Available online at www.sciencedirect.com

SciVerse ScienceDirect

Mathematics and Computers in Simulation 127 (2016) 263–272

MATHEMATICS AND COMPUTERS IN SIMULATION

www.elsevier.com/locate/matcom

Original article

Computing Riemann theta functions in Sage with applications

Christopher Swierczewski *, Bernard Deconinck

Department of Applied Mathematics, University of Washington, Seattle, WA 98195-3925, United States

Received 17 November 2011; received in revised form 27 September 2012; accepted 28 April 2013
Available online 16 May 2013

Abstract

A new implementation for the computation of the Riemann theta function in the open-source mathematical software Sage is discussed. This implementation is used in two applications. The first is the computation of three-phase solutions of the Kadomtsev–Petviashvili equation using an algorithm due to Dubrovin, originally implemented by Dubrovin et al. Our implementation is significantly easier, due to our more straightforward computation of the theta function. The second application is that of the computation of the bitangents of a quartic plane algebraic curve, relevant in convex optimization. Since Sage currently lacks the tools for computing with Riemann surfaces, this second application relies partially on results obtained using Maple's `algcurves` package. The current manuscript is the first step towards porting the functionality of the `algcurves` package to Sage as well as other scientific Python distributions.

Keywords: Riemann theta functions; Sage; Algebraic curves; Integrable equations; Bitangent lines

1. Introduction

The Riemann theta function is the fundamental building block for the Abelian functions [3]. These are functions in g complex variables, which are meromorphic and doubly periodic in each variable. As such, the Abelian functions generalize the classical elliptic functions to multiple variables. Like the elliptic functions, Abelian functions arise in a plethora of applications: all integrable equations have solutions that are expressed in terms of Abelian functions, the so-called algebro-geometric solutions [4]. Riemann theta functions can be associated with Riemann surfaces [15] and as such they are important for different optimization and cryptography applications which rely on algebraic curves, see e.g. [20].

Until recently, the Riemann theta function could not be computed efficiently. In 1997, Dubrovin, Flickinger and Segur [12], while computing solutions of the Kadomtsev–Petviashvili (KP) equation (see below), outlined how Riemann theta functions of genus no more than 3 can be computed, using different representations for different values of the parameters describing the theta function. The number of representations used grows exponentially with the number of variables g. As such, the approach used in [12] does not allow for an efficient numerical implementation for increasing g. In 2004, Deconinck et al. [7] described a new algorithm for the computation of the Riemann theta function, using a single representation for all values parameterizing the function. The essence of this algorithm is briefly recalled below. The computation of the Riemann theta function in Maple uses an implementation of this algorithm by the second

* Corresponding author. Tel.: +1 253 223 3721.
E-mail addresses: cswiercz@uw.edu (C. Swierczewski), bernard@uw.edu (B. Deconinck).

0378-4754/$36.00
http://dx.doi.org/10.1016/j.matcom.2013.04.018

author and Mark van Hoeij. To our knowledge this algorithm is also the one used in Mathematica. Here we report on an implementation by the first author in the open-source mathematical software Sage [23]. This presents the first step towards the larger goal of porting the functionality of Maple's `algcurves` package [8] to Sage and other common Python-based scientific computing packages.

In addition we present two applications of the Riemann theta function using this Sage implementation. First, we duplicate the results of Dubrovin, Flickinger and Segur [12] for the computation of three-phase solutions of the KP equation. Our implementation of their algorithm is much more concise, due to the use of the results of [7] for the computation of the Riemann theta function. Second, we computationally realize some classical results about the bitangents (tangents at two distinct points) of quartic plane algebraic curves. Using our Sage implementation of the Riemann theta function, combined with some results obtained using Maple's `algcurves` package (since Sage does not yet include its full capabilities), we compute all bitangents of such curves in terms of the Riemann theta function determined by the Riemann surface associated with the quartic curve. Other applications of the Riemann theta function abound, such as in convex optimization, number theory and other areas [3,20]. We will delay their computational investigation until the complete `algcurves` package has been ported.

2. Riemann theta functions

In this section we summarize the basic facts about the Riemann theta function. We mention the challenges in computing the Riemann theta function, a solution of the computational problem due to Deconinck et al. [7] and we discuss a recent implementation in Sage due to the first author.

Definition 1. A matrix $\Omega \in \mathbb{C}^{g\times g}$ is called a **Riemann matrix** if it is symmetric and its imaginary part $\mathrm{Im}(\Omega)$ is positive definite.

The set of Riemann matrices is denoted $\mathcal{H}_g$ and is referred to as the Siegel upper half space [15]. Different conventions are used by different authors, corresponding to their different conventions for the definition of the Riemann theta function below.

