Computational Methods and Function Theory
Volume 00 (0000), No. 0, 000–000 XXYYYZZ

Efficient computation of the branching structure of an algebraic curve

J. Frauendiener, C. Klein, and V. Shramchenko

Abstract. An efficient algorithm for computing the branching structure of a compact Riemann surface defined via an algebraic curve is presented. Generators of the fundamental group of the base of the ramified covering punctured at the discriminant points of the curve are constructed via a minimal spanning tree of the discriminant points. This leads to paths of minimal length between the points, which is important for a later stage where these paths are used as integration contours to compute periods of the surface. The branching structure of the surface is obtained by analytically continuing the roots of the equation defining the algebraic curve along the constructed generators of the fundamental group.

Keywords. Riemann surfaces, algebraic curves, monodromies, fundamental group.

2000 MSC.

1. Introduction

Riemann surfaces have many applications in natural sciences and engineering, for instance in the solutions of certain integrable partial differential equations (PDE) appearing in hydrodynamics and optics, see e.g. [2]. For a long time the full potential of related techniques could not be realized due to the absence of efficient numerical approaches. In [5], the Maple package *algcurves* (starting with Maple 7) for algebraic curves which gives a mixed symbolic-numeric approach was published, see also [6]. Since all compact Riemann surfaces can be defined via non-singular plane algebraic curves (see e.g. [16]), all quantities characterizing a Riemann surface can in principle be computed along these lines. For a different numerical approach to Riemann surfaces based on Schottky uniformizations see [3, 15].

Version August 11, 2011.
This work has been supported in part by the project FroM-PDE funded by the European Research Council through the Advanced Investigator Grant Scheme, the Conseil Régional de Bourgogne via a FABER grant and the ANR via the program ANR-09-BLAN-0117-01. JF and VS thank for the hospitality at the IMB as visiting professors, where part of the work was written. The work of VS was supported by NSERC .

ISSN 1617-9447/$ 2.50

Though being very useful, the mixed symbolic-numeric approach has the disadvantage that only algebraic curves with exact arithmetic coefficients, i.e., not floating point coefficients can be used, and that the performance of the numerics is consequently reduced in addition to the limitations imposed by the restriction to exact arithmetic expressions. Thus a fully numeric approach was presented in [7, 8] for real hyperelliptic curves and in [9] for general algebraic curves. The gain in performance and in flexibility allows the study of higher genus curves and of families of curves, i.e., of the modular properties of Riemann surfaces. Such modular dependences are important for instance in the study of solutions to certain integrable PDE appearing in the context of gravity and surface theory [12] and the description of the asymptotic behavior of highly oscillatory regimes in dispersive PDE [13, 10] as well as the study of modular invariants discussed in topological field theories, see for instance [14, 11].

A plane algebraic curve C is defined as a subset in $\mathbb{C}^2$, $C = \{(x, y) \in \mathbb{C}^2 | f(x, y) = 0\}$, where $f(x, y)$ is an irreducible polynomial in x and y,

$$f(x,y) = \sum_{i=1}^{M} \sum_{j=1}^{N} a_{ij} x^i y^j = \sum_{j=1}^{N} a_j(x) y^j = 0 \, . \tag{1}$$

We assume that not all a_{iN} vanish and that N is thus the degree of the polynomial in y. At a generic point x there are N distinct roots $y^{(k)}$, $k = 1, \ldots, N$, which implies that the algebraic curve defines an N-sheeted ramified covering of the x-plane. The surface is then compactified in a standard way (see for example [1]) so that we have a ramified covering of the x-sphere $\mathbb{CP}^1$. At a point where both $f(x, y) = 0$ and $f_y(x, y) = 0$, the number of distinct roots $y^{(k)}$ is lower than N, i.e., this *branch point* belongs to several sheets of the covering. To describe the associated Riemann surface, one has to be able to identify the branching structure of the curve at the branch points, in other words, one has to specify which sheets of the covering are connected in which way at a given branch point. This is equivalent to identifying the *monodromy* of the surface. It is the purpose of this paper to give an efficient algorithm for this crucial step in the numerical treatment of Riemann surfaces. The structure of the Riemann surface obviously does not depend on whether the algebraic curve (1) is studied as a covering of the x- or of the y-sphere. We will concentrate here on the covering of the x-sphere since this covering appears as the input data in many applications such as algebro-geometric solutions to certain integrable equations. Notice that the inverse problem, to find the equation of an algebraic curve for a given monodromy, is very involved and could so far be only addressed for low genus, see e.g. [4] and references therein.

The points on an algebraic curve (1) with $f(x, y) = f_y(x, y) = 0$ can be computed in a standard way as the zeros of the discriminant or resultant of $f(x, y)$ and $f_y(x, y)$, see e.g. [6, 9] and references therein. Their projections to the x-sphere

$\mathbb{CP}^1$, the base of the covering, are called the *discriminant points*[1] $b_1, \ldots, b_n$ and are assumed in this paper to be given. The task is thus to construct generators $\{\gamma_k\}_{k=1}^n$ of the fundamental group $\pi_1(\mathbb{CP}^1 \backslash \{b_1, \ldots, b_n\})$. This means to construct a set of closed contours $\gamma_1, \ldots, \gamma_n$, all starting at a (finite) common base point b_0 not being a discriminant point, each of the γ_k encircling exactly one discriminant point b_k in positive direction and being disjoint from other γ_j everywhere apart from the base point as shown in Fig. 1. To take into account a branching of the surface at the point at infinity, a contour γ_∞ starting and ending at b_0 and encircling all finite discriminant points in negative direction is used. We need the contours γ_k to satisfy the relation

$$\gamma_1 \gamma_2 \ldots \gamma_n \gamma_\infty = \mathrm{id}. \tag{2}$$

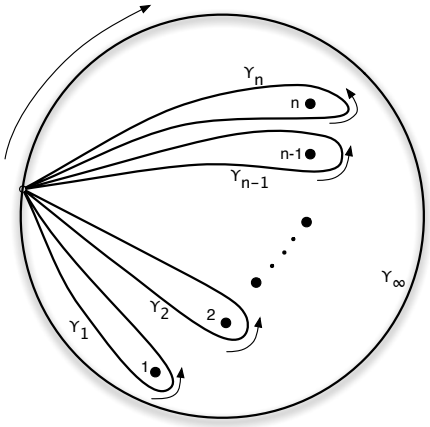

FIGURE 1. Generators of the fundamental group of $\mathbb{C} \setminus \{b_1, \ldots, b_n\}$.

