Continuity in Dynamic Geometry

An Algorithmic Approach

Stefan Kranich

Technische Universität München
Fakultät für Mathematik

Continuity in Dynamic Geometry

An Algorithmic Approach

Stefan Helmut Kranich

Vollständiger Abdruck der von der Fakultät für Mathematik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Die Dissertation wurde am 12.11.2015 bei der Technischen Universität München eingereicht und durch die Fakultät für Mathematik am 23.03.2016 angenommen.

Acknowledgements

First and foremost, I want to thank my advisor, Jürgen Richter-Gebert. He has inspired and guided me ever since I have started to work with him. His love for geometry has permanently rubbed of on me. I thank him for introducing me to the topic of this thesis, which has been very demanding but also very rewarding for me. I am indebted to him for giving me great freedom in pursuing my research interests. Beyond mathematics, as a role model, Jürgen has taught me how to present, how to teach, and to strive for a healthy balance between perfectionism and pragmatism. Thank you, Jürgen, for always supporting me in every possible way.

I am very grateful to Ulrich Kortenkamp and Elias Wegert for examining my thesis. I would also like to thank Elias Wegert for his kind permission to use his domain colouring colour scheme.

Moreover, I would like to thank all colleagues at the Chair for Geometry and Visualization for countless wonderful discussions, helpful suggestions, and the great working atmosphere. I thank Tim Hoffmann for introducing me to discrete Darboux transformations. I thank Susanne Apel, Martin von Gagern, and Katharina Schaar, who shared office with me, for many constructive debates and exchanging ideas. I am especially grateful to Katharina Schaar for reading several manuscripts and giving valuable feedback. I thank Diane Clayton-Winter for offering her advice as a native speaker whenever I was struggling with the English language; the remaining mistakes are all mine.

I want to express my gratitude to DFG Collaborative Research Center TRR 109, “Discretization in Geometry and Dynamics”, and Technische Universität München, Department of Mathematics, for funding my research.

Last but not least, I thank my family for their love, continuous support, encouragement, and patience. This thesis is dedicated to them.

Contents

1 Introduction

1.1 Poncelet's Principle of Continuity

Jean-Victor Poncelet (1788–1867) was a French engineer and mathematician. From 1808 to 1810, he studied at the École Polytechnique under Gaspard Monge (1746–1818). In 1812, Poncelet joined the French army in Napoleon's campaign against Russia. Poncelet gives the following account of his wartime experiences in the preface of (Poncelet, 1862). At the Battle of Krasnoi, he was left for dead, captured by the Russians and imprisoned as a prisoner of war at Saratov (1813–1814). Having nothing else to do, and not having access to scientific literature, Poncelet began to redevelop from memory what he had learned at École Polytechnique. He started to form new ideas on geometry, about which he lectured to fellow polytechniciens imprisoned with him. After his release in 1814, Poncelet returned to France. He elaborated and published his ideas from Saratov in his *Traité des propriétés des figures* (Poncelet, 1822). In this work, Poncelet laid the foundations for modern projective geometry.

The programme of the *Traité* (as Poncelet describes in its introduction) is the revival of synthetic geometry ('géométrie ordinaire') in an attempt to make it an equal rival to analytic geometry. Poncelet questions what makes analytic geometry so successful compared to the synthetic geometry of the ancients. He finds that the power of analytic geometry lies in its possibility to abstract from a concrete instance of a geometric construction and to obtain general results by manipulation of equations in which indeterminates replace concrete quantities. Consequently, Poncelet wants to introduce general principles for proof and discovery to synthetic geometry that do not rely on abstract algebraic computation with indeterminates.

Unlike his teacher Monge, Poncelet was strongly opposed to use of analytical methods. Ironically, the edition of his manuscripts from Saratov as *Applications d'analyse et de géométrie* (Poncelet, 1862) shows that Poncelet did use analytical methods to work out his ideas. In any case, Poncelet became one of the main advocates of synthetic geometry in the ideological war between synthetic and analytic geometry of the nineteenth century. (Kline, 1972, p. 841f.)

The main ideas of the *Traité* are the following: Poncelet makes systematic use of central projection between different planes in three-dimensional space as a unifying geometric principle. He analyzes which properties of geometric constructions are preserved under projection and section. Moreover, Poncelet presents the theory of pole and polar with respect to a conic. The idea of greatest importance for us is Poncelet's Principle of Continuity. Related concepts are already present in the work of Kepler (1571–1630), Leibniz (1646–1716), Monge (1746–1818), and Carnot (1753–1823). Poncelet introduces it thus:

> 'Considérons une figure quelconque, dans une position générale et en quelque sorte indéterminée, parmi toutes celles qu'elle peut prendre sans violer les lois, les conditions, la liaison qui subsistent entre les diverses parties du système; supposons que, d'après ces données, on ait trouvé une ou plusieurs relations ou propriétés, soit *métriques*, soit *descriptives*,

> appartenant à la figure, en s'appuyant sur le raisonnement explicite ordinaire, c'est-à-dire par cette marche que, dans certains cas, on regarde comme seule rigoureuse. N'est-il pas évident que si, en conservant ces mêmes données, on vient à faire varier la figure primitive par degrés insensibles, ou qu'on imprime à certaines parties de cette figure un mouvement continu d'ailleurs quelconque, n'est-il pas évident que les propriétés et les relations, trouvées pour le premier système, demeureront applicables aux états successifs de ce système, pourvu toutefois qu'on ait égard aux modifications particulières qui auront pu y survenir, comme lorsque certaines grandeurs se seront évanouies, auront changé de sens ou de signe, etc., modifications qu'il sera toujours aisé de reconnaître *à priori*, et par des règles sûres?'

(Poncelet, 1865, p. xiii), or in English,

> 'Let us consider some geometrical diagram, its actual position being arbitrary and in a way indeterminate with respect to all the possible positions it could assume without violating the conditions which are supposed to hold between its different parts. Suppose now that we discover a property of this figure .... Is it not clear that if, observing the given conditions, we gradually alter the original diagram by imposing a continuous but arbitrary motion on some of its parts, the discovered properties of the original diagram will still hold throughout the successive stages of the system always provided that we note certain changes, such as that various quantities disappear, etc.—changes that can easily be recognized *a priori* and by means of sound rules?'

(Shapiro, 2005, p. 259). Geometers of the nineteenth century used the Principle of Continuity as an axiom that was self-evident and need not be proved (Kline, 1972, p. 844f.).

The novelty in Poncelet's understanding of the Principle of Continuity lies in the fact that he considers only those properties of a construction true properties that remain unaltered under continuous movement of its parts (Shapiro, 2005, p. 259). Moreover, Poncelet acknowledges that points of a construction can move infinitely far in the limit—as for example the intersection of two lines that gradually become parallel—, or can become complex—as for example the intersections of a line and a circle when line and circle move apart. Consequently, he introduces points at infinity and complex elements, albeit using a different terminology.

Poncelet calls any element imaginary ('imaginaire') that was a real element at the beginning of a continuous movement but has become inconstructible (as a real element) in its course; he calls any element ideal ('idéal') that remains real although some elements on which its construction depends have become imaginary—consider for example the real point of intersection of two complex conjugate lines. Continuous movement through instances of a construction with imaginary or ideal elements links otherwise seemingly unrelated instances of the construction and makes it possible to discover common properties of all instances (Poncelet, 1865, p. 28).

A more modern interpretation of the Principle of Continuity is given by Felix Klein (1849–1925) in his *Vorlesungen über die Entwicklung der Mathematik im 19. Jahrhundert*:

> 'Was endlich das Prinzip der Kontinuität selbst anbetrifft, so ist auch dieses mit den Mitteln der modernen Funktionentheorie nicht schwer zu begründen. Ein jeder geometrischer Satz ist analytisch auszudrücken (wenn wir Geometrie so umgrenzen, wie es damals üblich war) durch die Nullsetzung einer algebraischen oder auch nur analytischen Funktion $f(a, b, c, \dots)$ der darin in Beziehung gesetzten Stücke $a, b, c \dots$ der Figur. Das Prinzip der Kontinuität spricht dann nichts anderes aus, als daß eine analytische Funktion, die längs eines noch so kleinen Stückes ihres Bereiches verschwindet, überhaupt gleich Null ist.'

(Klein, 1926, p. 82), or in English,

> 'And as for the principle of continuity itself, it is not hard to establish rigorously by the modern theory of functions. Every geometrical statement can be expressed analytically (understanding geometry in the restricted sense that was usual in Poncelet's day) by equating to zero an algebraic or even analytic function $f(a, b, c, \dots)$ of the parts $a, b, c, \dots$ of the figure. Then the principle of continuity simply states that an analytic function which vanishes on any part, no matter how small, of its domain, must vanish everywhere.'

(Klein and Hermann, 1979, p. 75).

1.2 Continuity in Dynamic Geometry

Today, one could argue that we are in a much better position than Poncelet: Firstly, the Principle of Continuity is theoretically well-justified by analytic continuation, as outlined in the above citation from (Klein, 1926); we should no longer need to defend it against the accusation of lack of rigour with which some of Poncelet's contemporaries, notably Cauchy, attacked the Principle of Continuity (Kline, 1972, p. 843). Secondly, continuous movement of parts of a construction is no longer merely a thought experiment; we have dynamic geometry software at our disposal with which we can actually move parts of a construction. Such software allows us to perform geometric constructions, for example a compass and straightedge construction. In these constructions, we can move a free (i.e. movable) element and let the software automatically update the positions of all elements depending on the free element accordingly.