Definition 2. The **Riemann theta function** $\theta : \mathbb{C}^g \times \mathcal{H}_g \to \mathbb{C}$ is defined in terms of its Fourier series

$$\theta(z|\Omega) = \sum_{n\in\mathbb{Z}^g} e^{2\pi i((1/2)n\cdot\Omega n + n\cdot z)}.$$

The Riemann theta function converges absolutely and uniformly on compact sets in $\mathbb{C}^g \times \mathcal{H}_g$. It is periodic in z with integer periods and quasi-periodic in z in the columns of Ω. In other words, if $m, n \in \mathbb{Z}^g$ then

$$\theta(z + m + \Omega n|\Omega) = e^{-2\pi i((1/2)n\cdot\Omega n + n\cdot z)}\theta(z|\Omega). \tag{1}$$

A generalization of the Riemann theta function, involving a non-integer shift in some of its arguments, is referred to as the Riemann theta function with characteristics.

Definition 3. Let $\alpha, \beta \in [0, 1)^g$. The **Riemann theta function with characteristic** $\begin{bmatrix}\alpha\\ \beta\end{bmatrix}$ is defined as

$$\begin{aligned}\theta\begin{bmatrix}\alpha\\ \beta\end{bmatrix}(z|\Omega) &= \sum_{n\in\mathbb{Z}^g} e^{2\pi i((1/2)(n+\alpha)\cdot\Omega(n+\alpha)+(n+\alpha)\cdot(z+\beta))}\\ &= e^{2\pi i((1/2)\alpha\cdot\Omega\alpha+\alpha\cdot(z+\beta))}\theta(z+\Omega\alpha+\beta|\Omega).\end{aligned}$$

Note that $\theta\begin{bmatrix}0\\ 0\end{bmatrix}(z,\Omega) = \theta(z,\Omega)$. See [15–17] for further definitions and properties of the Riemann theta function.

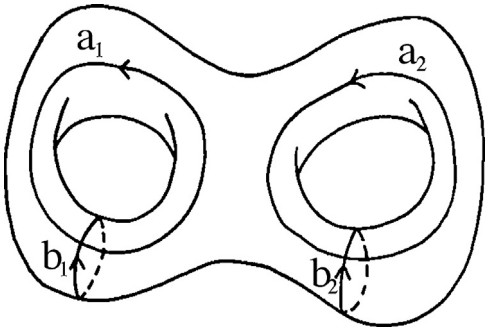

Fig. 1. A cartoon of a genus 2 Riemann surface, with a homology basis indicated.

3. Algebraic curves, Riemann surfaces, and Riemann matrices

Riemann matrices, and thus Riemann theta functions, are naturally associated with compact Riemann surfaces. It appears that only Riemann theta functions originating from Riemann surfaces arise in applications. In this section we outline briefly the details of how plane algebraic curves give rise to Riemann surfaces, and how Riemann surfaces give rise to Riemann theta functions. The reader can find more detail in standard references such as [11,15]. Deconinck and Van Hoeij [9] present a description specifically aimed at applied mathematicians. The main goal of this section is to recall some concepts for the benefit of the reader and to establish some notation, which is used in the remainder of the paper.

For our purposes, a plane algebraic curve $\mathcal{V}_{\mathbb{C}}(f)$ is the zero set of a polynomial $f(x, y)$ in two variables $x, y \in \mathbb{C}$ with complex coefficients $\{a_{jk} \in \mathbb{C},\ j + k \leq N\}$. It is determined by an equation of the form

$$f(x, y) = \sum_{j,k} a_{jk} x^j y^k = 0. \tag{2}$$

Here N is the degree of the algebraic curve. Typically such algebraic curves have singular points, i.e., (x, y) for which $f(x, y) = 0, f_x(x, y) = 0 = f_y(x, y)$. A plane algebraic curve also has "points at infinity", which are found by homogenizing the curve by considering $z^N f(x/z, y/z) = 0$ and equating $z = 0$ (we have assumed that $\{a_{jk}, j+k=N\} \neq \emptyset$). If we desingularize and compactify $\mathcal{V}_{\mathbb{C}}(f)$, we obtain a compact Riemann surface, which we may conveniently represent by $\mathcal{V}_{\mathbb{C}}(f)$ as well. The desingularizing can be done, for instance, by imposing analyticity of the different branches of $\mathcal{V}_{\mathbb{C}}(f)$ at the singular points, for instance by using Puiseux series representations for these branches [9]. Compactification is as straightforward as incorporating the points at infinity by using homogenous triples $(x : y : z)$ to describe $\mathcal{V}_{\mathbb{C}}(f)$. It should be noted that *all* compact Riemann surfaces can be represented as plane algebraic curves. In what follows Riemann surfaces are assumed to be compact and we move freely between the Riemann surface and its representing plane algebraic curve.