In [9], the generators $\{\gamma_k\}$ were constructed in the same way as in the Maple package *algcurves*; here we briefly describe this approach. For numerical reasons it is important to stay away as much as possible from the discriminant points. Therefore we draw small disjoint circles centered at the discriminant points, with diameters strictly smaller than the minimal distance between the discriminant points. Each circle contains two marked points, the intersections of the circle with a straight line through the discriminant point parallel to the real axis. The left and right marked points are denoted by $b_k^{(1)}$ and $b_k^{(2)}$, respectively. One of

[1] An algebraic curve of the form (1) can have singularities, i.e., points where in addition to $f(x, y)$ and $f_y(x, y)$ also $f_x(x, y)$ vanishes. Such points as e.g. double points can have trivial monodromies, but are included in the monodromy computation. The notion of discriminant points thus includes branch points and singular points. Note that an algebraic curve has to be desingularized to define a Riemann surface, a process not to be discussed here (see for instance [6, 9] and references therein).

the leftmost marked points is denoted by b_0 and chosen to be the base point for $\pi_1(\mathbb{CP}^1 \setminus \{b_1, \ldots, b_n\})$. Starting from this base point straight lines are drawn to the marked points around each of the finite discriminant points. The contours γ_k are formed by these straight lines and the circles around the discriminant points.

This procedure is best illustrated by an example. Consider the curve given by $f(x, y) = y^3 - 2x^3y - x^9 = 0$. One quickly checks that the discriminant points are given by the roots of $x^9 = 2^5/3^3$ and the singular point $x = 0$. The resulting pattern can be seen in Fig. 2. In this example the contour γ_7 is just the positively

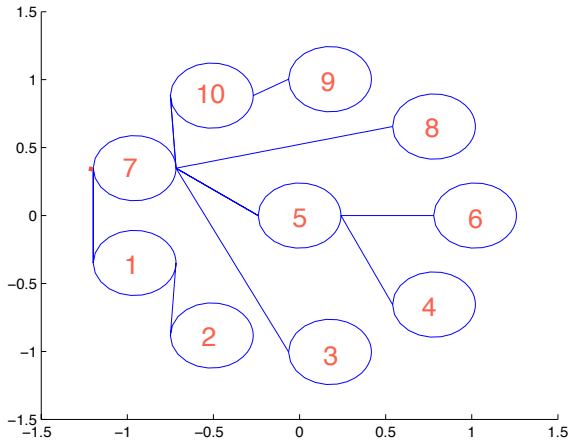

FIGURE 2. Contours for the monodromy computation for the curve $f(x, y) = y^3 - 2x^3y - x^9 = 0$ obtained via the deformation approach. The base point is marked with a small square.

oriented circle around the point b_7 marked with 7 starting and ending at the base point indicated with a small square. Similarly the contour γ_1 is formed by the line segment from the base point to $b_1^{(1)}$, the positively oriented full circle around b_1 and the straight line back to b_0. Drawing a straight line from b_0 to one of the marked points near b_{10} would lead to a line coming too close to b_7. The easiest way to remedy this is to consider the straight line between $b_7^{(2)}$ and $b_{10}^{(1)}$ instead. The contour γ_{10} thus consists of the upper half circle between $b_7^{(1)}$ and $b_7^{(2)}$, the line between $b_7^{(2)}$ and $b_{10}^{(1)}$ and the circle around b_{10}. It can happen that the distance between a line from b_0 to a point $b_k^{(j)}$ and another discriminant point b_i is smaller than some prescribed minimal distance δ, as would be the case for the line from $b_7^{(2)}$ to $b_6^{(1)}$ and b_5. In this case the contour is deformed as follows: instead of this line one considers the line between $b_7^{(2)}$ and $b_5^{(1)}$, the upper half circle around b_5 and the line between $b_5^{(2)}$ and $b_6^{(1)}$.

Thus if a connecting line comes closer than the distance δ to another discriminant point b_i, it is replaced by lines to and from the $b_i^{(j)}$ and a half circle around b_j. Since it is well possible that the new lines come also too close to other discriminant points, this procedure has to be iterated. A proof that the algorithm terminates has not been given (though such a proof should be possible given the finite number of problem points). More importantly, the resulting connecting lines will in general not be numerically optimal in the sense that they will not have the shortest possible lengths as is obvious from Fig. 2.

It is the purpose of this paper to address the outlined problems. Instead of deforming the connecting paths, we construct a minimal spanning tree having vertices at the discriminant points starting with the b_k closest to the base point. By construction, edges of this tree will have minimal lengths. The contours γ_k are then built as before from line segments between the marked points near discriminant points as they appear on the tree and the half circles. The result of this procedure for the same curve as in Fig. 2 can be seen in Fig. 3. The tree

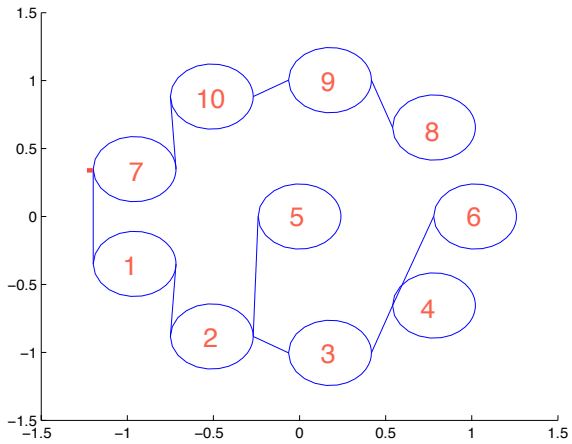

FIGURE 3. Contours for the monodromy computation for the curve $f(x,y) = y^3 - 2x^3y - x^9 = 0$ obtained via the minimal spanning tree. The base point is marked with a small square.

defines an initial set of contours $\tilde{\gamma}_k$, $k = 1, \ldots, n$ which are numerically optimal, but which do not yet satisfy condition (2). In a second step, the $\tilde{\gamma}_k$ will be combined in a way to form a new set of contours γ_k, $k = 1, \ldots, n$ which satisfy condition (2).

