

which consists of the discriminant points of f and the point at infinity. The divisor $(\tilde{\omega}_3)_{\mathrm{val}}$ happens to have non-zero valuation at the places lying above each of these points:

$$(\tilde{\omega}_3)_{\mathrm{val}} = \sum_{\alpha \in \mathcal{X}_{\tilde{\omega}_3}} P_{x=\alpha}. \tag{36}$$

```
In [12]: C2 = omega[2].valuation_divisor()
         for place,multiplicity in C2:
             print multiplicity, place
         print 'Degree:', C2.degree
```

```
Out[12]: 1 (-t**3, -1/t**2 + O(t**2))
         1 (-t**3/4 - 1, t + O(t**2))
         1 (-t**3*RootOf(_x**2 + 1, 0)/4 + RootOf(_x**2 + 1, 0), t + O(t**2))
         1 (t**(-3), t**(-2) + O(t**2))
         1 (t**3/4 + 1, t + O(t**2))
         1 (-t**3*RootOf(_x**2 + 1, 1)/4 + RootOf(_x**2 + 1, 1), t + O(t**2))
         Degree: 6
```

Finally, we verify that $(\tilde{\omega}_4)_{\text{val}} = 6P_{x=0}$.

```
In [13]: C3 = omega[3].valuation_divisor()
         for place,multiplicity in C3:
             print multiplicity, place
         print 'Degree:', C3.degree
```

```
Out[13]: 6 (-t**3, -1/t**2 + O(t**2))
         Degree: 6
```

Each of these canonical divisors satisfy the degree requirement $\deg \mathcal{C} = 2g - 2 = 6$.

3.2 Computing a Half-Lattice Vector

Now that we have a canonical divisor $\mathcal{C}$ it remains to determine $\boldsymbol{K}(P_0)$ knowing that $2\,\boldsymbol{K}(P_0) \equiv -\boldsymbol{A}(P_0, \mathcal{C})$. For now, consider $\boldsymbol{K}(P_0)$ and $\boldsymbol{A}(P_0, \mathcal{C})$ to be vectors in $\mathbb{C}^g$ and set $\boldsymbol{K}_0 := \boldsymbol{K}(P_0)$ and $\boldsymbol{A}_0^{\mathcal{C}} := \boldsymbol{A}(P_0, \mathcal{C})$, for notational convenience. In $\mathbb{C}^g$ we have

$$2\,\boldsymbol{K}_0 + \boldsymbol{A}_0^{\mathcal{C}} = \boldsymbol{\lambda}, \tag{37}$$

where $\boldsymbol{\lambda} \in \mathbb{C}^g$ is unknown and $\boldsymbol{\lambda} \equiv \boldsymbol{0} \bmod \Lambda$, *i.e.* the vector $\boldsymbol{\lambda}$ is one of the 2^{2g} lattice vectors lying in the fundamental region of Λ. Division by two is now legal: setting $\boldsymbol{h} = \boldsymbol{\lambda}/2$ yields

$$\boldsymbol{K}_0 = \boldsymbol{h} - \tfrac{1}{2}\,\boldsymbol{A}_0^{\mathcal{C}}\,. \tag{38}$$

Reducing this expression modulo Λ gives the corresponding equivalence in $J(X)$

$$\boldsymbol{K}_0 \equiv \boldsymbol{h} - \tfrac{1}{2}\,\boldsymbol{A}_0^{\mathcal{C}}, \tag{39}$$

where the half-lattice vector $\boldsymbol{h}$ is unknown.

Algorithm 3 `half_lattice_vector`$(X, \mathcal{C})$

Input: a Riemann surface X
Input: a canonical divisor $\mathcal{C}$
Output: a half lattice vector $\boldsymbol{h}$