A compact Riemann surface has one topological invariant, its genus g. The genus is determined by the degree and the branching structure of $\mathcal{V}_{\mathbb{C}}(f)$, as well as by its singularities [1]. On a genus g Riemann surface there exist exactly g linearly independent holomorphic differentials $\omega_j, j = 1, \ldots, g$, i.e., differentials that are analytic in the neighborhood of *any* place on the Riemann surface, including those places that correspond to points at infinity or singular points of the plane algebraic curve. This is a consequence of the Riemann–Roch Theorem [11,15]. For the purposes of integrating these differentials on $\mathcal{V}_{\mathbb{C}}(f)$, there exists a basis of $2g$ closed paths $\{a_j, b_j, j = 1, \ldots, g\}$ on the Riemann surface, by Cauchy's Theorem. These paths cannot be contracted to a point. They form a basis in the sense that the integral of any holomorphic differential around any closed path on $\mathcal{V}_{\mathbb{C}}(f)$ can be written as a linear combination with integer coefficients of integrals of this differential around the cycles $\{a_j, b_j, j = 1, \ldots, g\}$. The basis cycles a_j and $b_j, j = 1, \ldots, g$ are chosen to have certain intersection properties [15]. A cartoon for $g = 2$ is displayed in Fig. 1. The values of $\oint_{a_k} \omega_j$ may be used to normalize the differentials so that $\oint_{a_k} \omega_j = \delta_{jk}$, where δ_{jk} is the Kronecker delta: $\delta_{jk} = 1$ if $j = k$ and 0

otherwise. In what follows we assume that the differentials ω_j are so normalized. Riemann showed [11,15] that the $g \times g$ matrix Ω obtained via

$$\oint_{b_k} \omega_j = \Omega_{jk}$$

is a Riemann matrix. It should be mentioned that not all Riemann matrices are associated with Riemann surfaces this way. In fact, the number of independent parameters characterizing a $g \times g$ Riemann matrix is $g(g+1)/2$, whereas the number of such parameters characterizing a $g \times g$ Riemann matrix that originates from a Riemann surface is $3g-3$. The Schottky problem is the problem of determining which Riemann matrices are associated with Riemann surfaces [11]. Note that for $g \leq 3$ the number of free parameters in a Riemann matrix is not restricted by its association with a Riemann surface. In other words, the Schottky problem does not present an obstacle when dealing with Riemann matrices of genus $\leq$3. All of them may be thought of as originating from a Riemann surface. In fact, for $g=3$, the equations determining this surface are constructed in [13]. In its full generality, the Schottky problem was effectively solved using the KP equation by Shiota [22].

4. Computation and implementation

A new implementation of the Riemann theta function is done in the open-source mathematics software package, Sage. Sage [23] is a free, open-source mathematics software system licensed under the GNU Public License. It combines other free, open-source mathematics software by packaging and unifying them through a common interface under the Python programming language. It is the result of a volunteer effort by an international team of students, professors, researchers, and software engineers from a variety of institutions.

Most of the underlying mathematical functionality of Sage is made possible through a collection of free, open-source C/C++and Fortran-based libraries. These include, but are not limited to, ATLAS and BLAS for linear algebra operations, GAP for computational discrete algebra and group theory, Maxima for symbolic computation, Pari/GP for computations in number theory, MPIR for multi-precision integer and rational arithmetic, along with many other core libraries.

Several pre-existing Python-based packages are included as part of the Sage library such as Numpy/Scipy for numerical calculations, Sympy for symbolic arithmetic, and matplotlib for two-dimensional plotting. Finally, Sage developers have written their own code, independent of pre-existing packages integrated into Sage. The code for computing Riemann theta functions is an example of such an implementation.

The two main challenges in computing the Riemann theta function are that it is defined in terms of an infinite series in $\mathbb{Z}^g$ and, as seen by the quasi-periodicity property, it grows doubly exponentially in the directions specified by the columns of Ω. Deconinck et al. [7] developed an algorithm that efficiently computes $\theta(z|\Omega)$ to a user-specified accuracy. The core of their technique is to factor out the exponential growth from the infinite sum, then determine an optimal number of integer points over which to approximate the infinite series up to a user-defined accuracy. It should be remarked that the algorithm does not require different series representations for the Riemann theta function depending on the entries of Ω, as was necessary in [12], for instance.

This algorithm is implemented in Sage by the first author and can be found in the `sage.functions.riemann_theta` module located in the Sage library. We present an example of a Sage calling sequence to compute the value of the Riemann theta function associated with the Riemann matrix

$$\Omega = \begin{pmatrix} i & -1/2 \\ -1/2 & i \end{pmatrix}.$$

We must first construct the matrix over a `ComplexField` object – a field of complex numbers with a prescribed precision. The desired number of bits of accuracy is given as an argument to `ComplexField`.

```
sage: R.<I> = ComplexField(32)   # I=sqrt(-1)
sage: Omega = Matrix(R,2,2,[I,-1/2,-1/2,I])
sage: Omega
[1.00000000*I -0.500000000]
[-0.500000000 1.00000000*I]
```