The paper is organized as follows: In section 2 we describe the construction of the minimal spanning tree and the contours $\tilde{\gamma}_k$ as well as the analytic continuation of the roots $y^{(k)}(b_0)$, $k = 1, \ldots, N$ along the contours. In section 3 the found contours are combined in a way that they satisfy condition (2). In section 4

we compare the actual numerical performance of the two approaches for some examples.

2. Contours for integration and minimal spanning tree

In this section we will explain how to construct the contours which generate the fundamental group of $\mathbb{CP}^1$ minus the finite discriminant points. These contours will be built from a minimal spanning tree, and the roots $y^{(k)}$, $k = 1, \dots, N$ of (1) will be analytically continued along them. The contours will not in general satisfy condition (2) which will be enforced in the next section.

We assume the finite discriminant points b_i, $i = 1, \dots, n$ to be given. Let ρ be the minimal distance between any two of these points,

$$\rho := \min_{\substack{i,j=1,\dots,n \\ i \neq j}} (|b_i - b_j|) \, .$$

For numerical reasons, one has to assume that ρ is considerably larger than the rounding error (in Matlab with double precision this error is typically of the order 10^{-14}). In practice, ρ has to be much larger for the reasons discussed below, see Remark 1. The code issues a warning if the ratio of the smallest distance to the largest distance between any two discriminant points is smaller than 10^{-4}, but will typically produce correct results in such cases. The code performs several checks to ensure that the obtained results are correct.

The starting configuration is as follows. Small disjoint circles centered at the discriminant points are drawn, with radius $R = \kappa\rho$ and $\kappa < 1/2$. In Figs. 2 and 3 we chose $\kappa = 1/2.9$ for plotting purposes, for the later computations values of up to $\kappa = 1/2.1$ are taken[2]. Each circle contains two marked points, the intersection of the circle with a straight line through the discriminant point parallel to the real axis. The marked points on the circle around b_k are denoted by $b_k^{(1)}$ and $b_k^{(2)}$, where $\mathrm{Re}\left\{b_k^{(1)}\right\} < \mathrm{Re}\left\{b_k^{(2)}\right\}$. One of the leftmost marked points is chosen to be the base point b_0 for $\pi_1(\mathbb{CP}^1 \setminus \{b_1, \dots, b_n\})$. The discriminant points $\{b_k\}_{k=1}^n$ are then ordered according to the ascending complex argument of the vectors $b_k - b_0$, the argument being measured from $-\pi$ to π; if two discriminant points lie on the same ray originating at b_0, then the discriminant point which is closer to

[2]The value of κ is essentially fixed by hand. Since the same number of collocation points is used on the half-circles and on the connecting lines between them, the length of the half-circles and the segments of the connecting line between them should ideally be the same to produce a homogeneous numerical resolution over the path. Therefore, a value of κ close to $1/(\pi + 2)$ would be an appropriate choice. However, for an efficient resolution of high order singularities, where several sheets of the covering coincide, we choose the distance from the path to each critical point to be almost the maximal possible. In practice the code uses $\kappa = 1/2.1$ to allow for connecting lines of positive length between the circles (the Maple package works with $\kappa = 2/5$).

the base point is preceding in the order. This ordering is shown for the studied example in Figs. 2 and 3 by the numbers in the circles.

Then a minimal tree originating at the discriminant point b_{k_0} closest to the base point, $b_0 = b_{k_0}^{(1)}$, is constructed. To this end, all distances to the remaining b_i, $i \neq k_0$ are computed, and the point with the smallest distance, denoted b_{k_1}, is chosen as the next vertex on the tree. If there are several points with the smallest distance, then the one with the smallest ordering number is chosen. The code stores the pair $[k_0, k_1]$ to indicate that the tree starts at b_{k_0} and that its next vertex is at b_{k_1}. To obtain the next vertex, the distances between both b_{k_0} and b_{k_1} and the remaining b_i, $i \neq k_0, k_1$ are computed. The smallest distance (in case of degeneracies again the point with the smallest ordering number is chosen) gives the next vertex b_{k_2} of the tree, connected by an edge either to b_{k_0} or to b_{k_1}. The code stores the pair $[k_0, k_2]$ or $[k_1, k_2]$*respectively*.Repeating this several times one ends up with a tree containing the points $b_{k_0}, b_{k_1}, \ldots, b_{k_m}$ as vertices. The next vertex on the tree is determined as before by computing the minimal distance between points already on the tree and points b_k, $k \neq k_0, \ldots, k_m$ and dealing with degeneracies as before. Thus by construction, one obtains in this way a minimal spanning tree of the b_k, $k = 1, \ldots, n$ originating at b_{k_0}. The tree is not unique because of possible degeneracies of distances between the points, but the described algorithm will always produce a connected spanning tree with minimal distances between the points. The result of this procedure for the curve in Fig. 3 is

```
paths' =

7     1     2     3     7    10     9     4     2
1     2     3     4    10     9     8     6     5,
```

where it can be seen that there is a connection between 7 and 1, then between 1 and 2 , between 2 and 3 and so on.

This tree just indicates in which order the discriminant points appear on the paths $\tilde{\gamma}_k$. The actual contours will consist of half circles around the b_k, $k = 1, \ldots, n$ and straight lines between the points $b_k^{(1,2)}$, $k = 1, \ldots, 2$. Thus in a separate step the precise paths will be determined. For each pair of consecutive points b_{k_0} and b_{k_1} appearing on the tree, the connecting lines between $b_{k_0}^{(1,2)}$ and $b_{k_1}^{(1,2)}$ are chosen in a way that they intersect the circles around these two points as little as possible. To this end the code determines the real part of the difference between the two points, $d = \mathrm{Re}(b_{k_1} - b_{k_0})$. If this distance is greater or equal to R, the line segment connects $b_{k_0}^{(2)}$ and $b_{k_1}^{(1)}$, if it is smaller than $-R$, the line will be between $b_{k_0}^{(1)}$ and $b_{k_1}^{(2)}$, and for values between $-R$ and R, the points $b_{k_0}^{(1)}$ and $b_{k_1}^{(1)}$ are connected. The result of this procedure is stored in a pair of numbers for each pair of neighbouring vertices in the tree. For the example of Fig. 3 the code gives

```
pathind' =

1     2     2     2     2     2     2     1     2
1     1     1     1     1     1     1     1     1.
```

This has to be read together with the above information given by the variable *path*. The two variables together specify the edge $[b_i^{(j)}, b_k^{(l)}]$ of the tree, where the variable *path* gives the pair (i, k) and *pathind* gives (j, l). In the considered example, the first line segment is thus between $b_7^{(1)}$ and $b_1^{(1)}$ as can be seen in Fig. 3. By construction, all connecting lines will have minimal possible lengths whilst keeping at least the distance $\delta = R\sqrt{1-\kappa^2}$ (see [9]) away from the discriminant points.