$\mathcal{J} \leftarrow \{1, \ldots, 2^{2g}\}$
$\mathcal{D} \leftarrow (g-1)P_0$
$\mathcal{J} \leftarrow$ `half_lattice_filter`$(\mathcal{J}, \mathcal{C}, \mathcal{D})$ ▷ filter pass #1
if $\mathcal{J} = \{j^*\}$ **return** $\boldsymbol{h}_{j^*}$
$\mathcal{D}_0 \leftarrow P_1 + \cdots P_{g-1}$ where the P_i's are distinct regular places
$\mathcal{J} \leftarrow$ `half_lattice_filter`$(\mathcal{J}, \mathcal{C}, \mathcal{D}_0)$ ▷ filter pass #2
if $\mathcal{J} = \{j^*\}$ **return** $\boldsymbol{h}_{j^*}$
for each $m_1, \ldots, m_{g-1} \geq 0$ with $m_1 + \cdots + m_{g-1} = g-1$ **do** ▷ filter pass #3
    $\mathcal{D}_k \leftarrow m_1 P_1 + \cdots + m_{g-1} P_{g-1}$
    $\mathcal{J} \leftarrow$ `half_lattice_filter`$(\mathcal{J}, \mathcal{C}, \mathcal{D}_k)$
    if $\mathcal{J} = \{j^*\}$ **return** $\boldsymbol{h}_{j^*}$
end for
raise error("Could not find appropriate half-lattice vector.")

To determine which of the 2^{2g} half-lattice vectors $\boldsymbol{h}_j, j = 1, \ldots, 2^{2g}$ is the correct half-lattice vector we use Theorem 13. The theorem requires a degree $g-1$ effective divisor. Consider the divisor

$$\mathcal{D} = (g-1)P_0. \tag{40}$$

Then

$$\begin{aligned}\theta\big(\boldsymbol{A}(P_0, \mathcal{D}) + \boldsymbol{K}(P_0), \Omega\big) &= \theta\bigg(\boldsymbol{A}\left(P_0, (g-1)P_0\right) + \boldsymbol{K}(P_0), \Omega\bigg) \\ &= \theta\big(\boldsymbol{0} + \boldsymbol{K}(P_0), \Omega\big) \\ &= \theta\big(\boldsymbol{K}(P_0), \Omega\big) = 0. \end{aligned} \tag{41}$$

Therefore, it is necessary that

$$\theta\left(\boldsymbol{h}_j - \tfrac{1}{2}\boldsymbol{A}_0^{\mathcal{C}}, \Omega\right) = 0 \tag{42}$$

for *at least* one of the 2^{2g} half lattice vectors $\boldsymbol{h}_j$. The choice of divisor $\mathcal{D}$ above simplifies the computations. However, any appropriate divisor can be used in conjunction with Theorem 13 to obtain the RCV.

The idea behind Algorithm 3 is to perform a number of "filter passes" to eliminate incorrect half-lattice vectors using the subroutine described in Algorithm 4. Each pass uses a different effective degree $g-1$ divisor beginning with the one defined in (40). This heuristic approach is used due to the numerical approximation inherent in evaluating the Riemann theta function as well as in numerically integrating the differentials along the Riemann surface paths.

Some notes on Algorithms 3 and 4:

Algorithm 4 `half_lattice_filter`$(\mathcal{J}, \mathcal{C}, \mathcal{D})$

Input: index set $\mathcal{J} \subset \{1, \ldots, 2^{2g}\}$
Input: a canonical divisor $\mathcal{C}$
Input: effective, degree $g-1$ divisor $\mathcal{D}$
Output: filtered index set $\tilde{\mathcal{J}}$

$\tilde{\mathcal{J}} \leftarrow \mathcal{J}$
$\theta(\cdot, \Omega) \leftarrow$ the Riemann theta function uniformly accurate to order ϵ
$\boldsymbol{Z} \leftarrow \boldsymbol{A}(P_0, \mathcal{D}) - \frac{1}{2}\boldsymbol{A}(P_0, \mathcal{C})$
for $j \in \mathcal{J}$ **do**
    $\boldsymbol{\kappa}_j \leftarrow \boldsymbol{h}_j + \boldsymbol{Z} \bmod \Lambda$
    if $||\theta(\boldsymbol{\kappa}_j)|| > \epsilon$ **then**
        remove j from $\tilde{\mathcal{J}}$
    end if
end for
return $\tilde{\mathcal{J}}$

- If a candidate vector $\boldsymbol{\kappa}_j \in J(X)$ is, in fact, the sought-after vector such that $\theta(\boldsymbol{\kappa}_j) = 0$ then it must be so for all effective degree $g-1$ divisors $\mathcal{D}$. That is, for every such divisor
$$\theta\big(\boldsymbol{\kappa}_j + \boldsymbol{A}(P_0, \mathcal{D}), \Omega\big) = 0. \tag{43}$$
This can be used for additional verification of our results.