To evaluate the Riemann theta function at a point $z \in \mathbb{C}^g = \mathbb{C}^2$, provide a Python list or Sage vector representing z as well as the Riemann matrix as input.

```
sage: RiemannTheta([0,0], Omega)
1.16540106 - 6.14229246e-24*I
sage: v = vector(R,[1/2,1/3 +I/4])
sage: val = RiemannTheta(v, Omega)
sage: val
0.795738522 - 0.187073837*I
sage: parent(val)
Complex Field with 32 bits of precision
```

Note that the precision of the output is only up to the precision given by the Riemann matrix. We also compute directional derivatives of the Riemann theta function. This is done by providing a list of Python `list` objects as a third input. For example, $z = (z_1, z_2) = (1.7 - 1.2I, -0.4 + I/3)$ and Riemann matrix Ω be defined as above. We compute

$$\frac{\partial}{\partial z_1}\theta(z, \Omega) \quad \text{and} \quad \frac{\partial}{\partial z_2}\theta(z, \Omega).$$

```
sage: RiemannTheta([1.7-1.2*I,-0.4+I/3], Omega, [[1,0]])
554.338151 - 409.949386*I
sage: RiemannTheta([1.7-1.2*I,-0.4+I/3], Omega, [[0,1]])
-127.329415 +139.848367*I
```

We also compute $(\partial^2\theta/\partial z_1 \partial z_1)(z, \Omega)$.

```
sage: RiemannTheta([1.7-1.2*I,-0.4+I/3], Omega, [[1,0],[0,1]])
-817.237957 - 970.730267*I
```

For plotting purposes we use the oscillatory part of the Riemann theta function, defined in [7],

$$\hat{\theta}(z|\Omega) = e^{-\pi \mathrm{Im}(z)\cdot \mathrm{Im}(\Omega)^{-1}\mathrm{Im}(z)}\theta(z|\Omega).$$

That is, the oscillatory part of the Riemann theta function is the Riemann theta function with the doubly exponential growth factored out, thus leaving only bounded, oscillatory behavior. The `RiemannTheta.exp_and_osc_at_point()` method allows the user to separately compute these exponential and oscillatory parts. It returns a tuple (u, v) such that $\theta(z|\Omega) = e^u v$. Below, we create the function $g(x, y) = \hat{\theta}(x + iy, 0|\Omega)$ and plot its real part, imaginary part and absolute value. These plots are shown in Fig. 2.

```
sage: def g(x,y):
....:   return RiemannTheta.exp_and_osc_at_point([x+I*y,0], Omega)[1]
sage: plot3d(lambda x,y: real(g(x,y)), (0,1), (0,5), adaptive=True)
sage: plot3d(lambda x,y: imag(g(x,y)), (0,1), (0,5), adaptive=True)
sage: plot3d(lambda x,y: abs(g(x,y)), (0,1), (0,2), adaptive=True)
sage: from matplotlib import cm # for matching color scheme to 3D plots
sage: contour_plot(lambda x,y: real(g(x,y)), (0,1), (0,5), cmap=cm.hsv)
sage: contour_plot(lambda x,y: imag(g(x,y)), (0,1), (0,5), cmap=cm.hsv)
sage: contour_plot(lambda x,y: abs(g(x,y)), (0,1), (0,2), cmap=cm.hsv)
```

Additional examples of Sage code using `RiemannTheta()` can be found in a Sage Notebook worksheet at http://www.cswiercz.info/publications.html.

5. Applications

In this section we present applications of Riemann theta functions to integrable equations and bitangent lines of quartic curves.