The contours $\tilde{\gamma}_k$ are then built from these connecting lines and half circles around discriminant points in the following way. The contour $\tilde{\gamma}_{k_1}$ is a contour starting at b_0 and encircling the point b_{k_1} only. Thus the code constructs $\tilde{\gamma}_{k_1}$ that starts at the base point b_0, goes between the points b_k in a sequence of connecting lines and half circles appearing on the tree before the index k_1, then it follows positively the circle around b_{k_1} and, finally, takes the same path (minus the circle around b_{k_1}) back to the base point. Any discriminant point appearing on this path is bypassed on a half-circle in positive direction. The contour $\tilde{\gamma}_5$ in Fig. 3 starts for instance at the base point $b_7^{(1)}$ with the straight line to $b_1^{(1)}$, then the half circle in positive direction around b_1 to $b_1^{(2)}$, from there the straight line to $b_2^{(1)}$, the half circle around b_2 in positive direction to $b_2^{(2)}$, from there the straight line to $b_5^{(1)}$, and after a full circle around b_5 in positive direction the same path from $b_5^{(1)}$ back to the base point $b_7^{(1)}$ in the opposite direction.

A schematic view of the relative positions of the loops $\tilde{\gamma}_k$, $k = 1, \ldots, 10$ constructed from the spanning tree in Fig. 3 is presented in Fig. 4.

To determine the monodromies for this set of contours, the algebraic equation (1) is solved for y at the base point $x = b_0$, which is a generic point of the curve. Thus, there will be N distinct roots $y_0^{(k)}$, $k = 1, \ldots, N$ at this point which can be determined numerically with the Matlab function *roots*. The roots are labeled $1, \ldots, N$, thus numbering the sheets of the covering of the x-sphere $\mathbb{CP}^1$. These roots are then analytically continued along $\tilde{\gamma}_k$. The analytical continuation results in the same set of roots $y_0^{(k)}$ at $x = b_0$ but in a different order. The permutation of the roots thus obtained is the monodromy of the Riemann surface along the path $\tilde{\gamma}_k$.

In order to compute the analytical continuation, we introduce a numerical grid, which amounts to choosing collocation points on each of the line segments and half circles. Since later on the code uses $\tilde{\gamma}_k$ as integration contours to compute integrals of *holomorphic differentials* (see [6, 9] how these can be determined) to obtain the *periods* of a Riemann surface (integrals of the holomorphic differentials

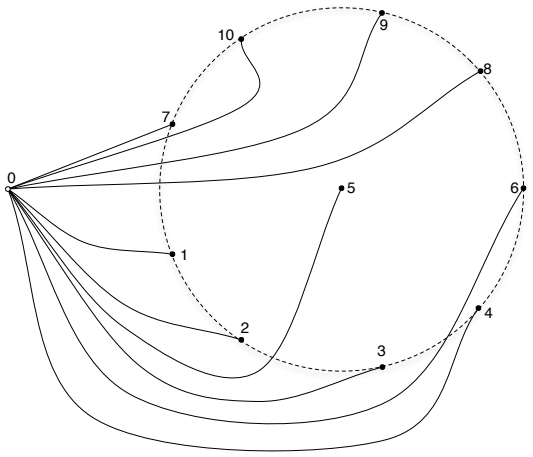

FIGURE 4. Schematic diagram for the relative position of the contours generated by the minimal tree shown in Fig. 3.

along closed contours formed by the $\tilde{\gamma}_k$), it is convenient to choose the collocation points in accordance with the used integration scheme. Since we use numerically optimal Gauss-Legendre integration, which can be implemented conveniently in Matlab with Trefethen's code [17, 19], we take Gauss-Legendre points. On each line segment and half circle there will be thus N_G points (typically N_G is between 32 and 128).

At each of these collocation points x^c, we use *roots* to solve numerically equation (1) to obtain N roots $y^{(k)}(x^c)$. In general, the ordering of these roots will not correspond to the one introduced at the base point. Thus the roots are sorted in a way to have minimal difference with the roots at the previous collocation point, i.e., $|y^{(k)}(x_i^c) - y^{(k)}(x_{i-1}^c)| = \min_{j=1,\ldots,N} |y^{(j)}(x_i^c) - y^{(k)}(x_{i-1}^c)|$. In this way the vector $\vec{y}$ of roots is analytically continued along the contours $\tilde{\gamma}_k$.

Remark 1. By construction, no discriminant points appear on the contours $\tilde{\gamma}_k$ which implies that there will be always N distinct roots $y^{(i)}$ of (1) on these contours. The roots can be computed in Matlab efficiently with the function *roots* as long as they are well separated. It is known that the computation of almost degenerate zeros of polynomials is an extremely difficult numerical problem, see for instance [20] and references therein. In the present context, having almost degenerate roots would mean that the sheets come very close to each other, which is typically the case near high order singularities. Such singularities can be seen as a condensation of many double points (the point (0,0) of the example in Fig. 3 is of this type). If another discriminant point comes so close to such a singularity that the sheets can no longer be numerically distinguished, i.e., if ρ becomes too small in such a case, the surface cannot be studied with the present code. It is in fact the ability to distinguish the sheets numerically with the *roots* function of Matlab along the contours $\tilde{\gamma}_k$ that imposes limitations on which curves can be

treated by the code. Since these limitations depend strongly on the considered curve, it is impossible to give a priori limits on the applicability of the code.