Despite its fundamental role for modern projective geometry, the Principle of Continuity seems to have been forgotten by the advent of dynamic geometry software. Much dynamic geometry software does not obey the Principle of Continuity and suffers from jumping elements—elements that behave discontinuously under movement of free elements on which they depend. This problem arises due to the ambiguity of some geometric operations:

Consider for example the intersection of a circle and a line. In general, there are two points of intersection. They correspond to the common solutions of the algebraic equation of the circle and the algebraic equation of the line. Suppose we colour one of the intersections red and the other one green. Then, if we move the line or the circle, the software needs to determine new positions for the intersections. In particular, it needs to determine which of the new intersections corresponds which of the old intersections, in order to colour them red and green appropriately. If the software makes the wrong decision, we observe that red and green point of intersection suddenly exchange positions. This jump may be somewhat unexpected, is not necessary, and lacks mathematical justification.

As another example, consider a triangle formed by three lines. Every pair of lines that meet at a vertex of the triangle has two angle bisectors. The triangle has a total of six angle bisectors. It is well-known that three of the angle bisectors meet in one point, the triangle incentre. (In fact, the six angle bisectors meet in triples in the triangle incentre and the three triangle excentres.) Suppose that we construct the incentre of a triangle formed by three lines as the intersection of three angle bisectors. If we move a vertex of the triangle, then the software needs to determine new positions for the angle bisectors. For every angle bisector, there are two possible choices. If the software makes a wrong choice, we see an angle bisector jump from one position to the perpendicular position. Consequently, if an odd number of the three angle bisectors flip, then the three angle bisectors no longer meet in one point. We can no longer observe that the three angle bisectors *always* meet in one point. If the software violates the Principle of Continuity like this, it may be less useful for discovering properties of a construction by experimentation.

Note that it may not always be possible to distinguish possible choices. If we move a circle and a line apart, then their two intersections coincide when the line touches the circle and become complex when we move beyond that point. When the points coincide, they become indistinguishable. Consequently, if we move from an initial instance where we can distinguish the intersections beyond the point at which circle and line touch each other, we can no longer tell which of the complex points of intersection corresponds to which point of intersection of the initial instance. The movement has a singularity at the point in time when circle and line touch each other and both intersections coincide. Analytic continuation of the coordinates of the intersections along any path containing such a singularity must stop at the singularity; we cannot continue beyond a singularity, at which the coordinates cease to be analytic.

For their dynamical geometry software Cinderella (Kortenkamp and Richter-Gebert, 2006), Kortenkamp and Richter-Gebert resolved the problem as follows: In order to obtain continuous respectively analytic behaviour of dependent elements, we must avoid singularities. To that end, we parameterize any movement of a free element using a time parameter t. Singularities often occur along the real time axis. Therefore, instead of performing homotopy continuation along the real time axis, we embed the real time axis as the real axis of the complex plane. This allows us to let time parameter t take detours through the complex plane between consecutive sampling points on the real time axis. If we choose such complex detours randomly, we avoid singularities with high probability. If we use a complex detour free of singularities as a homotopy path, we obtain a uniquely defined instance of the construction at its end point. If the real time axis and the homotopy path do not enclose singularities between them, then by the monodromy theorem they are equivalent w.r.t. homotopy continuation, i.e. they yield the same final instance of the construction.

In order to determine the correct final instance of a construction obtained from homotopy continuation along a given homotopy path of a free element (assumed to be free of singularities), Cinderella uses a heuristic that appears to work well in most practical examples (Richter-Gebert, 2014). To decide whether an instance of a geometric construction results from another instance under homotopy continuation along a given homotopy path of a free element is the *tracing problem* of dynamic geometry. Kortenkamp and Richter-Gebert show that, using geometric straight-line problems to describe the construction, the tracing problem of dynamic geometry is NP-hard, already for constructions using only free points, join (connecting line of two points), meet (intersection of two lines), and angle bisector as admissible geometric primitive operations (Kortenkamp and Richter-Gebert, 2002, Section 5). The *reachability problem* of dynamic geometry, to decide whether an instance of a construction can be continuously deformed into another by movement of free elements, is NP-hard, PSPACE-hard, or even undecidable, depending on the set of admissible geometric primitive operations and depending on the exact modelling of the problem (Kortenkamp and Richter-Gebert, 2002, Sections 4, 6, and 7). Note that even though the tracing problem and the reachability problem may be intractable in the worst case (if $\mathrm{P} \neq \mathrm{NP}$), we may still find algorithms for these problems with acceptable average-case complexity.

Another problem of dynamic geometry closely related to the tracing problem is the problem of finding loci. Consider a geometric construction with a semi-free element, i.e. an element constrained to movement with one degree of freedom, for example a point on a line, a point on a circle, or a line through a point. We call the semi-free element ‘mover’. Suppose that the elements of the construction behave continuously under movement of the mover. Then under the constrained one-dimensional motion of the mover, any point of the construction whose position depends on that of the mover traces a curve. We call such a point ‘tracer’. If the geometric operations by which we construct the tracer from the mover and other elements of the construction possess an algebraic representation, then the curve traced by the tracer is (part of) a plane algebraic curve. The problem of finding the locus of the tracer under movement of the mover is to determine the real connected component of the plane algebraic curve that contains the initial position of the tracer.

The goal of this thesis is to give, analyze, or extend algorithms, some of them based on previous work by Kortenkamp, Richter-Gebert, Lebmeir, and Denner-Broser, that address the tracing problem of dynamic geometry and the problem of finding loci.

1.3 Summary

We discuss algorithms that realize or exploit continuity in dynamic geometry.

In particular, we examine certified homotopy continuation of systems of plane algebraic curves (Chapter 2) and of geometric constructions (Chapter 3), generation and identification of real algebraic loci (Chapter 4 and Chapter 5), and GPU-based visualization of plane algebraic curves as domain-coloured Riemann surfaces (Chapter 6). Moreover, we provide a sampler of plane algebraic curves, which we construct as loci and visualize as domain-coloured Riemann surfaces (Chapter 7).

Chapter 2

In Chapter 2, we explain how, given a plane algebraic curve $\mathcal{C}\colon f(x,y)=0$, $x_1 \in \mathbb{C}$ not a singularity of y w.r.t. x, and $\varepsilon > 0$, we can compute $\delta > 0$ such that $|y_j(x_1) - y_j(x_2)| < \varepsilon$ for all holomorphic functions $y_j(x)$ that satisfy $f(x, y_j(x)) = 0$ in a neighbourhood of x_1 and for all x_2 with $|x_1 - x_2| < \delta$. Consequently, we obtain a certified algorithm for homotopy continuation of plane algebraic curves. The certificate is rigorous for exact real arithmetic; for floating point arithmetic, it is a soft certificate. As an example application, we study continuous deformation of closed discrete Darboux transforms. Moreover, we discuss a scheme for reliable homotopy continuation of triangular polynomial systems. A general implementation has remained elusive so far. However, the epsilon-delta bound enables us to handle the special case of systems of plane algebraic curves. The bound helps us to determine a feasible step size and paths that are equivalent w.r.t. analytic continuation to the actual paths of the variables but along which we can proceed more easily. Several examples demonstrate the practicability of our approach.

Chapter 3

In Chapter 3, we change perspective from algebraic equations describing geometric elements of a construction to algebraic equations describing elementary operations of the computation of coordinates. This leads us from the general scheme for certified homotopy continuation of triangular systems of polynomials of Section 2.5 to Denner-Broser's approach to the tracing problem in dynamic geometry (Denner-Broser, 2008, Section 6; 2013). We model the computation of coordinates as a special triangular system of polynomials, where each polynomial equation encodes an elementary operation (arithmetic or n-th root). Analogously to the epsilon-delta bound of Chapter 2, we use complex circular arithmetic to bound the range of elementary operations. We obtain an algorithm (based on (Denner-Broser, 2008, Section 6.2, Algorithm 2 and Algorithm 3)) that determines a feasible step size for homotopy continuation of a sequence of elementary operations. It certifies that we do not encounter singularities along homotopy paths and that we can choose the right branch of n-th root operations by proximity.

Chapter 4

In Chapter 4, we discuss the locus generation algorithm used by the dynamic geometry software Cinderella, and how it uses complex detours to resolve singularities. We show that the algorithm is independent of the clockwise or anticlockwise orientation of its complex detours. We conjecture that the algorithm terminates if it takes small enough complex detours and small enough steps on every complex detour. Moreover, we introduce a variant of the algorithm that possibly generates entire real connected components of real algebraic loci. Several examples illustrate its use for organic generation of real algebraic loci. Another example shows how we can apply the algorithm to simulate mechanical linkages. Apparently, the use of complex detours produces physically reasonable motion of such linkages.

Chapter 5

The locus generation algorithms of Chapter 4 yield a point cloud of points on a locus. In Chapter 5, we discuss a randomized algorithm that determines an approximate coefficient vector of a real plane algebraic curve containing a real algebraic locus from such a point cloud. It is a randomized version of the algorithm presented in (Lebmeir and Richter-Gebert, 2007). We extend the algorithm so that we can also determine an approximate rational parameterization for rational plane algebraic curves.