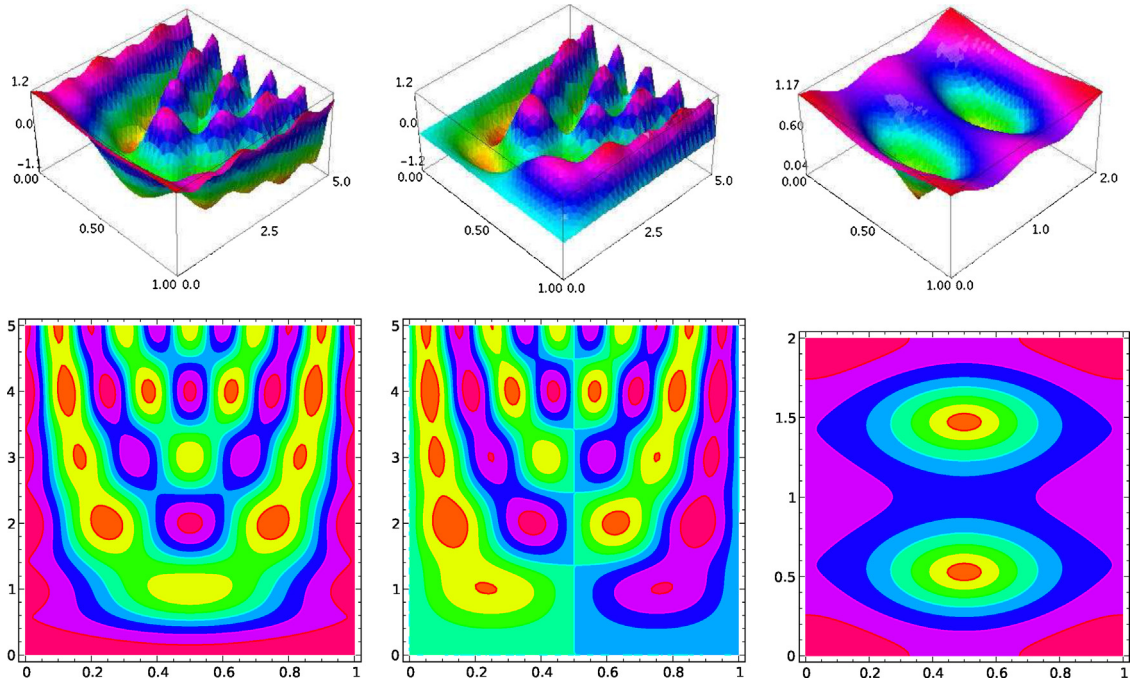

Fig. 2. Three-dimensional and contour plots of the real part, imaginary part, and absolute value of the oscillatory part $\hat{\theta}(x+iy, 0|\Omega)$ of the Riemann theta function $\theta(x+iy, 0|\Omega)$.

5.1. The Kadomtsev–Petviashvili equation

The Kadomtsev–Petviashvili (KP) equation is a well-studied and fundamental example of an integrable partial differential equation. It describes long waves in shallow water that are weakly two-dimensional, among other applications [2]. It is given by

$$(-4u_t + 6uu_x + u_{xxx})_x + 3u_{yy} = 0. \tag{3}$$

The equation admits a large family of (quasi-) periodic solutions of the form

$$u(x, y, t) = c + 2\frac{\partial^2}{\partial x^2} \log \theta(z|\Omega) \tag{4}$$

where Ω is a Riemann matrix obtained from an algebraic curve, c is a constant, and the phase variable $z = (z_1, \ldots, z_g)$ is given by

$$z = kx + ly + \omega t + \phi, \quad k, l, \omega, \phi \in \mathbb{C}^g,$$

and k, l, ω and ϕ are parameterized by the same Riemann surface that determines Ω, see [11,14]. Such a solution to KP is referred to as a *genus g solution*. Krichever [14] describes a method for generating these solutions starting with a Riemann surface and a divisor of degree g on this surface (i.e., a set of g points on the surface).

If the genus does not exceed 3 then the Schottky problem is not an issue, in that, all 2×2 and 3×3 Riemann matrices are derived from algebraic curves, as mentioned above. Therefore, without needing to use the theory of Riemann surfaces, the calculation of solutions to KP of the form in Eq. (4) is reduced to solving algebraic equations in the components of the vectors k, l and ω. The algorithm is described in [12]. Furthermore, in the genus $g = 3$ case the vector ϕ can be chosen to be real, but otherwise arbitrary.

Computing genus 3 solutions to KP is an example of an application of Riemann theta functions in Sage, and serves as additional verification that the Riemann theta function algorithm is correctly implemented. The code is too long to be shown here in its entirety. (Approximately 40 lines, not including the code to compute values of the Riemann theta function and its derivatives.) The difference between this implementation and that of Dubrovin et al. [12] is in

the computation of the Riemann theta function. They required several different representations for computing $\theta(z|\Omega)$ depending on where Ω lies in $\mathcal{H}_g$. With the algorithm of Deconinck et al. [7], special treatment for different Riemann matrices is unnecessary.

We present the major steps of the computation with tools particular to Sage. In order to match the convention of Dubrovin, Flickinger, and Segur for the definition of the Riemann theta function we make the transformation

$$z \mapsto \frac{1}{2\pi} z, \qquad \Omega \mapsto \frac{1}{2\pi i}\Omega.$$

Following [12] we introduce the following notation: given the characteristic $m \in \{0, 1/2\}^3$ and the parameter $k = (k_1, k_2, k_3)^T$,

$$\hat{\theta}[m] = \theta \begin{bmatrix} m \\ 0 \end{bmatrix} (z|\Omega)|_{z=0}, \qquad \hat{\theta}_{ij}[m] = \frac{\partial^2 \theta \begin{bmatrix} m \\ 0 \end{bmatrix} (z|\Omega)}{\partial z_i \partial z_j}|_{z=0},$$

$$\hat{\theta}_{ijkl}[m] = \frac{\partial^4 \theta \begin{bmatrix} m \\ 0 \end{bmatrix} (z|\Omega)}{\partial z_i \partial z_j \partial z_k \partial z_l}|_{z=0}, \qquad \partial_k^4 \hat{\theta}[m] = \sum_{1 \le i,j,k,l \le 3} k_i k_j k_k k_l \hat{\theta}_{ijkl}[m].$$