If we start at the base point b_0 and analytically continue the vector of roots $\vec{y}$ with components $y^{(i)}$, $i = 1, \dots, N$ as described above along one of the contours $\tilde{\gamma}_k$, we will in general obtain a permutation σ_k of the components of the vector $\vec{y}$ back at the base point,

$$\sigma_k \vec{y} := (y^{\sigma_k(1)}(a), \dots, y^{\sigma_k(N)}(a)). \tag{3}$$

The group generated by the $\{\sigma_i\}_{i=1}^n$ is called the monodromy group of the covering. The code stores the monodromies σ_i in the form of a vector of the indices $(\sigma_i(1), \dots, \sigma_i(N))$. For the curve in Fig. 3 and the set of contours $\tilde{\gamma}_k$, $k = 1, \dots, n$, one obtains the base point

```
base =

      -1.2895 + 0.3485i
ybase =

-0.9546 - 2.8682i
 1.9591 + 1.1931i
-1.0044 + 1.6751i
```

and the monodromies

```
    Mon =

 3     2     1     3     3     1     1     3     1     3
 2     1     3     2     2     3     3     2     3     2
 1     3     2     1     1     2     2     1     2     1.
```

This is to be read in the following way: the vector of roots *ybase* is analytically continued along the loop $\tilde{\gamma}_1$; the result of this continuation is a new vector of roots obtained from *ybase* by permuting the components as specified by the first vector of *Mon*, the permutation (321), i.e., starting in the first sheet, one ends up in the third, starting from the second one stays there, and starting in the third one ends up in the first.

As was already mentioned, the monodromy at infinity can be computed by analytically continuing the vector of roots $\vec{y}$ at the base b_0 along a closed contour starting and ending at b_0 and encircling once all finite discriminant points in negative direction as shown in Fig. 1. Since the radius of such a loop can be very large, a high number of collocation points would be needed to obtain the same accuracy as for the loops around the finite discriminant points. Thus we will obtain the monodromy there from the relation $\gamma_1 \dots \gamma_n = \gamma_\infty^{-1}$, see (2).

3. Generators of the fundamental group

The set of *initial loops* $\{\tilde{\gamma}_k\}$ constructed in the previous section does not satisfy in general condition (2), as can be seen from Fig. 4 . We call the monodromies $\{\tilde{\phi}_k\}_{k=1}^n$ computed along these loops the *initial monodromies.* In this section we explain how generators of the fundamental group of the base of the covering punctured at the discriminant points $\mathbb{CP}^1 \setminus \{b_1, \ldots, b_n\}$ satisfying (2) can be constructed from the initial loops.

Let us suppose that a permutation $\sigma \in S_n$ exists such that

$$\tilde{\gamma}_{\sigma(1)}\tilde{\gamma}_{\sigma(2)} \ldots \tilde{\gamma}_{\sigma(n)} = \tilde{\gamma}_\infty^{-1}. \tag{4}$$

Then there are two possible ways to proceed. First, the discriminant points $\{b_k\}$ can be reordered according to σ, i.e., $b_k := b_{\sigma(k)}$ and the corresponding generators of the fundamental group are given by $\gamma_k := \tilde{\gamma}_{\sigma(k)}$. However, we would like to keep the initial ordering of the branch points. To this end, the following algorithm is applied to construct the $\{\gamma_k\}$ satisfying (2): (i) if the permutation σ is trivial, then put $\gamma_k = \tilde{\gamma}_k$. Otherwise, (ii) let $\sigma(m)$ be the largest number such that $\sigma(m) \neq m$. Then $\sigma(m+1) < \sigma(m)$ and the loop $\tilde{\gamma}_{\sigma(m)}$ is redefined as follows: $\tilde{\gamma}_{\sigma(m)} := \tilde{\gamma}_{\sigma(m+1)}\tilde{\gamma}_{\sigma(m)}\tilde{\gamma}_{\sigma(m+1)}^{-1}$ (the loop $\tilde{\gamma}_{\sigma(m+1)}^{-1}$ is traced first, then $\tilde{\gamma}_{\sigma(m)}$ followed by $\tilde{\gamma}_{\sigma(m+1)}$, see also Fig. 5 below). The permutation σ is then composed with the transposition swapping $\sigma(m)$ and $\sigma(m+1)$, i.e., $\sigma := (\sigma(m)\sigma(m+1)) \circ \sigma$. After this, the algorithm is reiterated.

Let us now find the permutation σ from (4) imposed by the minimal tree and the construction of the initial loops $\{\tilde{\gamma}_k\}$.

Consider two loops γ, $\delta \in \pi_1(\mathbb{CP}^1 \setminus \{b_1, \ldots, b_n\}, b_0)$, more precisely, consider the parts from b_0 to the respective points they encircle. These might have a common part close to the base point like some of the initial loops $\{\tilde{\gamma}_k\}$ do. At the point when γ and δ separate, denote by $\overrightarrow{\gamma}$ and $\overrightarrow{\delta}$ the tangent vectors to the loops. Then the orientation of the pair $(\overrightarrow{\gamma}, \overrightarrow{\delta})$ indicates the relative position of the two loops.

Lemma 1. *Suppose now that a set of generators $\{\gamma_k\}_{k=1}^n$ of the group $\pi_1(\mathbb{CP}^1 \setminus \{b_1, \ldots, b_n\}, b_0)$ is such that (i) each loop encircles only one puncture; (ii) the pair $(\overrightarrow{\gamma}_k, \overrightarrow{\gamma}_{k+1})$ is positively oriented at any point of separation of γ_k and γ_{k+1} for all $k = 1, \ldots, n-1$; (iii) the loops γ_k do not intersect each other apart from the base point (there exist representatives in the corresponding homotopy classes which do not intersect). Then the loops $\{\gamma_k\}$ satisfy (2).*

The proof of this lemma is obvious.

By construction, the initial loops $\{\tilde{\gamma}_k\}$ satisfy conditions (i) and (iii) of the lemma. Therefore, we are looking for the permutation $\sigma \in S_n$ such that the pairs $(\overrightarrow{\gamma}_{\sigma(k)}, \overrightarrow{\gamma}_{\sigma(k+1)})$ be positively oriented at the corresponding points of separation of loops for all $k = 1, \ldots, n-1$.

Let us introduce some notation. Whenever the path contains a sequence of edges of the type $[\dots, b_j^{(1)}]\ [b_j^{(1)}, \dots]$ or $[\dots, b_j^{(2)}]\ [b_j^{(2)}, \dots]$, the point $b_j^{(1)}$ or $b_j^{(2)}$, is called a *v-point*. The only such point in Fig. 3 is $b_4^{(1)}$. We call a part of a single branch of a tree without v-points a *string*. A *node* is a vertex where several branches meet. In what follows, the v-points are considered as nodes, where the corresponding discriminant point becomes a separate branch consisting of only one point. We call a node *simple* if all of its descendants are strings.