Chapter 6

In Chapter 6, we examine an algorithm for the visualization of domain-coloured Riemann surfaces of plane algebraic curves. The approach faithfully reproduces the topology and the holomorphic structure of the Riemann surface. We discuss how the algorithm can be implemented efficiently in OpenGL with geometry shaders, and (less efficiently) even in WebGL with multiple render targets and floating point textures. While the generation of the surface takes noticeable time in both implementations, the visualization of a cached Riemann surface mesh is possible with interactive performance. This allows us to visually explore otherwise almost unimaginable mathematical objects. As examples, we look at the complex square root and the folium of Descartes. For the folium of Descartes, the visualization reveals features of the algebraic curve that are not obvious from its equation.

Chapter 7

Chapter 7 presents a sampler of plane algebraic curves in a novel way. For every curve, we give a construction that produces the curve as a real algebraic locus.

Moreover, we visualize the corresponding domain-coloured Riemann surface using the algorithms of Chapter 6.

Chapter 8

Chapter 8 provides an outlook on open theoretical questions and planned practical applications.

Previously published work

Chapter 2, Chapter 4, and Chapter 6 are based on three articles posted as preprints on the arXiv, (Kranich, 2015a; 2015b; 2015c).

2 Homotopy Continuation of Systems of Plane Algebraic Curves

2.1 Motivation

In many geometric problems, variables depend analytically on some parameter. If we want to analyze and experiment with these problems using interactive software, whenever the user continuously modifies the parameter, we must update the dependent variables accordingly. For many applications, in doing so, the analytical relationship between variables and parameter should be preserved at all times. Therefore we need reliable algorithms for analytic continuation.

Consider for example the following problem of discrete differential geometry (Hoffmann, 2009, Section 2.6). Let there be a regular discrete curve γ in $\mathbb{CP}^1$, i.e. a polygonal chain with distinct vertices $\gamma_0, \gamma_1, \ldots, \gamma_n \in \mathbb{CP}^1$. We define the *discrete Darboux transform* $\tilde{\gamma}$ of γ with initial point $\tilde{\gamma}_0 \in \mathbb{CP}^1$ and parameter $\mu \in \mathbb{C}$ as follows: for all $j = 1, 2, \ldots, n$, let $\tilde{\gamma}_j \in \mathbb{CP}^1$ be the unique point for which the cross-ratio

$$(\gamma_{j-1}, \gamma_j; \tilde{\gamma}_j, \tilde{\gamma}_{j-1}) := \frac{(\gamma_{j-1} - \tilde{\gamma}_j)(\gamma_j - \tilde{\gamma}_{j-1})}{(\gamma_{j-1} - \tilde{\gamma}_{j-1})(\gamma_j - \tilde{\gamma}_j)} = \mu.$$

It can be shown that $\tilde{\gamma}_{j-1}$ is mapped to $\tilde{\gamma}_j$ by a unique Möbius transformation, which depends only on γ_{j-1}, γ_j, and μ, but not on $\tilde{\gamma}_j$. Hence, there exists a unique Möbius transformation M depending on γ_0, γ_1, $\ldots$, γ_n, and μ, which maps an initial point $\tilde{\gamma}_0$ to the corresponding last point $\tilde{\gamma}_n$ of $\tilde{\gamma}$. Consequently, for every choice of $\mu \in \mathbb{C}$, there are two choices of initial point $\tilde{\gamma}_0$ (counted with multiplicity) such that $\tilde{\gamma}$ is a closed polygonal chain. These are exactly the fixed points of M or, in other words, the roots of the characteristic polynomial of M. The vanishing of the characteristic polynomial establishes an algebraic (particularly analytical) relationship between μ and $\tilde{\gamma}_0$.

If we want to study closed Darboux transforms of a discrete curve γ for varying parameter μ using interactive software, then we must analytically continue $\tilde{\gamma}_0$. Otherwise we may observe sudden jumps of $\tilde{\gamma}_0$ under continuous movement of μ, which have no mathematical justification.

In practice, of course, we cannot modify a parameter continuously. Instead, we obtain a series of parameter values at a series of discrete points in time. We do not know how the parameter moves between sample points. A natural approach would be to interpolate linearly between consecutive parameter values (using a time parameter in the unit interval). However, the segment between parameter values may contain singularities beyond which analytic continuation becomes impossible. Thus it seems reasonable to analytically continue along the polygonal chain of parameter values as long as this is possible, and to deviate from that path otherwise. Such a deviation can still be interpreted as a linear interpolation between consecutive parameter values if we let the time parameter run from 0 to 1 on an arbitrary path through the complex plane instead of restricting it to the unit interval.

This is the paradigm of 'complex detours' invented by Kortenkamp and Richter-Gebert for their interactive geometry software Cinderella (Kortenkamp and Richter-Gebert, 2006). It is described in more detail in (Kortenkamp, 1999, esp. Chapter 7; Kortenkamp and Richter-Gebert, 2001b; 2002). Essentially the same concept was conceived in the context of homotopy continuation by Morgan and Sommese (Morgan and Sommese, 1987), who later named it the 'gamma trick' (Sommese and Wampler, 2005, Lemma 7.1.3 on p. 94).

Once we have chosen a path for the parameter, we must determine the right value of the dependent variable at consecutive sample points. How this can be achieved may in fact be relatively easy to see for us—just determine values in a way such that there are no jumps—but hard to see for an algorithm. The tracing problem of dynamic geometry, i.e. tracing the positions of dependent elements of a geometric construction under movement of a free element, is NP-complete already for constructions that only involve points, lines through two points, intersection of lines, and angle bisectors (Kortenkamp and Richter-Gebert, 2002).

The interactive geometry software Cinderella currently uses a heuristic for path following. Most homotopy continuation methods use a predictor-corrector approach, which is generally also heuristic. For an overview of homotopy continuation methods, consider the books by Allgower and Georg (1990) or Sommese and Wampler (2005). Lately, certified homotopy continuation methods have emerged (Beltrán and Leykin, 2012; 2013; Hauenstein and Sottile, 2012; Hauenstein et al., 2014). They are based on Smale's alpha theory (Smale, 1986).

In what follows, we derive a certified algorithm for analytic continuation of plane algebraic curves based on the following simple observation: Due to continuity, if the parameter moves little, so does the dependent variable. Hence, if we take small enough steps along the parameter path, we can choose the right value of the dependent variable based on proximity. As an application, we return to the example of continuous deformation of closed discrete Darboux transforms. Moreover, we show how the algorithm generalizes to systems of plane algebraic curves. A comparison with other approaches demonstrates the practicability of our algorithms.

2.2 Computing an epsilon-delta bound for plane algebraic curves

Theorem 2.2.1. *Let $\mathcal{C}\colon f(x,y)=0$ be a complex plane algebraic curve, where*

$$f(x,y)=\sum_{k=0}^{n} a_k(x)y^{n-k}$$

is a polynomial of degree n in y whose coefficients $a_k(x)$ are polynomials in x. Let $x_1 \in \mathbb{C}$ be a point in the complex plane at which neither the leading coefficient $a_0(x)$ nor the discriminant of $f(x,y)$ w.r.t. y vanish. Then for every $\varepsilon > 0$, we can algorithmically compute $\delta > 0$ such that

$$|y_j(x_1) - y_j(x_2)| < \varepsilon$$

for all holomorphic functions $y_j(x)$, $j = 1, 2, \dots, n$, that satisfy $f(x, y_j(x)) = 0$ in a neighbourhood of x_1 and for all x_2 with $|x_1 - x_2| < \delta$.

Remark 2.2.2. How does Theorem 2.2.1 help us to perform analytic continuation? Let ε be half the minimal distance between the y-values at x_1. Then for any x_2 less than δ away from x_1 the following holds: The y-value $y_j(x_2)$, which results from analytic continuation of $y_j(x)$ along the segment from x_1 to x_2, is closer to $y_j(x_1)$ than to any other y-value at x_1. In other words, δ provides an upper bound for the step width of parameter x such that we may match y-values on the same branch based on proximity.

Our plan for the proof of Theorem 2.2.1 is as follows: We will see that there is an upper bound of δ depending on

1. the radius of convergence of the Taylor expansion of $y_j(x)$ at x_1,
2. the modulus of the derivative of $y_j(x)$ at x_1,
3. the maximum modulus of $y_j(x)$ on a circle centred at x_1,

for $j = 1, 2, \dots, n$, respectively. We derive a formula for that upper bound and then compute bounds for its ingredients. To this end, we need the following lemmas.

Lemma 2.2.3. *Let $U \subset \mathbb{C}$ be an open subset of the complex plane, and let*

$$y_j \colon U \to \mathbb{C}$$

be holomorphic. Taylor expansion of y_j around $x_1 \in U$ yields

$$y_j(x_2) = y_j(x_1) + (x_2 - x_1)y_j'(x_1) + (x_2 - x_1)^2 R(x_2),$$

for all $x_2 \in \mathbb{C}$ such that $|x_2 - x_1| < \rho$ and sufficiently small $\rho > 0$. The remainder $R(x_2)$ satisfies

$$|R(x_2)| \leq \frac{M}{\rho(\rho - |x_2 - x_1|)}$$

where

$$M = \max_{t \in [0,2\pi]} |y_j(x_1 + \rho e^{\mathrm{i}t})|.$$

Lemma 2.2.3 is a standard result of complex analysis (Ahlfors, 1979, p. 124–126), which we therefore do not prove here.