Choose characteristics $m_1, \ldots, m_7$ such that the 7×7 matrix

$$\begin{pmatrix} \hat{\theta}_{11}[m_1] & \cdots & \hat{\theta}_{33}[m_1] & \hat{\theta}[m_1] \\ \vdots & \ddots & \vdots & \vdots \\ \hat{\theta}_{11}[m_7] & \cdots & \hat{\theta}_{33}[m_7] & \hat{\theta}[m_7] \end{pmatrix}$$

is invertible. Denote the elements of the inverse matrix by

$$\begin{pmatrix} a_{m_1}^{11} & \cdots & a_{m_7}^{11} \\ \vdots & \ddots & \vdots \\ a_{m_1}^{33} & \cdots & a_{m_7}^{33} \\ a_{m_1} & \cdots & a_{m_7} \end{pmatrix}.$$

We define the polynomials $Q_{ij}(k)$ and $P_{ij}(k)$:

$$\begin{aligned} Q_{ij} &= -\sum_{m \in \{m_1, \ldots, m_7\}} a_m^{ij} \partial_k^4 \hat{\theta}[m], \\ P_{ij} &= \frac{1}{2}[k_i^2 Q_{ii}(k) - k_i k_j Q_{ij}(k) + k_j^2 Q_{jj}(k)]. \end{aligned}$$

With these definitions, we describe the algorithm for generating genus $g = 3$ solutions to the KP equation.

1. Construct a 3×3 Riemann matrix over `ComplexField`.
2. Compute the quantities $\partial_k^4 \hat{\theta}[m]$ over all half-period characteristics $m \in \{0, 1/2\}^3$ with symbolic variables k_1, k_2, k_3 as coefficients. Note that in the notation of [12] $\alpha = m$ and $\beta = 0$.
3. Compute the quantities $\hat{\theta}_{ij}[m]$ over all half-period characteristics $m \in \{0, 1/2\}^3$.
4. Construct and invert a 7×7 matrix with entries of the form $\hat{\theta}_{ij}[m]$.
5. Compute the polynomials $Q_{ij}(k)$ and $P_{ij}(k)$ and substitute values for k_1 and k_2. Solve for and choose a particular value of k_3 using the compatibility condition H.11 in [12] and substitute into Q_{ij} and P_{ij} as well.
6. Write out the list of equations in Eqs. (H.9) and (H.10) of [12], substitute a value for l_1, and solve for the remaining unknowns l_2, l_3 and ω using a numerical root finder in the Numpy/Scipy numerics package included with Sage.

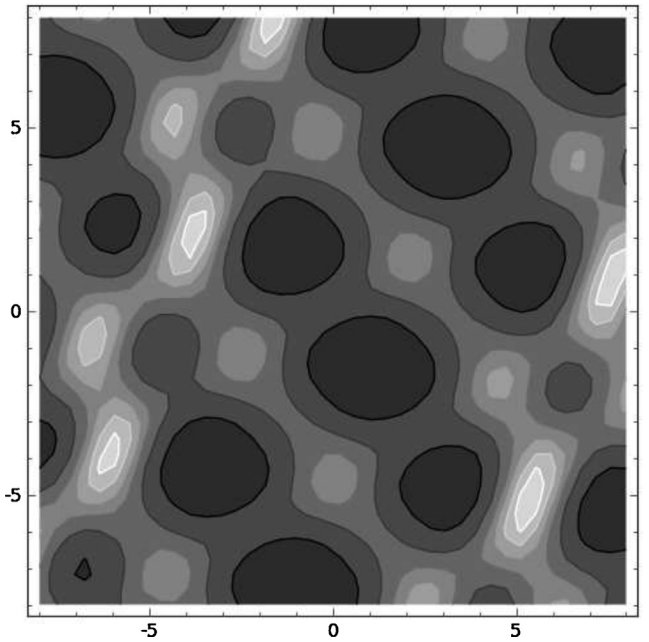

Fig. 3. A contour plot of a genus 3 solution to the KP equation. The x- and y-axes are the horizontal and vertical axes, respectively. Here, the Riemann matrix is that given in (5). The phase vectors are $k=(0.5, 1, 1.2060)^T$, $l=(-0.2, -1.3974, 0.6148)^T$, $\omega=(-1.1427, -6.2228, -0.3940)^T$, and $\phi=(0, 0, 0)^T$. Lastly, $t=0.5$.

An example contour plot of a genus 3 solution to the KP equation corresponding to the Riemann matrix

$$\Omega = \begin{pmatrix} 2 & 1 & 1 \\ 1 & 4.5 & 0.9 \\ 1 & 0.9 & 4.54 \end{pmatrix} \tag{5}$$

is shown in Fig. 3. The horizontal and vertical axes are the x- and y-coordinates, respectively, and the color indicates surface wave height. This solution matches that found in Fig. 4 of [12].