In order to find an algorithm which produces the permutation σ from (4) for the given minimal tree, we first discuss two particular types of trees.

I. In the case when the minimal tree is a string, the permutation $\sigma = (\sigma(1)\dots\sigma(n))$ is given by the sequence of labels of the branch points read from the end of the string towards the base point b_0 so that $b_0 = b^{(1)}_{\sigma(n)}$. This follows directly by construction of the initial loops.

II. Suppose the tree contains only one node which coincides with the base point b_0. Suppose there are m branches meeting at the node, each of which is a string. To each branch of the tree the algorithm associates a sequence $s^i = (s^i_1, \dots, s^i_{n_i})$, $i = 1, \dots, m$ of numbers like above - a sequence of labels of the discriminant points read from the end of the string towards the node. This sequence indicates the order in which the loops $\{\tilde\gamma_k\}$ should be composed to give a positively oriented loop around all points contained in the string. Now we need to decide on the relative position of the branches at the node, i.e., a relative position of the vectors $\overrightarrow{\tilde\gamma}_{s^i_1}$, $i = 1, \dots, m$ at b_0. We order the vectors according to the ascending angle they make with the horizontal ray going from the base point b_0 to the left, the angle is measured from 0 to 2π. This order is expressed as a permutation $\rho \in S_m$ of the indices attached to the branches, i.e., the sequences s^i are ordered as follows: $s^{\rho(1)}, \dots, s^{\rho(m)}$. Now the required permutation σ is obtained by writing the sequences s^i one after the other in the order they appear at the node: $\sigma = (\sigma(1)\dots\sigma(n)) := (s_1^{\rho(1)} \dots s_{n_{\rho(1)}}^{\rho(1)} s_1^{\rho(2)} \dots s_{n_{\rho(2)}}^{\rho(2)} \dots s_1^{\rho(m)} \dots s_{n_{\rho(m)}}^{\rho(m)})$.

Now we are in a position to present the complete algorithm.

Algorithm. In the general case, the algorithm first identifies the end points of the tree (those without descendants) and the nodes (including the v-points). Then starting at each of the end points, the algorithm forms a sequence of labels of the discriminant points going from the current point to its parent until it hits a node. This process results in a set of all simple nodes and a set of sequences of numbers attached to every such node. For each of the simple nodes, the procedure from case **II** is performed where, if the node is different from b_0, instead of the horizontal ray going left from the base point, the line of arrival to the node with the reversed orientation is taken (the line of arrival is the line segment from the previous marked point on the path to the current one). The result of this procedure is a sequence $\mathbf{s}$ of numbers at each node which indicates the order in which the loops $\{\tilde\gamma_k\}$ should be composed to obtain a positively oriented loop

encircling all descendants of the given simple node and only them. Thus the part of the tree descending from a simple node can be treated as a string in which the points are arranged in the obtained order **s**. Therefore, after the ordering of the loops at all simple nodes is done, each simple node is considered as an end point of the tree whose label is given by the sequence of numbers **s**. Then the algorithm is reiterated.

We illustrate the algorithm on the example of Fig. 3. The code first identifies the endpoints

```
endpoints =

 8
 6
 5,
```

and the nodes

```
nodes =

 7     2
 1     2,
```

corresponding to the points $b_7^{(1)}$ (the base point) and $b_2^{(2)}$. For technical reasons the v-points are not identified at this stage. Starting from the endpoints, the code then traces the branches until it hits the first node on each branch. At each point, it is checked whether the point is a v-point. If such a point is reached, the standard order procedure at a node is applied. Each point not being a v-point on such a branch is listed in the order of appearance. Thus at the v-point $b_4^{(1)}$, the code considers two branches 6 and 4 and places 4 in front of 6 since the angle between the reversed arrival line $\overline{b_4^{(1)}b_3^{(2)}}$ and $\overline{b_4^{(1)}b_4}$ is smaller than the one between $\overline{b_4^{(1)}b_3^{(2)}}$ and $\overline{b_4^{(1)}b_6^{(1)}}$. At the node $b_2^{(2)}$ we thus get the two strings

```
tree{2} =

         4     6     3

tree{3} =

         5.
```

At the node $b_2^{(2)}$, these strings are combined into a single string, where the sequence $tree\{2\}$ comes in front of $tree\{3\}$ because the angle between the reversed arrival line $\overline{b_2^{(2)}b_2^{(1)}}$ and $\overline{b_2^{(2)}b_3^{(1)}}$ is smaller than that between $\overline{b_2^{(2)}b_2^{(1)}}$ and $\overline{b_2^{(2)}b_5^{(1)}}$. This leads to

```
Tree =

         4      6      3      5
```

This branch and the one from the remaining endpoint are then traced to the base point, where we get

```
tree{1} =

        8      9     10      7

tree{2} =

        4      6      3      5      2      1.
```

Again the two strings are combined at the base point according to the angles of the connecting lines to give

```
Tree =

        4      6      3      5      2      1      8      9     10      7.
```

In words, this string gives the relative 'position' of homotopy classes of the loops which can also be seen from Fig. 4: in this sense the contour $\tilde{\gamma}_{10}$ is entirely to the 'left' of $\tilde{\gamma}_k$, $k \neq 7, 10$ as wanted, but it is to the 'right' of $\tilde{\gamma}_7$. Thus the contour γ_{10} will be obtained by conjugating $\tilde{\gamma}_{10}$ with $\tilde{\gamma}_7$ to get $\gamma_{10} := \tilde{\gamma}_7\tilde{\gamma}_{10}\tilde{\gamma}_7^{-1}$ (by this notation we mean that the contour $\tilde{\gamma}_7^{-1}$ is traced first, then $\tilde{\gamma}_{10}$ followed by $\tilde{\gamma}_7$). The action of such a conjugation is illustrated in Fig. 5: the situation before the conjugation can be seen on the right, and the effect of the conjugation on the left.