Lemma 2.2.4 (implicit differentiation)**.** *Let $f(x, y)$ be a complex polynomial. Let $U \subset \mathbb{C}$ be an open subset of the complex plane. Let $y_j \colon U \to \mathbb{C}$ be a holomorphic function that satisfies $f(x, y_j(x)) = 0$ for all $x \in U$. Then for all $x_1 \in U$ with $f_y(x_1, y_j(x_1)) \neq 0$ it follows that*

$$y_j'(x_1) = -\frac{f_x(x_1, y_j(x_1))}{f_y(x_1, y_j(x_1))}.$$

Proof. By the chain rule, the total differential of $f(x, y_j(x)) = 0$ w.r.t. x is

$$Df(x, y_j(x)) = f_x(x, y_j(x)) + f_y(x, y_j(x)) \cdot y_j'(x) = 0.$$

Therefore

$$y_j'(x_1) = -\frac{f_x(x_1, y_j(x_1))}{f_y(x_1, y_j(x_1))}. \qquad \square$$

Lemma 2.2.5 (Fujiwara (1916, Inequality 3 on p. 168)). *Consider a polynomial*

$$p(x) = \sum_{k=0}^{n} a_k x^{n-k}$$

of degree n with complex coefficients $a_k \in \mathbb{C}$, $k = 0, 1, \dots, n$. Then all $\bar{x} \in \mathbb{C}$ with $p(\bar{x}) = 0$ satisfy

$$|\bar{x}| < 2 \max \left\{ \left| \frac{a_k}{a_0} \right|^{\frac{1}{k}} : k = 1, \dots, n \right\}.$$

Proof. Consider the inequality

$$|p(x)| \geq |a_0||x|^n - \sum_{k=1}^{n} |a_k||x|^{n-k}. \tag{2.1}$$

The RHS of (2.1) is positive if

$$|a_0||x|^n \geq 2^k |a_k||x|^{n-k}, \quad k = 1, 2, \dots, n,$$

because then

$$|a_0||x|^n > (1 - 2^{-n})|a_0||x|^n = \sum_{k=1}^{n} 2^{-k} |a_0||x|^n \geq \sum_{k=1}^{n} |a_k||x|^{n-k}.$$

Hence, $|p(x)| > 0$ if

$$|x| \geq \max \left\{ 2^k \left| \frac{a_k}{a_0} \right| \right\}^{\frac{1}{k}}$$

and thus

$$|\bar{x}| < 2 \max \left\{ \left| \frac{a_k}{a_0} \right|^{\frac{1}{k}} : k = 1, \dots, n \right\}$$

for all zeros $\bar{x} \in \mathbb{C}$ of $p(x)$. $\square$

Lemma 2.2.6 (bounds for trigonometric polynomials). *Consider a trigonometric polynomial of degree n of the form*

$$p(x_1 + \rho e^{\mathrm{i}t}) = \sum_{k=0}^{n} a_k (x_1 + \rho e^{\mathrm{i}t})^{n-k}.$$

Then

$$|p(x_1 + \rho e^{\mathrm{i}t})| \leq \sum_{k=0}^{n} |a_k|(|x_1| + |\rho|)^{n-k}.$$

Moreover, if the zeros $\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_n$ *of* $p(x)$ *satisfy* $|\bar{x}_k - x_1| > \rho$ *then*

$$|p(x_1 + \rho e^{\mathrm{i}t})| \geq |a_0| \prod_{k=0}^{n} (|\bar{x}_k - x_1| - \rho) > 0.$$

Proof. The upper bound follows from the triangle inequality. The lower bound follows from the factorization

$$p(x_1 + \rho e^{\mathrm{i}t}) = a_0 \prod_{k=0}^{n} (x_1 + \rho e^{\mathrm{i}t} - \bar{x}_k)$$

and the fact that $|x_1 + \rho e^{\mathrm{i}t} - \bar{x}_k| \geq |\rho - |\bar{x}_k - x_1||$. Note that the lower bound is positive by the assumptions that $|\bar{x}_k - x_1| > \rho$ and that p has degree n, i.e. $a_0 \neq 0$. $\square$

Proof of Theorem 2.2.1. Let $y_j(x)$, $j = 1, \ldots, n$, denote the holomorphic functions that satisfy $f(x, y_j(x)) = 0$ in a neighbourhood of x_1. By Lemma 2.2.3,

$$y_j(x_2) = y_j(x_1) + (x_2 - x_1)y_j'(x_1) + (x_2 - x_1)^2 R_j(x_2) \tag{2.2}$$

for all $x_2 \in \mathbb{C}$ such that $|x_2 - x_1| < \rho$ and sufficiently small $\rho > 0$. If we bring $y_j(x_1)$ to the LHS of (2.2), take the absolute value on both sides, and apply the triangle inequality, we see that

$$\begin{aligned} |y_j(x_1) - y_j(x_2)| &= |x_2 - x_1||y_j'(x_1) + (x_2 - x_1)R_j(x_2)| \\ &\leq |x_2 - x_1|(|y_j'(x_1)| + |x_2 - x_1||R_j(x_2)|) \\ &= |R_j(x_2)||x_2 - x_1|^2 + |y_j'(x_1)||x_2 - x_1|. \end{aligned} \tag{2.3}$$

Hence, under the above assumptions,

$$|R_j(x_2)||x_2 - x_1|^2 + |y_j'(x_1)||x_2 - x_1| - \varepsilon < 0. \tag{2.4}$$

is a sufficient condition for $|y_j(x_1) - y_j(x_2)| < \varepsilon$.

The LHS of (2.4) is strictly increasing in $|y_j'(x_1)|$ and $|R_j(x_2)|$. Therefore, if we plug in the bounds

$$|y_j'(x_1)| \leq \max_j |y_j'(x_1)| =: Y \tag{2.5}$$

and

$$|R_j(x_2)| \leq \frac{M}{\rho(\rho - |x_2 - x_1|)}$$

(see Lemma 2.2.3) into (2.4), we obtain a stronger sufficient condition for

$$|y_j(x_1) - y_j(x_2)| < \varepsilon,$$

namely

$$\begin{aligned}&\frac{M}{\rho(\rho-|x_2-x_1|)}|x_2-x_1|^2+Y|x_2-x_1|-\varepsilon<0\\ \Leftrightarrow\ & M|x_2-x_1|^2+\rho(\rho-|x_2-x_1|)(Y|x_2-x_1|-\varepsilon)<0\\ \Leftrightarrow\ & (M-\rho Y)|x_2-x_1|^2+\rho(\rho Y+\varepsilon)|x_2-x_1|-\varepsilon\rho^2<0. \qquad (2.6)\end{aligned}$$

How we can transform (2.6) into a sufficient bound on $|x_2-x_1|$ depends on the sign of $M-\rho Y$.

First case: $M-\rho Y>0$. The LHS of (2.6) describes a smile parabola in $|x_2-x_1|$ with a positive and a negative root. Since $|x_2-x_1|\geq 0$, we need only bound $|x_2-x_1|$ from above by the positive root, i.e.

$$\begin{aligned}|x_2-x_1|&<\frac{-\rho(\rho Y+\varepsilon)+\sqrt{\rho^2(\rho Y+\varepsilon)^2+4(M-\rho Y)\varepsilon\rho^2}}{2(M-\rho Y)}\\ &=\frac{\rho\left(\sqrt{(\rho Y-\varepsilon)^2+4\varepsilon M}-(\rho Y+\varepsilon)\right)}{2(M-\rho Y)}.\end{aligned}$$

Second case: $M-\rho Y<0$. The LHS of (2.6) describes a frown parabola in $|x_2-x_1|$ with one root greater than ρ and one root between 0 and ρ. Since $|x_2-x_1|<\rho$ by definition, we need only bound $|x_2-x_1|$ from above by the smaller root, i.e.

$$\begin{aligned}|x_2-x_1|&<\frac{\rho(\rho Y+\varepsilon)-\rho\sqrt{(\rho Y+\varepsilon)^2-4(\rho Y-M)\varepsilon}}{2(\rho Y-M)}\\ &=\frac{\rho\left(\sqrt{(\rho Y-\varepsilon)^2+4\varepsilon M}-(\rho Y+\varepsilon)\right)}{2(M-\rho Y)}.\end{aligned}$$

Third case: $M-\rho Y=0$. The LHS of (2.6) reduces to

$$\rho(\rho Y+\varepsilon)|x_2-x_1|-\varepsilon\rho^2<0 \quad\Leftrightarrow\quad |x_2-x_1|<\frac{\varepsilon\rho}{\rho Y+\varepsilon}.$$

This bound is asymptotically equivalent to the previous bounds as M approaches ρY. Altogether, we thus arrive at the sufficient bound

$$|x_2-x_1|<\frac{\rho\left(\sqrt{(\rho Y-\varepsilon)^2+4\varepsilon M}-(\rho Y+\varepsilon)\right)}{2(M-\rho Y)}. \qquad (2.7)$$

The RHS of (2.7) has the expected qualitative behaviour: It is strictly increasing in ε and ρ, and strictly decreasing in M and Y.

It remains to be shown that we can compute bounds for the ingredients ρ, Y, and M of (2.7).