5.2. *Bitangents of plane quartic curves*

The computation of bitangent lines is useful for computations in optimization-related fields such as algebraic geometry and convex optimization. In algebraic geometry, bitangents can be used to represent smooth complex projective plane quartic curves as both a symmetric determinant of a linear form (or, determinantal representation) and a sum of three squares [18]. In convex optimization, bitangents are used to construct a "visibility complex" which, in turn, is used to solve the shortest path problem in Euclidean space [20].

Definition 4. A **bitangent** to a curve $\mathcal{V}_{\mathbb{C}}(f)$ determined by $f(x, y)=0$ (see (2)) is a line $\mathcal{L}$ that lies tangent to $\mathcal{V}_{\mathbb{C}}(f)$ at least two distinct points.

By Bezout's Theorem, if a curve has a bitangent it necessarily must be of degree at least four [5]. A result of Plücker determines that a degree four complex curve admits exactly 28 complex bitangents [19]. In particular, Plücker showed that the number of real bitangents of any quartic must be 28, 16, or fewer than 9. The connection between Riemann theta functions and the bitangent lines of smooth quartics was known to Riemann [21]. Note that the genus of the Riemann surface corresponding to a nonsingular curve $\mathcal{V}_{\mathbb{C}}(f)$ of degree 4 is exactly 3 [1]. Thus the Riemann theta function associated with $\mathcal{V}_{\mathbb{C}}(f)$ depends on $z=(z_1, z_2, z_3)^T$ and a 3×3 Riemann matrix Ω. Further, there are three

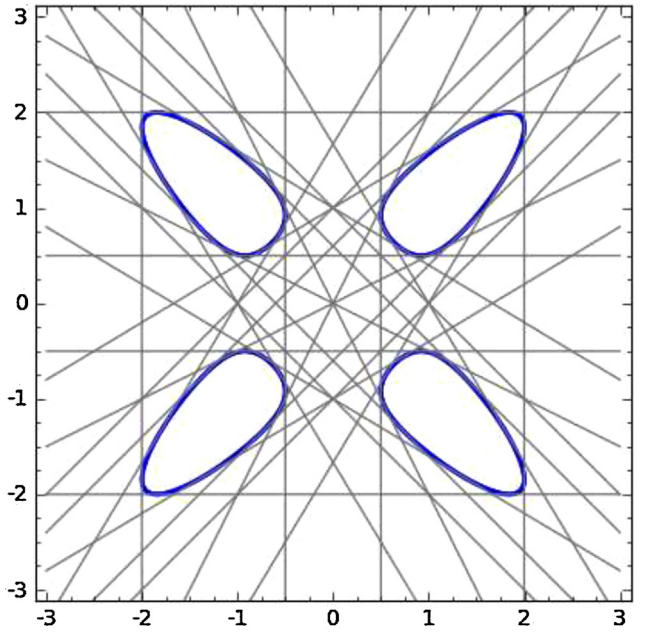

Fig. 4. The real graph of the Edge Quartic $f(x, y) = 25(x^4 + y^4 + 1) - 34(x^2y^2 + x^2 + y^2) = 0$ and its 28 real bitangents. Note that four of them lie tangent to $\mathcal{V}_{\mathbb{C}}(f)$ at infinity.

linearly independent normalized holomorphic differentials ω_j, $j = 1, 2, 3$ on $\mathcal{V}_{\mathbb{C}}(f)$, see Section 3. Using the Newton Polygon method [6] these are of the form

$$\omega_j = \frac{A_j(x, y)}{\partial f/\partial y} dx, \tag{6}$$

with $A_j(x, y)$ a polynomial of degree one. The following theorem concretely gives an equation for the bitangents in terms of the Riemann theta function and these holomorphic differentials [3,10].

Theorem. *The bitangents of a smooth plane quartic curve* $\mathcal{V}_{\mathbb{C}}(f)$ *with corresponding Riemann theta function* $\theta(z_1, z_2, z_3|\Omega)$ *and normalized holomorphic differentials are determined by*

$$\sum_{j=1}^{3} \frac{\partial\theta \begin{bmatrix} \alpha \\ \beta \end{bmatrix} (z_1, z_2, z_3|\Omega)}{\partial z_j}\Bigg|_{(z_1,z_2,z_3)=(0,0,0)} \omega_j = 0, \tag{7}$$

for all odd characteristics $[\alpha, \beta]$.