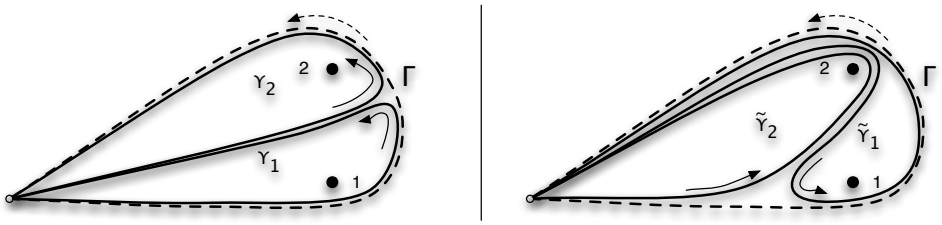

FIGURE 5. The contour Γ surrounding the two points 1 and 2 is represented in two different ways by contours around the individual points. On the left, $\Gamma = \gamma_2\gamma_1$ while $\Gamma = \tilde{\gamma}_1\tilde{\gamma}_2$ on the right. Since $\gamma_2 = \tilde{\gamma}_2$ it follows that $\tilde{\gamma}_1 = \gamma_2\gamma_1\gamma_2^{-1}$.

The code identifies the largest number m in the string *Tree*. If it is in the rightmost position, this number is deleted from $Tree$. If there is a number k to

the right, the loop $\tilde{\gamma}_m$ and the monodromy $\tilde{\phi}_m$ are redefined by $\tilde{\gamma}_m := \tilde{\gamma}_k \tilde{\gamma}_m \tilde{\gamma}_k^{-1}$ and $\tilde{\phi}_m := \tilde{\phi}_k \tilde{\phi}_m \tilde{\phi}_k^{-1}$. The numbers m and k are then swapped in $Tree$. This procedure is repeated until m is in the right-most position. It is then deleted from $Tree$ and the procedure is repeated until there is only one element in $Tree$ left. The resulting loops and monodromies are the desired γ_k and ϕ_k. Finally the monodromy at infinity is obtained from relation (2). For the considered example this leads to

```
Mon =

 1     2     2     3     2     1     3     3     1     3
 3     1     1     2     1     3     2     2     3     2
 2     3     3     1     3     2     1     1     2     1.
```

In this example infinity is a singular point with trivial monodromy which is why the code gives no monodromy at infinity (it would appear at position $n+1$). We note that it is possible to compute the genus g, the only topological invariant of a Riemann surface, from the monodromies via the Riemann-Hurwitz formula,

$$g = 1 - N + \frac{1}{2} \sum_{i=1}^{N_B} \beta_i \ ,$$

where N is the total number of sheets, β_i is the branching number, the number of sheets connected at a point on the covering minus 1, and where N_B is number of discriminant points on the covering. For the studied example one finds thus $g = 3$.

4. Performance of the code

We have described two algorithms for computing monodromies. The first constructs contours γ_k by the deformation approach, while the second achieves this from a spanning tree construction. In order to judge the performance of both approaches we will compute characteristic quantities of a Riemann surface for several examples.

It is known (see for instance the standard literature on Riemann surfaces such as [16]) that the space of holomorphic one-forms ω of a surface of genus g is g-dimensional. For the homology of the surface one can introduce a canonical basis of cycles a_i, b_i, $i = 1, \ldots, g$ such that $a_i \circ b_j = \delta_{ij}$. These a- and b-cycles can be obtained from the loops γ_k, $k = 1, \ldots, n$ via an algorithm by Tretkoff and Tretkoff [18]. For normalized holomorphic one-forms such that $\oint_{a_i} \omega_j = \delta_{ij}$, the matrix of b-periods $\mathbb{B}_{ij} = \oint_{b_i} \omega_j$ is a Riemann matrix, it has a positive definite imaginary part and is symmetric. Thus for a given basis of the holomorphic one-forms, the code computes the periods from the integrals along the $\tilde{\gamma}_k$, $k = 1, \ldots, n$ via Gauss-Legendre integration. The found numerical Riemann matrix will not be

exactly symmetric. Since the symmetry of $\mathbb{B}$ is not imposed, its asymmetry is a strong test of the quality of the numerics. In the following we will use the norm Δ of $\mathbb{B} - \mathbb{B}^T$ (the eigenvalue of the matrix having the largest absolute value) as a measure of the numerical error. We take two codes which are identical except for the part where the contours γ_k are generated and compute their performance for typical examples.

As already stated, the curve of Fig. 2 and 3 has genus 3. A basis of the holomorphic one-forms is given by $x^3/f_y(x,y)$, $x^4/f_y(x,y)$, and $xy/f_y(x,y)$. The errors we obtain for $\kappa = 1/2.9$ are given in Table 1.

N_G	Δ_{def}	Δ_{st}
8	$2.14 * 10^{-5}$	$1.13 * 10^{-4}$
16	$8.15 * 10^{-9}$	$1.55 * 10^{-9}$
32	$1.61 * 10^{-13}$	$1.63 * 10^{-15}$

TABLE 1. Norm of $\mathbb{B}-\mathbb{B}^T$ for the curve $f(x,y) = y^3-2x^3y-x^9 = 0$ for the deformation approach (Δ_{def}) and the spanning tree (Δ_{st}) in dependence of the number N_G of collocation points on each segment of the γ_k.

It can be seen that the error shows the expected spectral convergence in both cases, but that the spanning tree gives a numerical error almost two orders of magnitude better than the deformation approach except for $N_G = 8$ where the resolution is too low in both cases. It is remarkable that machine precision can be reached with this method with just 32 collocation points on each segment of the contours. The whole computation takes just 0.5s on a laptop in this case.

The advantage of the spanning tree is more visible for more involved curves such as $f(x,y) := y^9 + 2x^2y^6 + 2x^4y^3 + x^6 + y^2 = 0$. This curve of genus 16 has 43 finite discriminant points with minimal distance $\rho = 0.018$ between them. The monodromy computation is extremely demanding in this case. The points in the outer ring in Fig. 6 represent pairs of discriminant points of the curve separated by a distance of only 0.018. For the deformation approach we have chosen the base point close to the geometric center of the discriminant points, i.e., close to the point $x = 0$. This gives shorter integration paths and was used in general for the deformation approach in [9]. For the spanning tree the choice of the base point has no influence on the length of the connecting lines since we use a minimal tree.