Lemma 2.2.3 (and thus our argument) is valid if and only if ρ is smaller than the radius of convergence of the Taylor expansion of $y_j(x)$. Therefore, we must choose ρ smaller than the distance between x_1 and the singularities of $y_j(x)$, $j = 1, 2, \ldots, n$. Recall that $y_j(x)$ satisfies $f(x, y_j(x)) = 0$ in a neighbourhood of x_1, where

$$f(x, y) = \sum_{k=0}^{n} a_k(x) y^{n-k}.$$

In particular, ρ must be smaller than the distance between x_1 and the zeros of $a_0(x)$. The zeros of $a_0(x)$ are exactly the poles of $y_j(x)$. The remaining finite singularities of $y_j(x)$ are exactly the finite ramification points of $y_j(x)$. These are zeros of the discriminant of $f(x, y)$ w.r.t. y. Hence, we may choose any

$$\rho < \min\{|x_1 - x| \colon a_0(x) \cdot \Delta_y(f(x, y))(x) = 0\},$$

where $\Delta_y(f(x, y))(x)$ denotes the discriminant of f w.r.t. y.

We can compute

$$Y = \max_j |y_j'(x_1)| = \max_j \left| \frac{f_x(x_1, y_j(x_1))}{f_y(x_1, y_j(x_1))} \right|$$

by Lemma 2.2.4. Note that the denominator does not vanish by the assumption that x_1 is not a zero of the discriminant of $f(x, y)$ w.r.t. y.

Therefore, M remains to be computed or bounded from above. To that end, we can apply Lemma 2.2.5 to

$$f(x, y_j(x)) = \sum_{k=0}^{n} a_k(x) y_j(x)^k,$$

interpreted as a polynomial in $y_j(x)$. By our choice of ρ, the leading coefficient $a_0(x)$ does not vanish for all x with $|x - x_1| \leq \rho$. For those x and for all $j = 1, 2, \ldots, n$, Lemma 2.2.5 yields

$$|y_j(x)| < 2 \max \left\{ \left| \frac{a_k(x)}{a_0(x)} \right|^{\frac{1}{k}} \mid k = 1, \ldots, n \right\}.$$

Consequently,

$$M < 2 \max_{t \in [0, 2\pi]} \left\{ \left| \frac{a_k(x_1 + \rho e^{it})}{a_0(x_1 + \rho e^{it})} \right|^{\frac{1}{k}} \mid k = 1, \ldots, n \right\}.$$

By Lemma 2.2.6, we can compute upper bounds $\tilde{a}_k$ of $\max_{t \in [0, 2\pi]} |a_k(x_1 + \rho e^{it})|$ and a lower bound $\tilde{a}_0 > 0$ of $\min_{t \in [0, 2\pi]} |a_0(x_1 + \rho e^{it})|$, which are much easier to compute than these extreme values.

The zeros of $a_0(x)$ and of $\Delta_y(f(x,y))(x)$ can be computed (at least to arbitrary precision) using a root-finding algorithm. Similarly, the values $y_j(x_1)$, $j = 1, 2, \ldots, n$, can be computed (at least to arbitrary precision) by solving

$$f(x_1, y_j(x_1)) = 0$$

for $y_j(x_1)$.

Let us summarize our argument: We may choose

$$\delta = \frac{\rho\left(\sqrt{(\rho Y - \varepsilon)^2 + 4\varepsilon M} - (\rho Y + \varepsilon)\right)}{2(M - \rho Y)}, \tag{2.8}$$

where

$$\rho < \min\{|x_1 - x| \colon a_0(x) \cdot \Delta_y(f(x,y))(x) = 0\},$$
$$Y := \max_j \left|\frac{f_x(x_1, y_j(x_1))}{f_y(x_1, y_j(x_1))}\right|, \quad M := 2 \max_k \left(\frac{\tilde{a}_k}{\tilde{a}_0}\right)^{\frac{1}{k}}. \qquad \square$$

Remark 2.2.7. For Theorem 2.2.1 to hold, $f(x,y)$ needs neither be irreducible nor square-free. However, if $f(x,y)$ is not square-free, the discriminant may vanish identically and the epsilon-delta bound is no longer useful. If $f(x,y)$ is square-free but not irreducible, the epsilon-delta bound for y-values on one irreducible component may be smaller than necessary due to the influence of zeros of the discriminant of other irreducible components.

2.3 Certified homotopy continuation of plane algebraic curves

Theorem 2.2.1 enables us to solve the following problem:

Problem 2.3.1. Consider a plane algebraic curve

$$\mathcal{C} \colon f(x,y) = 0.$$

Let $x \colon [0,1] \to \mathbb{C}, \; t \mapsto x(t)$ be a monotonic (distance non-decreasing) path, i.e.

$$|x(0) - x(t_1)| \leq |x(0) - x(t_2)| \quad \text{for} \quad 0 \leq t_1 \leq t_2 \leq 1.$$

Let $y(0) \in \mathbb{C}$ satisfy $f(x(0), y(0)) = 0$. If analytic continuation of y along $x(t)$ is possible, determine the value $y(1)$ that results from initial value $y(0)$ under analytic continuation of y along $x(t)$.

The algorithm for Problem 2.3.1 follows from Remark 2.2.2:

Algorithm 2.3.2. Let $f(x,y)$, $x(t)$, and $y(0)$ be defined as in Problem 2.3.1.

1. Let $T = 0$.
2. While $T < 1$,
    a) Let ε be half the minimum distance between the y with
    $$f(x(T), y) = 0.$$
    b) Compute δ by the epsilon-delta bound of Theorem 2.2.1.
    c) Use bisection to maximize $T^* \in [T, 1]$ such that $|x(T) - x(T^*)| < \delta$.
    d) Let $T = T^*$.
    e) Let $y(T)$ be the y with $f(x(T), y) = 0$ closest to $y(0)$.
3. Output $y(1)$ and stop.

2.4 Case study: continuous deformation of closed discrete Darboux transforms

Algorithm 2.3.2 shows how the epsilon-delta bound can be used for certified homotopy continuation of plane algebraic curves. In this section, as an example application, let us return to the closed discrete Darboux transform introduced in Section 2.1.

We generally follow the exposition of Hoffmann (2009, Section 2.6) but use a slightly different definition of *cross-ratio*. (A value μ of our cross-ratio corresponds to a value $1 - \mu$ of the cross-ratio in (Hoffmann, 2009, Section 2.6) and vice versa.)

Recall the definition of discrete Darboux transform:

Definition 2.4.1 (discrete Darboux transform)**.** Let γ be a regular discrete curve in $\mathbb{CP}^1$ with vertices $\gamma_0, \gamma_1, \ldots, \gamma_n \in \mathbb{CP}^1$. We choose an initial point $\tilde{\gamma}_0 \in \mathbb{CP}^1$ and prescribe a cross-ratio $\mu \in \mathbb{C}$. The *discrete Darboux transform of γ with initial point $\tilde{\gamma}_0$ and parameter μ* is the unique discrete curve $\tilde{\gamma}$ whose vertices $\tilde{\gamma}_j$, $j = 1, 2, \ldots, n$, satisfy

$$(\gamma_{j-1}, \gamma_j; \tilde{\gamma}_j, \tilde{\gamma}_{j-1}) := \frac{(\gamma_{j-1} - \tilde{\gamma}_j)(\gamma_j - \tilde{\gamma}_{j-1})}{(\gamma_{j-1} - \tilde{\gamma}_{j-1})(\gamma_j - \tilde{\gamma}_j)} = \mu.$$

Lemma 2.4.2. *Let $a, b, d \in \mathbb{CP}^1$ be in general position. For every $\mu \in \mathbb{C}$, there exists a Möbius transformation depending on a, b, and μ that maps d to $c \in \mathbb{CP}^1$ such that $(a, b; c, d) = \mu$.*

Proof. Consider the Möbius transformation

$$M \colon x \mapsto \frac{x - a}{x - b},$$

which maps a, b, and d to 0, ∞, and d' respectively. The cross-ratio is invariant under Möbius transformations. Hence, if we denote the image of c under M as c', we want that

$$(0, \infty; c', d') = \frac{(0 - c')(\infty - d')}{(0 - d')(\infty - c')} = \frac{c'}{d'} = \mu.$$

We define the Möbius transformations

$$N\colon d' \mapsto c' = \mu d', \quad M^{-1}\colon x' \mapsto \frac{bx' - a}{x' - 1}.$$

Then the Möbius transformation

$$M^{-1} \circ N \circ M\colon d \mapsto \frac{(\mu b - a)d - (\mu - 1)ab}{(\mu - 1)d + b - \mu a}$$

maps $d \in \mathbb{CP}^1$ to $c \in \mathbb{CP}^1$ such that $(a, b; c, d) = \mu$. □

Note that $(M^{-1} \circ N \circ M)(a) = a$ and $(M^{-1} \circ N \circ M)(b) = b$, independent of μ.