Equivalently, using (6), the bitangent lines can be written as

$$\sum_{j=1}^{3} \frac{\partial\theta \begin{bmatrix} \alpha \\ \beta \end{bmatrix} (z_1, z_2, z_3|\Omega)}{\partial z_j}\Bigg|_{(z_1,z_2,z_3)=(0,0,0)} A_j(x, y) = 0,$$

for all odd characteristics $[\alpha, \beta]$. It is this form of the bitangents that is used to produce Fig. 4, which displays an example computation where we compute the 28 real bitangents of the Edge quartic, defined by

$$25(x^4 + y^4 + 1) - 34(x^2y^2 + x^2 + y^2) = 0.$$

Some missing functionality prevents us from computing bitangents entirely within Sage. Primarily, the computation of a Riemann matrix from an algebraic curve has yet to be implemented. Doing so requires computing monodromy

groups, homology, cohomology bases, etc. However, Sage does provide an interface to Maple where different parts of the `algcurves` package, written by Deconinck, van Hoeij, and Patterson [8] can be used. Thus, given a Riemann matrix, we can, in fact, calculate (7).

Acknowledgements

The authors acknowledge support from the National Science Foundation under grant NSF-DMS-1008001. Sage development by the first author was made possible by grant NSF-DMS-0821725 from the National Science Foundation. 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] S.S. Abhyankar, Algebraic Geometry for Scientists and Engineers, vol. 35 of Mathematical Surveys and Monographs, American Mathematical Society, Providence, RI, 1990.
[2] M.J. Ablowitz, H. Segur, Solitons and the Inverse Scattering Transform, vol. 4 of SIAM Studies in Applied Mathematics, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1981.
[3] H.F. Baker, Abelian Functions: Abel's Theorem and the Allied Theory of Theta Functions, Cambridge University Press, Cambridge, 1897.
[4] E.D. Belokolos, A.I. Bobenko, V.Z. Enol'skii, A.R. Its, V.B. Matveev, Algebro-geometric approach to nonlinear integrable problems, in: Springer Series in Nonlinear Dynamics, Springer-Verlag, Berlin, 1994.
[5] E. Bézout, Théorie générale des équations algébriques, University of Lausanne, 1779 (Ph.D. Pierres).
[6] G.A. Bliss, Algebraic Functions, Dover Publications, Inc., New York, 1966.
[7] B. Deconinck, M. Heil, A. Bobenko, M. van Hoeij, M. Schmies, Computing Riemann theta functions, Mathematics of Computation 73 (247) (2003) 1417–1442.
[8] B. Deconinck, M. Patterson, Computing with plane algebraic curves and Riemann surfaces: the algorithms of the Maple package Algcurves, in: Computational Approach to Riemann Surfaces, Lecture Notes in Mathematics, Springer Berlin, Heidelberg, 2011.
[9] B. Deconinck, M. van Hoeij, Computing Riemann matrices of algebraic curves, Physica D 152 (153) (2001) 28–46.
[10] I. Dolgachev, Topics in Classical Algebraic Geometry, Lecture Notes, 2013, posted at http://www.math.lsa.umich.edu/idolga
[11] B. Dubrovin, Theta functions and non-linear equations, Russian Mathematical Surveys 36 (2) (1981) 11–92.
[12] B. Dubrovin, R. Flickinger, H. Segur, Three-phase solutions fo the Kadomtsev–Petviashvili equation, Studies in Applied Mathematics 99 (1997) 137–203.
[13] J. Guàrdia, On the Torelli problem and Jacobian Nullwerte in genus three, Michigan Mathematical Journal 60 (1) (2011) 51–65.
[14] I.M. Krichever, Integration of nonlinear equations by the methods of algebraic geometry, Functional Analysis and its Applications 11 (1) (1977) 12–26.
[15] D. Mumford, Tata Lectures on Theta I, Modern Birkhäuser Classics, Birkhäuser, Boston, MA, 1983.
[16] D. Mumford, Tata Lectures on Theta II, Modern Birkhäuser Classics, Birkhäuser, Boston, MA, 1983.
[17] National Institute of Standards and Technology, Digital Library of Mathematical Functions, 2011 http://dlmf.nist.gov/21
[18] D. Plaumann, B. Sturmfels, C. Vinzant, Quartic curves and their bitangents, Journal of Symbolic Computation 46 (2011) 712–733.
[19] J. Plücker, Solution d'une question fondamentale concernant theorie generale des courbes, Journal fur Die Reine und Angewandte Mathematik 12 (1834) 105–108.
[20] M. Pocchiola, G. Vegter, The visibility complex, in: SCG'93 Proceedings of the ninth annual symposium on Computational gemoetry, 1993, pp. 328–337.
[21] B. Riemann, Zur theorie der Abelschen funktionen für den fall $p = 3$ (1876) 466–472.
[22] T. Shiota, Characterization of Jacobian varieties in terms of soliton equations, Inventiones Mahtematicae 83 (2) (1986) 333–382.
[23] W.A. Stein, et al., Sage Mathematics Software (Version 4.6.2), The Sage Development Team, 2011 http://www.sagemath.org