- Care must be used when setting the numerical accuracy for the computation of the Riemann theta function. The default accuracy of $\epsilon = 10^{-8}$ used in ABELFUNCTIONS may be insufficient for finding a unique solution. All numerical computations are performed using native double precision so one must not set ϵ too close to 10^{-16} or else numerical round-off error may affect results as well.

- In practice it is observed that *only one* of the 2^{2g} half-lattice vectors $\boldsymbol{h}_j$ yields $\theta(\boldsymbol{K}(P_0), \Omega) = 0$. We have yet to find an example where the solution is not unique and expect that uniqueness can be shown mathematically.

Example 15. Finally, we provide examples of computing with the RCV.

```
In [14]: K = RiemannConstantVector   # alias the RCV function for brevity
         P0 = X.base_place()
         print K(P0)
```

```
Out[14]: [ 0.8488+0.7203j -0.5941-0.1146j -0.7432+0.8913j -0.8189+1.1381j]
```

We computationally verify that the RCV satisfies Theorems 11 and 13. First, we demonstrate that
$$2\,\boldsymbol{K}(P_0) + \boldsymbol{A}(P_0, \mathcal{C}) \equiv \boldsymbol{0}. \tag{44}$$

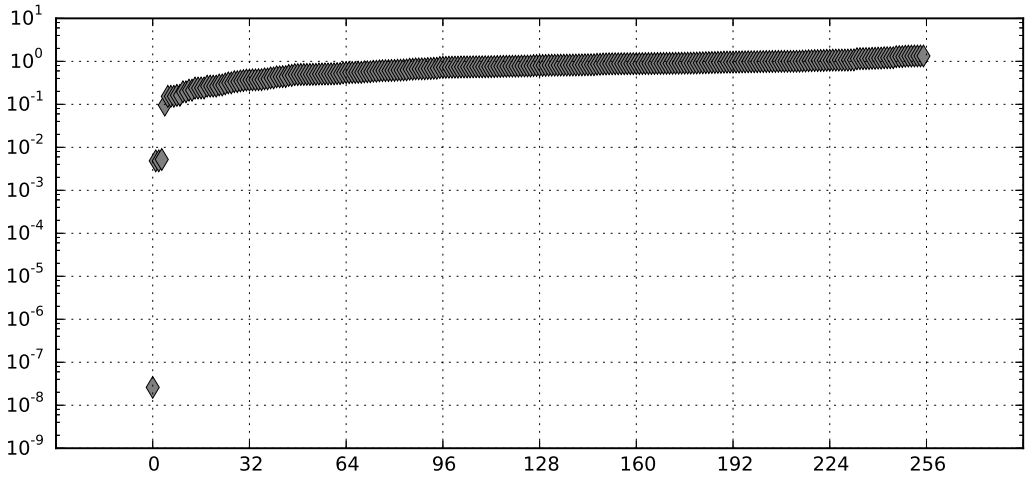


Figure 1: The sorted magnitudes of the oscillatory parts of $\theta\big(\boldsymbol{h}_j - \frac{1}{2}\boldsymbol{A}_0^{\mathcal{C}}, \Omega\big)$ for each of the 256 half-lattice vectors $\boldsymbol{h}_j$. Note that the half-lattice vector resulting in the correct RCV produces a theta value approximately five orders of magnitude closer to zero than the others.

In [15]:
```
z = J(2*K(P0) + AbelMap(C3))
print z
```

Out[15]:
```
[ 0.+0.j  0.+0.j -0.-0.j -0.-0.j]
```

Next, we verify that $\boldsymbol{K}(P_0)$ belongs to the theta divisor. To test for membership we factor the Riemann theta function into its exponential and oscillatory parts [9, 10],

$$\theta(z, \Omega) = e^u v. \tag{45}$$

Since the exponential part never vanishes we only examine the vanishing of the oscillatory part when determining $\boldsymbol{K}(P_0)$.

In [16]:
```
W = K(P0)
v = RiemannTheta.oscillatory_part(W,Omega)
print abs(v)
```

Out[16]:
```
2.60180216631e-08
```

Even with the default numerical integration and Riemann theta accuracy of 10^{-8} the choice of half-lattice vector producing the above RCV results in a Riemann theta value that

is several orders of magnitude closer to zero than with the incorrect choices of half-lattice vector, as shown in Figure 1.