Lemma 2.4.3. *There exists a Möbius transformation depending on γ_0, γ_1, …, γ_n, and μ that maps an initial point $\tilde{\gamma}_0$ of a discrete Darboux transform of γ with parameter μ to the corresponding end point $\tilde{\gamma}_n$.*

Proof. By Lemma 2.4.2, there exist Möbius transformations M_j, $j = 1, 2, \ldots, n$, depending on γ_{j-1}, γ_j, and μ that map $\tilde{\gamma}_{j-1}$ to $\tilde{\gamma}_j$. Therefore their composition $M_n \circ M_{n-1} \circ \cdots \circ M_1$ is a Möbius transformation depending on γ_0, γ_1, …, γ_n, and μ that maps $\tilde{\gamma}_0$ to $\tilde{\gamma}_n$. □

Remark 2.4.4. A discrete Darboux transform $\tilde{\gamma}$ is closed if and only if its initial point $\tilde{\gamma}_0$ is a fixed point of the Möbius transformation of Lemma 2.4.3. The Möbius transformation of Lemma 2.4.3 is of the form

$$x \mapsto \frac{ax + b}{cx + d},$$

where a, b, c, and d are polynomials in μ with complex coefficients depending on γ_0, γ_1, …, γ_n. Its fixed points are the roots of the equation

$$(cx + d)x - (ax + b) = cx^2 + (d - a)x - b = 0.$$

This equation is quadratic in x. Its degree in μ increases with the number of points of γ. Equivalently, in homogeneous coordinates, the fixed points are the eigenvectors of matrix

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix}.$$

Example 2.4.5. As a simple but interesting enough example, consider the closed discrete curve γ spanned by the fifth roots of unity,

$$\gamma_j = \mathrm{e}^{2\pi \mathrm{i} j/5}, \quad j = 0, 1, \ldots, 5.$$

The relationship between μ and the initial point $\tilde{\gamma}_0$ of a closed discrete Darboux transform $\tilde{\gamma}$ of γ is governed by the equation

$$\begin{split}\Big[\Big(\big(-3 + \sqrt{5}\big)\mu^2 + 6\mu - 3 - \sqrt{5}\Big)\tilde{\gamma}_0^2 + \Big(\big(-2 - 4\sqrt{5}\big)\mu + 1 + \sqrt{5}\Big)\tilde{\gamma}_0 \\ + \big(-3 + \sqrt{5}\big)\mu^2 + 6\mu - 3 - \sqrt{5}\Big](1 - \mu) = 0.\end{split} \tag{2.9}$$

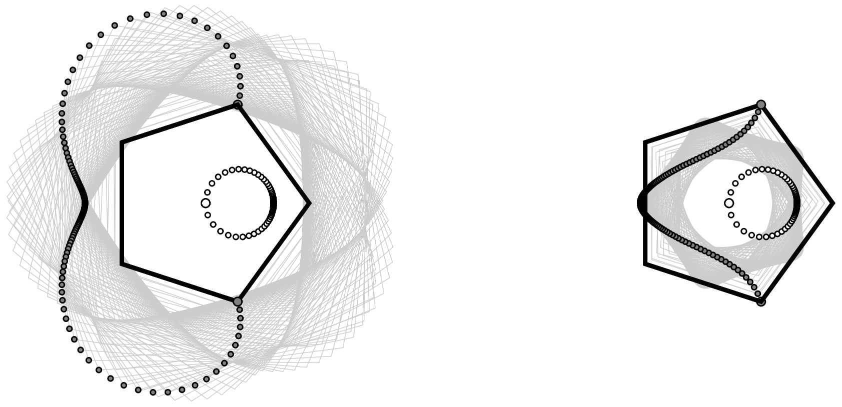


Figure 2.4.6: continuous deformation of a closed discrete Darboux transform

Equation (2.9) is quadratic in $\tilde{\gamma}_0$, cubic in μ, and has total degree 5. For almost every value of μ, exactly two values of $\tilde{\gamma}_0$ satisfy the equation. The only exceptions are $\mu = \frac{3+\sqrt{5}}{8}$ and $\mu = \infty$, which are ramification points of $\tilde{\gamma}_0$, i.e. points where there is only one value of $\tilde{\gamma}_0$, which is a root of multiplicity 2 of (2.9).

The discrete Darboux transform $\tilde{\gamma}$ of γ with initial point $\tilde{\gamma}_0 = \gamma_1$ and parameter $\mu = 0$ is identical to γ up to a rotation by $2\pi/5$, i.e.

$$\tilde{\gamma}_{j-1} = \mathrm{e}^{2\pi\mathrm{i}/5} \cdot \gamma_{j-1} = \gamma_j$$

for all $j = 0, 1, \ldots, n$. Particularly, the discrete curve $\tilde{\gamma}$ is closed.

We would like to examine how the closed Darboux transform $\tilde{\gamma}$ behaves when μ makes two full anticlockwise turns around the ramification point $\frac{3+\sqrt{5}}{8}$ on a circle through the origin centred at $\left(\frac{3+\sqrt{5}}{8}\right)/2 + \frac{1}{1000}$.

Figure 2.4.6 illustrates the behaviour of the closed Darboux transform $\tilde{\gamma}$ under the aforementioned motion of μ. We can read Figure 2.4.6 in two different ways:

Firstly, the left image shows the movement of $\tilde{\gamma}_0$ (grey points) as μ (white points) completes one full circle. Then the right image shows the movement of $\tilde{\gamma}_0$ (grey points) as μ (white points) completes another full circle. The position of the Darboux transform (black) after one turn of μ is identical to the initial position up to rotation. The final position of the Darboux transform after the second turn is absolutely identical to the initial position.

Secondly, the left image shows the movement of one choice of $\tilde{\gamma}_0$ such that $\tilde{\gamma}$ is closed as μ completes one full circle. The right image shows how the other choice of $\tilde{\gamma}_0$ such that $\tilde{\gamma}$ is closed moves at the same time. After one turn of μ we reach the initial position up to interchanged choices of $\tilde{\gamma}_0$. (In the left image, $\tilde{\gamma}_0$ moves from γ_1 to γ_4 while in the right image, $\tilde{\gamma}_0$ moves from γ_4 to γ_1.) After another turn of μ, the two choices of $\tilde{\gamma}_0$ reach their initial positions again.

Following Remark 2.2.2, the steps of μ are chosen according to the epsilon-delta bound of Theorem 2.2.1 for (2.9) with ε half the distance between the two choices of $\tilde{\gamma}_0$. As we expect, the closer μ approaches the singularity the smaller the steps become. At its rightmost position, μ is only $\frac{1}{1000}$ away from the singularity. It takes 127 steps until μ completes one full circle.

Remark 2.4.7. If we want to prevent jumps, μ and $\tilde{\gamma}_0$ cannot both be freely movable, i.e. we cannot let μ and $\tilde{\gamma}_0$ interchange their roles as movable and dependent point. Otherwise, we can force a jump as follows: We move parameter μ to $\mu = 1$. At the same time, according to (2.9), the two possible initial points of $\tilde{\gamma}$ move to the origin and the point at infinity, respectively. Without loss of generality, we assume that $\tilde{\gamma}_0$ moved to the origin. Note that $\mu = 1$ describes an irreducible component of the plane algebraic curve (2.9). This means that if we remove $\tilde{\gamma}_0$ from the origin, μ will simply rest at $\mu = 1$. Then we cannot move μ without a jump of $\tilde{\gamma}_0$ because in order to move continuously, $\tilde{\gamma}_0$ would initially have to be arbitrarily close to the origin (or the point at infinity).

Remark 2.4.8. Floating point arithmetic introduces rounding errors into the computation of $\tilde{\gamma}_0$. This has a peculiar effect: If $\tilde{\gamma}$ is closed, we know that $\tilde{\gamma}_0$ must be one of the two fixed points of the Möbius transformation M that maps the initial point $\tilde{\gamma}_0$ of $\tilde{\gamma}$ to its last point $\tilde{\gamma}_n$. In general, a Möbius transformation has one attracting and one repelling fixed point. When the fixed point is repelling, any numerical error in its position is amplified by Möbius transformation M. Therefore, the closed Darboux transform may (numerically) no longer be closed when computed naively. We have observed (see Figure 2.4.6) that we can move from one choice of $\tilde{\gamma}_0$ to the other by moving μ around a ramification point. The natural domain for the map $c\colon \mu \mapsto \tilde{\gamma}_0$ is a Riemann surface. Different choices of $\tilde{\gamma}_0$ correspond to different branches of the Riemann surface. When we compute the vertices of $\tilde{\gamma}$, we step by step compute $M \circ c = M_n \circ M_{n-1} \circ \cdots \circ M_1 \circ c$. Note that function $M \circ c$ is an example of a function on a Riemann surface that is numerically stable on one branch and numerically unstable on the other.

Luckily, we can stabilize the computation by considering the inverse Möbius transformation M^{-1}. A repelling fixed point of a Möbius transformation is an attracting fixed point of its inverse. We can step by step compute $M^{-1} \circ c = M_1^{-1} \circ M_2^{-1} \circ \cdots \circ M_n^{-1} \circ c$ to obtain the vertices of $\tilde{\gamma}$ in reversed order. Since the cross-ratio of A, B, C, D satisfies $(A, B; C, D) = (B, A; D, C)$, we need only change our algorithm very little in order to obtain $M^{-1} = M_1 \circ M_2 \circ \cdots \circ M_n$ instead of $M = M_n \circ M_{n-1} \circ \cdots \circ M_1$; we only need to reverse the order of the vertices of $\tilde{\gamma}$ before we compute the Möbius transformations.

Besides, we can determine whether $\tilde{\gamma}_0$ approximates an attracting fixed point of M by considering the derivative of M at $\tilde{\gamma}_0$. The fixed point near $\tilde{\gamma}_0$ is attracting if the absolute value of the derivative is smaller than 1.

2.5 Towards homotopy continuation of triangular systems of polynomials

In this section, we discuss a scheme for certified homotopy continuation of triangular systems of polynomials. A general implementation has remained elusive so far. However, we later follow the same scheme when we derive an algorithm for certified homotopy continuation of systems of plane algebraic curves (Algorithm 2.6.4).