The code produces the values of the norm of the antisymmetric part of the computed Riemann matrices given in Table 2. The computation with $N_G = 64$ takes $20s$ in the latter case. The deformation approach did not produce a result for $N_G = 32$. More importantly the spanning tree needs in this case just half the number of modes to reach the same precision as the deformation approach until

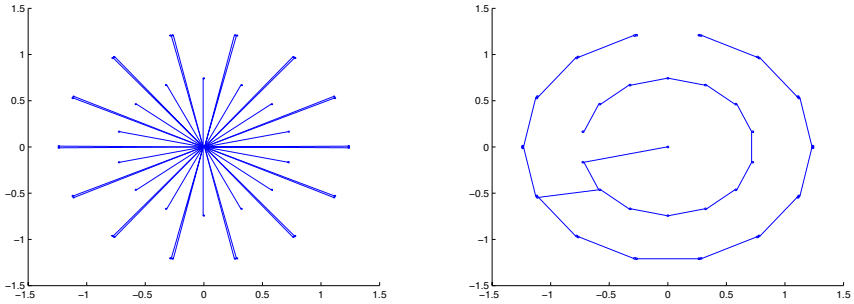

FIGURE 6. Loops for the monodromy computation for the curve $f(x,y) := y^9 + 2x^2y^6 + 2x^4y^3 + x^6 + y^2 = 0$ obtained with a deformation approach with base point close to $x = 0$ on the left and with a spanning tree on the right.

both reach the saturation level. This implies that a factor 2 in allocated resources and CPU time can be gained with this approach which allows consequently the study of more involved curves.

References

[1] Bobenko, A.I.: Introduction to Compact Riemann Surfaces, In Bobenko, A.I., and Klein, C. (ed.), 'Computational Approach to Riemann Surfaces', Lect. Notes Math. **2013** (2011).

[2] Belokolos, E.D., Bobenko, A.I., Enolskii, V.Z., Its, A.R., Matveev, V.B.: Algebro-geometric approach to nonlinear integrable equations. Springer, Berlin (1994)

[3] Bobenko, A., Bordag, L.: Periodic multiphase solutions to the Kadomtsev-Petviashvili equation. J. Phys. A: Math. Gen., **22**, 1259 (1989)

[4] Couveignes, J.-M.: Tools for the computation of families of coverings, In Aspects of Galois theory (Gainesville, FL, 1996), 38-65, London Math. Soc. Lecture Note Ser., 256, (Cambridge Univ. Press, Cambridge, 1999).

[5] Deconinck, B., van Hoeij, M.: Computing Riemann matrices of algebraic curves. Physica D, **152–153**, 28–46 (2001)

N_G	Δ_{def}	Δ_{st}
32	...	$5.49 * 10^{-6}$
64	$6.24 * 10^{-7}$	$9.37 * 10^{-12}$
128	$3.66 * 10^{-11}$	$1.28 * 10^{-13}$

TABLE 2. Norm of $\mathbb{B} - \mathbb{B}^T$ for the curve $f(x,y) := y^9 + 2x^2y^6 + 2x^4y^3 + x^6 + y^2 = 0$ for the deformation approach (Δ_{def}) and the spanning tree (Δ_{st}) in dependence of the number of collocation points on each segment of the γ_k.

[6] Deconinck, B. and Patterson, M.: Computing with plane algebraic curves,In Bobenko, A.I., and Klein, C. (ed.), 'Computational Approach to Riemann Surfaces', Lect. Notes Math. **2013** (2011).
[7] Frauendiener, J., Klein, C.: Hyperelliptic theta-functions and spectral methods. J. Comp. Appl. Math., **167**, 193 (2004)
[8] Frauendiener, J., Klein, C.: Hyperelliptic theta-functions and spectral methods: KdV and KP solutions, Lett. Math. Phys., **76**, 249-267 (2006)
[9] Frauendiener, J., Klein, C.: Algebraic curves and Riemann Surfaces in Matlab, In Bobenko, A.I., and Klein, C. (ed.), 'Computational Approach to Riemann Surfaces', Lect. Notes Math. **2013** (2011).
[10] Grava, T., Klein, C.: Numerical solution of the small dispersion limit of Korteweg de Vries and Whitham equations, Comm. Pure Appl. Math., **60**, 1623-1664 (2007)
[11] Klein, C., Kokotov, A., Korotkin, D.: Extremal properties of the determinant of the Laplacian in the Bergman metric on the moduli space of genus two Riemann surfaces. Math. Zeitschr. **261**(1), 73–108 (2009)
[12] Klein, C., Richter, O.: Ernst Equation and Riemann Surfaces. Lecture Notes in Physics **685**, Springer, Berlin (2005)
[13] Lax, P.D., Levermore, C.D.: The small dispersion limit of the Korteweg de Vries equation, I,II,III. Comm. Pure Appl. Math., **36**, 253-290, 571-593, 809-830 (1983)
[14] Quine, J.R., Sarnak, P. (ed): Extremal Riemann surfaces. Contemporary Mathematics, **201** AMS (1997)
[15] Schmies, M.: Computing Poincaré Theta Series for Schottky Groups, In Bobenko, A.I., and Klein, C. (ed.), 'Computational Approach to Riemann Surfaces', Lect. Notes Math. **2013** (2011).
[16] Springer, G.: Introduction to Riemann surfaces. Addison-Wesley Publishing Company, Inc., Reading, Mass. (1957)
[17] Trefethen, L.N.: Spectral Methods in Matlab. SIAM, Philadelphia, PA (2000)
[18] Tretkoff, C.L., Tretkoff, M.D.: Combinatorial group theory, Riemann surfaces and differential equations. Contemporary Mathematics, **33**, 467–517 (1984)
[19] www.comlab.ox.ac.uk/oucl/work/nick.trefethen
[20] Zeng, Z.: Computing multiple roots of inexact polynomials. Math. Comp. **74**, 869-903 (2004)

J. Frauendiener E-MAIL: joergf@maths.otago.ac.nz
ADDRESS: *Department of Mathematics and Statistics, University of Otago, P.O. Box 56, Dunedin 9010, New Zealand, and: Centre of Mathematics for Applications, University of Oslo, P.O. Box 1053, Blindern, NO-0316 Oslo, Norway*

C. Klein E-MAIL: Christian.Klein@u-bourgogne.fr
ADDRESS: *Institut de Mathématiques de Bourgogne, Université de Bourgogne, 9 avenue Alain Savary, 21078 Dijon Cedex, France*

V. Shramchenko E-MAIL: Vasilisa.Shramchenko@Usherbrooke.ca
ADDRESS: *Département de mathématiques, Université de Sherbrooke, 2500, boul. de l'Université Sherbrooke (Québec) Canada J1K 2R1*