Let $\mathcal{D}$ be the divisor consisting of the three places lying above $x = 2$ each of multiplicity one. $\mathcal{D}$ is an effictive divisor of degree $g - 1 = 3$. Therefore, $\boldsymbol{K}(P_0) + \boldsymbol{A}(P_0, \mathcal{D})$ is also a member of the theta divisor.

In [17]:
```
D = sum(places_above_two)
W = J(AbelMap(D) + K(P0))
v = RiemannTheta.oscillatory_part(W, Omega)
print abs(v)
```

Out[17]: `1.09506634962e-10`

4 Acknowledgements

This work was generously supported by the National Science Foundation under grant NSF-DMS-1008001. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the funding sources.

References

[1] M. J. Ablowitz and P. A. Clarkson, *Solitons, nonlinear evolution equations and inverse scattering*, London Mathematical Society Lecture Note Series, vol. 149, Cambridge University Press, Cambridge, 1991. MR 1149378 (93g:35108)

[2] M. J. Ablowitz and H. Segur, *Solitons and the inverse scattering transform*, SIAM Studies in Applied Mathematics, vol. 4, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, Pa., 1981. MR 642018 (84a:35251)

[3] H. F. Baker, *Abelian functions*, Cambridge Mathematical Library, Cambridge University Press, Cambridge, 1995, Abel's theorem and the allied theory of theta functions, Reprint of the 1897 original, With a foreword by Igor Krichever. MR 1386644 (97b:14038)

[4] E. Belokolos, A. Bobenko, V. Enol'skii, A. Its, and V. Matveev, *Algebro-geometric approach to nonlinear integrable problems*, Springer Series in Nonlinear Dynamics, Springer Berlin, Heidelberg, 1994.

[5] G. A. Bliss, *Algebraic functions*, Dover Publications, Inc., New York, 1966. MR 0203007 (34 #2866)

[6] A. I. Bobenko, *Introduction to compact Riemann surfaces*, Computational approach to Riemann surfaces, Lecture Notes in Math., vol. 2013, Springer, Heidelberg, 2011, pp. 3–64. MR 2905610

[7] A. I. Bobenko and L. A. Bordag, *Periodic multiphase solutions of the Kadomsev-Petviashvili equation*, J. Phys. A **22** (1989), no. 9, 1259–1274. MR 994366 (90i:35228)

[8] E. Brieskorn and H. Knörrer, *Plane algebraic curves*, Birkhäuser Basel, 1986.

[9] B. Deconinck, *Multidimensional theta functions*, NIST handbook of mathematical functions, U.S. Dept. Commerce, Washington, DC, 2010, pp. 537–547. MR 2655361

[10] B. Deconinck, M. Heil, A. Bobenko, M. van Hoeij, and M. Schmies, *Computing Riemann theta functions*, Math. Comp. **73** (2004), no. 247, 1417–1442. MR 2047094 (2005c:65015)

[11] B. Deconinck and M. S. Patterson, *Computing the Abel map*, Phys. D **237** (2008), no. 24, 3214–3232. MR 2477016 (2010d:37139)

[12] ———, *Computing with plane algebraic curves and Riemann surfaces: the algorithms of the Maple package "algcurves"*, Computational approach to Riemann surfaces, Lecture Notes in Math., vol. 2013, Springer, Heidelberg, 2011, pp. 67–123. MR 2905611

[13] B. Deconinck and H. Segur, *The KP equation with quasiperiodic initial data*, Phys. D **123** (1998), no. 1-4, 123–152, Nonlinear waves and solitons in physical systems (Los Alamos, NM, 1997). MR 1664932 (99m:35203)

[14] B. Deconinck and M. van Hoeij, *Computing Riemann matrices of algebraic curves*, Phys. D **152/153** (2001), 28–46, Advances in nonlinear mathematics and science. MR 1837895 (2002j:30001)

[15] B. A. Dubrovin, *Theta-functions and nonlinear equations*, Uspekhi Mat. Nauk **36** (1981), no. 2(218), 11–80, With an appendix by I. M. Krichever. MR 616797 (83i:35149)

[16] B. A. Dubrovin, R. Flickinger, and H. Segur, *Three-phase solutions of the Kadomtsev-Petviashvili equation*, Stud. Appl. Math. **99** (1997), no. 2, 137–203. MR 1458597 (98m:35179)