Problem 2.5.1. Consider a triangular system of polynomials, without loss of generality

$$\begin{aligned} p_1(x_0, x_1) &= 0, \\ p_2(x_0, x_1, x_2) &= 0, \\ &\vdots \\ p_n(x_0, x_1, \ldots, x_n) &= 0. \end{aligned} \tag{2.10}$$

Let $x_0(0)$, $x_1(0)$, …, $x_n(0)$ be initial values that satisfy the system of equations and let $x_0(1)$ be a target value for variable x_0, i.e. a value to which x_0 should move continuously. We define function $x_0(t)$ as a parameterization of the segment between $x_0(0)$ and $x_0(1)$,

$$x_0(t) = (1-t)x_0(0) + tx_0(1).$$

By analytic continuation w.r.t. $t \in [0,1]$ we can (unless there are singularities on the curves along which we perform analytic continuation) step by step define holomorphic functions $x_1(t)$, $x_2(t)$, …, $x_n(t)$. For example, we obtain $x_1(t)$ from $p_1(x_0(t), x_1(t)) = 0$, then $x_2(t)$ from $p_2(x_0(t), x_1(t), x_2(t)) = 0$, etc.

Compute the target values $x_1(1)$, $x_2(1)$, …, $x_n(1)$ from the given polynomial system, all initial values and the first target value.

Remark 2.5.2. Any algorithm for this problem has to face the following fundamental difficulty: Among all paths $x_j(t)$ along which we perform analytic continuation to define the next path $x_k(t)$, generally only $x_0(t)$ is linear. The other paths $x_1(t)$, $x_2(t)$, …, $x_n(t)$ are almost always curvilinear—and unknown. We can at best evaluate $x_1(t)$, $x_2(t)$, …, $x_n(t)$ at finitely many discrete points in time and interpolate between the sample points. However, we must make sure that the approximate paths we obtain by discretization remain close enough to the actual paths such that they yield the same result w.r.t. analytic continuation. In particular, we must make sure that in every step no singularities lie between approximate and actual path. To make things worse, this includes singularities of variables that occur only in later equations, whose position in time may change depending on how we approximate the current step.

One way to attack this difficulty is to eliminate x_1, x_2, …, x_{n-1} from the polynomial system (2.11), e.g. using resultants. However, this approach is expensive and suffers from exponential expression swell. The resulting polynomial equation in x_0, x_n very likely has a high total degree, huge coefficients, and many (artificial) critical points. This means that we can in principle apply the method for analytic continuation of plane algebraic curves of Algorithm 2.3.2 but that in practice it will often be too expensive (see Example 2.7.4). If elimination introduces artificial critical points on the path of x_0, Algorithm 2.3.2 does not even terminate.

Instead we pursue the following idea:

Remark 2.5.3 (General scheme for homotopy continuation of triangular systems). We perform homotopy continuation of one equation after another, interpolating linearly between sample points (using a time parameter in the unit interval). In order to obtain sample points on the actual paths of the variables, we synchronize the time step. This means that we let all variables make time steps of the same size. We determine a step width such that analytic continuation by proximity is possible (as in Remark 2.2.2), and such that the linearly interpolated paths between consecutive sample points are equivalent to the actual paths of the variables w.r.t. analytic continuation. To fulfil the latter requirement, the step width must be small enough such that there are no singularities between linearly interpolated paths and actual paths. We cannot foresee whether linear paths and actual paths enclose singularities of variables that occur only in later equations. We must determine whether this is the case when we later analytically continue the respective variable. Should we find that we have 'caught' a singularity, we start over with a smaller step width. Unless there are singularities on the actual paths of variables, there is a small neighbourhood around the actual paths that is free of singularities. Eventually, after finitely many reductions of step size, the linear paths approximate the actual paths of the variables well enough such that we do not encounter singularities anymore. Then we can make one synchronized time step with all variables. We proceed until we reach time $t = 1$.

2.6 Certified homotopy continuation of systems of plane algebraic curves

In full generality, it may be very difficult to decide whether or not there are singularities between linearly interpolated paths and actual paths. (Among other things, we may want to ensure that the $(k-1)$-dimensional discriminant locus of variable x_k w.r.t. equation $p_k(x_0, x_1, \ldots, x_k) = 0$ does not intersect the polydisc around the last sample point with radii lengths of the linear paths.) Therefore, we restrict ourselves to systems of plane algebraic curves, a special case of Problem 2.5.1.

Problem 2.6.1. Consider a system of bivariate polynomials, without loss of generality

$$\begin{aligned} p_1(x_0, x_1) &= 0, \\ p_2(x_1, x_2) &= 0, \\ &\vdots \\ p_n(x_{n-1}, x_n) &= 0. \end{aligned} \tag{2.11}$$

Let $x_0(0)$, $x_1(0)$, ..., $x_n(0)$ be initial values that satisfy the system of equations and let $x_0(1)$ be a target value. We define function $x_0(t)$ as a parameterization of the segment between $x_0(0)$ and $x_0(1)$,

$$x_0(t) = (1-t)x_0(0) + tx_0(1).$$

By analytic continuation w.r.t. $t \in [0,1]$ we can (unless there are singularities on the curves along which we perform analytic continuation) step by step define holomorphic functions $x_1(t)$, $x_2(t)$, ..., $x_n(t)$. For example, we obtain $x_1(t)$ from $p_1(x_0(t), x_1(t)) = 0$, then $x_2(t)$ from $p_2(x_1(t), x_2(t)) = 0$, etc.

Compute the target values $x_1(1)$, $x_2(1)$, ..., $x_n(1)$ from the given polynomial system, all initial values and the first target value.

Before we describe an algorithm for Problem 2.6.1, we need the following lemma.

Lemma 2.6.2. *Let $\mathcal{C}\colon f(x,y) = 0$, $x_1 \in \mathbb{C}$ be defined as in Theorem 2.2.1. Let $\varepsilon > 0$. Suppose that we have determined $\delta > 0$ by the epsilon-delta bound of Theorem 2.2.1 such that*

$$|y_j(x_1) - y_j(x_2)| < \varepsilon$$

for all holomorphic functions $y_j(x)$ that satisfy $f(x, y_j(x)) = 0$ in a neighbourhood of x_1 and for all x_2 with $|x_1 - x_2| < \delta$.
Then for all x_2 with $\delta' = |x_1 - x_2| < \delta$,

$$\varepsilon' = \frac{\delta'}{\delta} \cdot \varepsilon < \varepsilon$$

satisfies

$$|y_j(x_1) - y_j(x_2)| < \varepsilon'.$$

This means that we can find a better estimate for the range of $y_j(x)$ w.r.t. an actual feasible movement of x from x_1 to x_2.

Proof. Under the assumptions of Lemma 2.6.2,

$$f(x) = \frac{y_j(\delta x + x_1) - y_j(x_1)}{\varepsilon}$$

is a holomorphic function from the open unit disk to the open unit disk. By the maximum modulus principle, we know that there exists a point on the boundary of the disk of radius δ' around x_1 where $|y_j(x) - y_j(x_1)|$ is greater or equal than at any point x with $|x - x_1| < \delta'$. Hence, there exists a point on the boundary of the disk of radius $\frac{\delta'}{\delta}$ around the origin where $|f(x)|$ is greater or equal than at any point x with $|x| < \frac{\delta'}{\delta}$. Schwarz lemma states that

$$|f(x)| \leq |x|$$

for all x in the open unit disk. Therefore

$$\frac{\varepsilon'}{\varepsilon} = \max_{t \in [0,1]} \left| f\left(\frac{\delta'}{\delta} \cdot \mathrm{e}^{2\pi i t}\right)\right| \leq \max_{t \in [0,1]} \left|\frac{\delta'}{\delta} \cdot \mathrm{e}^{2\pi i t}\right| = \frac{\delta'}{\delta},$$

and thus

$$\varepsilon' \leq \frac{\delta'}{\delta} \cdot \varepsilon$$

for all x_2 with $|x_1 - x_2| < \delta' < \delta$. □

Remark 2.6.3. Alternatively, if we plug in $\delta = \delta'$ and $\varepsilon = \varepsilon'$ into (2.8) and solve for ε', we obtain

$$\varepsilon' = \delta' \left(\tilde{y} + \frac{M\delta'}{\rho(\rho - \delta')} \right) < \varepsilon,$$

with M, ρ, $\tilde{y}$ as in the proof of Theorem 2.2.1. This yields another better estimate for the range of $y_j(x)$ w.r.t. an actual feasible movement of x from x_1 to x_2.