[17] D. Duval, *Rational Puiseux expansions*, Compositio Math. **70** (1989), no. 2, 119–154. MR 996324 (90c:14001)

[18] H. M. Farkas and I. Kra, *Riemann surfaces*, second ed., Graduate Texts in Mathematics, vol. 71, Springer-Verlag, New York, 1992. MR 1139765 (93a:30047)

[19] A. P. Fordy, *Soliton theory: a brief synopsis*, Soliton theory: a survey of results, Nonlinear Sci. Theory Appl., Manchester Univ. Press, Manchester, 1990, pp. 3–22. MR 1090584

[20] J. Frauendiener and C. Klein, *Hyperelliptic theta-functions and spectral methods: KdV and KP solutions*, Lett. Math. Phys. **76** (2006), no. 2-3, 249–267. MR 2238720 (2007f:14027)

[21] ———, *Computational Approach to Hyperelliptic Riemann Surfaces*, Lett. Math. Phys. **105** (2015), no. 3, 379–400. MR 3312511

[22] P. A. Griffiths, *Introduction to algebraic curves*, Translations of Mathematical Monographs, vol. 76, American Mathematical Society, Providence, RI, 1989, Translated from the Chinese by Kuniko Weltin. MR 1013999 (90i:14028)

[23] J. W. Helton and V. Vinnikov, *Linear matrix inequality representation of sets*, Comm. Pure Appl. Math. **60** (2007), no. 5, 654–674. MR 2292953 (2009a:93050)

[24] M. Mňuk, *An algebraic approach to computing adjoint curves*, J. Symbolic Comput. **23** (1997), no. 2-3, 229–240, Parametric algebraic curves and applications (Albuquerque, NM, 1995). MR 1448696 (98f:14049)

[25] M. Noether, *Rationale ausführungen der operationen in der theorie der algebraischen funktionen*, Math. Ann. **23** (1983), 311–358.

[26] D. Plaumann, B. Sturmfels, and C. Vinzant, *Quartic curves and their bitangents*, J. Symbolic Comput. **46** (2011), no. 6, 712–733. MR 2781949 (2012e:14065)

[27] ———, *Computing linear matrix representations of Helton-Vinnikov curves*, Mathematical methods in systems, optimization, and control, Oper. Theory Adv. Appl., vol. 222, Birkhäuser/Springer Basel AG, Basel, 2012, pp. 259–277. MR 2962788

[28] M. Pocchiola and G. Vegter, *The visibility complex*, Internat. J. Comput. Geom. Appl. **6** (1996), no. 3, 279–308, ACM Symposium on Computational Geometry (San Diego, CA, 1993). MR 1409648 (97i:68205)

[29] A. Poteaux and M. Rybowicz, *Towards a symbolic-numeric method to compute Puiseux series: the modular part*, arXiv preprint arXiv:0803.3027 (2008).

[30] H. Segur and A. Finkel, *An analytical model of periodic waves in shallow water*, Stud. Appl. Math. **73** (1985), no. 3, 183–220. MR 818729 (87g:76026)

[31] T. Shiota, *Characterization of Jacobian varieties in terms of soliton equations*, Invent. Math. **83** (1986), no. 2, 333–382. MR 818357 (87j:14047)

[32] G. Springer, *Introduction to Riemann surfaces*, Addison-Wesley Publishing Company, Inc., Reading, Mass., 1957. MR 0092855 (19,1169g)

[33] C. Swierczewski, *Abelfunctions: A library for computing with Abelian functions, Riemann surfaces, and algebraic curves.*, 2015, http://abelfunctions.cswiercz.info.

[34] C. Swierczewski and B. Deconinck, *Computing Riemann theta functions in Sage with applications*, Mathematics and Computers in Simulation (2013), http://www.sciencedirect.com/science/article/pii/S0378475413000888.

[35] C. L. Tretkoff and M. D. Tretkoff, *Combinatorial group theory, Riemann surfaces and differential equations*, Contributions to group theory, Contemp. Math., vol. 33, Amer. Math. Soc., Providence, RI, 1984, pp. 467–519. MR 767125 (86g:30055)

[36] M. van Hoeij, *An algorithm for computing an integral basis in an algebraic function field*, J. Symbolic Comput. **18** (1994), no. 4, 353–363. MR 1324494 (96e:11166)