Algorithm 2.6.4. Consider the system of bivariate polynomials of Problem 2.6.1 with initial values $x_0(0)$, $x_1(0)$, …, $x_n(0)$ and a target value $x_0(1)$.

```
1. Define $x_0(t) = (1-t)x_0(0) + tx_0(1)$.
2. Let $T = 1$.
3. Let $\varepsilon'_0 = |x_0(0) - x_0(T)|$.
4. For all $k = 1, 2, \ldots, n$:
    a) Let $\varepsilon_k$ be half the minimum distance between the $x_k$ with
           $p_k(x_{k-1}(0), x_k) = 0.$
    b) Compute $\delta_k$ according to the epsilon-delta bound of Theorem 2.2.1 for
       $f(x,y) = p_k(x_{k-1}, x_k)$, $x_1 = x_{k-1}(0)$ and $\varepsilon = \varepsilon_k$.
    c) If $\delta_k < \varepsilon'_{k-1}$ then let $T = T/2$ and go to 3.
    d) Let $x_k(T)$ be the $x_k$ with $p_k(x_{k-1}(T), x_k) = 0$ closest to $x_k(0)$.
    e) Let $\delta'_k = |x_{k-1}(0) - x_{k-1}(T)|$.
    f) Let $\varepsilon'_k = (\delta'_k + \epsilon)/\delta_k \cdot \varepsilon_k$ with $\epsilon > 0$.
5. If $T = 1$ then output $x_1(T), x_2(T), \ldots, x_n(T)$ and stop.
6. Let $x_0(0) = x_0(T)$, $x_1(0) = x_1(T)$, …, $x_n(0) = x_n(T)$ and go to 1.
```

Theorem 2.6.5. *If the target values $x_1(1)$, $x_2(1)$, …, $x_n(1)$ of Problem 2.6.1 are well-defined, Algorithm 2.6.4 computes them in finitely many steps.*

Proof. The first two steps of Algorithm 2.6.4 are initialization steps. In step 1, we define a linear homotopy between initial value $x_0(0)$ and final value $x_0(1)$ of x_0. We first want to test whether we can perform analytic continuation of the system in a single time step. Therefore, in step 2, we set target time $T = 1$.

Steps 3–6 form the main loop of our algorithm. They are repeated until we reach time $T = 1$, in which case step 5 terminates the algorithm.

In step 3, we estimate the range of x_0 as it runs from its initial position $x_0(0)$ to its target position $x_0(T)$. Since $x_0(t)$ is linear by definition (step 1), our estimate $\varepsilon'_0 = |x_0(0) - x_0(T)|$ is exact.

Step 4 is the inner loop of our algorithm, in which we try to perform analytic continuation equation by equation of our system. Run variable k denotes the index of the equation $p_k(x_{k-1}, x_k)$ under consideration.

In steps 4a–4b, we use the epsilon-delta bound of Theorem 2.2.1 and Remark 2.2.2 to compute a feasible step width δ_k for variable x_{k-1}. If x_{k-1} moves at most δ_k then we can perform analytic continuation of x_k w.r.t. $p_k(x_{k-1}, x_k) = 0$ by selecting as $x_k(T)$ the value of x_k with $p_k(x_{k-1}(T), x_k) = 0$ closest to $x_k(0)$.

Hence, in step 4c, we test whether feasible step δ_k is smaller than an upper bound ε'_{k-1} of the range of x_{k-1} as it runs from $x_{k-1}(0)$ to $x_{k-1}(T)$.

If $\delta_k < \varepsilon'_{k-1}$, we cannot be sure that there are no singularities between the actual path of x_{k-1} and the interpolated path, i.e. the segment from $x_{k-1}(0)$ to $x_{k-1}(T)$. Our attempt to reach target time T in one step has failed. Therefore, we halve target time T and go back to step 3.

Otherwise, if $\delta_k \geq \varepsilon'_{k-1}$, the epsilon-delta bound of Theorem 2.2.1 guarantees that actual path and interpolated path of x_{k-1} are equivalent w.r.t. analytic continuation of x_k. Then in step 4d, we determine target value $x_k(T)$. By construction, $x_k(T)$ is a point on the actual path of x_k.

In steps 4e–4f, we use Lemma 2.6.2 to compute an upper bound for the range of x_k as it runs from $x_k(0)$ to $x_k(T)$. The computation is independent of whether x_{k-1} runs along actual or interpolated path. The bound ε'_k holds for both paths, particularly also for analytic continuation of x_k along the actual path of x_{k-1}.

We then proceed with analytic continuation of the next variable, if any. When we leave the inner loop (step 4), we obtain valid positions for $x_1, x_2, \ldots, x_n$ at target time T. If $T = 1$, we output the solution and stop (step 5). Otherwise, we use $x_0(T), x_1(T), \ldots, x_n(T)$ as a valid initial position from which we again try to reach target time $T = 1$ (step 6).

By the assumption that $x_1(1), x_2(1), \ldots, x_n(1)$ are well-defined, there are only finitely many singularities in a neighbourhood of the actual paths of x_0, x_1, …, x_n. The algorithm terminates after finitely many steps as eventually the interpolated paths of x_1, x_2, …, x_n approximate the actual paths well enough such that we do not encounter singularities anymore. □

2.7 Comparison with other approaches

Let us discuss more examples, which allow us to compare the performance of our algorithm with that of other approaches. (The number of steps needed by Algorithm 2.3.2 and Algorithm 2.6.4 stated below relate to an experimental implementation in Haskell.)

Example 2.7.1 (Hauenstein et al. (2014, Section 7.1))**.** Consider the Newton homotopy

$$H(x,t) = f(x) + vt$$

where $f(x) = x^2 - 1 - m$ and $v = m$ for various values of $m > -1$. The goal is to analytically continue x as t moves from 1 to 0.

In Table 2.7.2 and Table 2.7.3, we compare the performance of Algorithm 2.3.2 with that of the algorithms of Beltrán and Leykin (2013) and Hauenstein et al. (2014), for various values of m. The data for the latter algorithms is quoted from (Hauenstein et al., 2014, Table 1 and Table 2).

m	Number of steps of Algorithm 2.3.2	Number of a priori steps of Beltrán and Leykin (2013)	Number of a posteriori certified intervals of Hauenstein et al. (2014)
10	9	184	51
20	12	217	67
30	14	237	78
40	16	250	82
50	17	260	88
60	18	269	92
70	19	276	96
80	20	282	99
90	21	288	103
100	21	292	105
1000	41	395	162
2000	49	426	180
3000	54	446	191
4000	58	457	197
5000	62	468	204
10000	73	499	220
20000	87	530	238
30000	96	547	250

Table 2.7.2: Performance of Algorithm 2.3.2 in comparison with the algorithms of Beltrán and Leykin (2013) and Hauenstein et al. (2014), for various values of m. The data in the last two columns is quoted from (Hauenstein et al., 2014, Table 1).

Both Beltrán and Leykin (2013) and Hauenstein et al. (2014) present algorithms designed for certified homotopy continuation of arbitrary polynomial systems whereas Algorithm 2.3.2 can only deal with plane algebraic curves. However, the example indicates that in the univariate case Algorithm 2.3.2 may perform much better than

k	Number of steps of Algorithm 2.3.2	Number of a priori steps of Beltrán and Leykin (2013)	Number of a posteriori certified intervals of Hauenstein et al. (2014)
1	5	176	64
2	9	287	68
3	14	390	70
4	18	492	71
5	22	593	71
6	27	695	71
7	31	798	71
8	36	901	71
9	40	1003	71
10	44	1108	71

Table 2.7.3: Performance of Algorithm 2.3.2 in comparison with the algorithms of Beltrán and Leykin (2013) and Hauenstein et al. (2014), for various values of $m = -1 + 10^{-k}$. The data in the last two columns is quoted from (Hauenstein et al., 2014, Table 2).

those more general algorithms, which do not exploit the special structure of the univariate case.

Furthermore, let us elaborate on Remark 2.5.2. The following example shows that it may be better to apply Algorithm 2.6.4 to a system of plane algebraic curves than to eliminate variables and apply Algorithm 2.3.2 to the resultant.

Example 2.7.4. Consider the system of bivariate polynomials

$$\begin{aligned} p_1(x_0, x_1) &= -4 + 2x_0 + x_1 + 2x_0x_1 + x_1^2 = 0, \\ p_2(x_1, x_2) &= x_1^2 + x_2^3 = 0, \end{aligned} \tag{2.12}$$

with initial values

$$x_0(0) = 0, \quad x_1(0) = \frac{-1-\sqrt{17}}{2}, \quad x_2(0) = \left(\frac{-9-\sqrt{17}}{2}\right)^{\frac{1}{3}},$$

and target value $x_0(1) = 1$. The x_1-resultant of $p_1(x_0, x_1)$ and $p_2(x_1, x_2)$ is

$$q(x_0, x_2) = 16 - 16x_0 + 4x_0^2 + 9x_2^3 + 4x_0^2x_2^3 + x_2^6 = 0. \tag{2.13}$$

Let us compare the performance of Algorithm 2.6.4 for (2.12) with the performance of Algorithm 2.3.2 for (2.13) as x_0 moves linearly (in the unit interval) from 0 to 1. We find that Algorithm 2.6.4 subdivides once, i.e. it needs two steps. In contrast, Algorithm 2.3.2 needs six steps. One possible explanation is that $x_0 = -\frac{1}{2}$ is a singularity of (2.13) but not of (2.12). For $x_0 = -\frac{1}{2}$, (2.13) has three zeros of multiplicity two, whereas (2.12) has six simple roots. Each zero of multiplicity two of (2.13) corresponds to two simple zeros of (2.12) with differing signs of x_1.

Generally, elimination introduces artificial singularities. Due to an artificial singularity it can even happen that we cannot analytically continue the resultant: For example, the x_1-resultant of

$$\begin{aligned}\tilde{p}_1(x_0, x_1) &= -4 + 2x_0 + x_2 - 2x_0x_1 + x_1^2 = 0,\\ p_2(x_1, x_2) &= x_1^2 + x_2^3 = 0,\end{aligned}$$

has an artificial singularity at $x_0 = \frac{1}{2}$. In this case, Algorithm 2.3.2 does not terminate whereas Algorithm 2.6.4 produces the desired result.

3 Complex Tracing in Geometry

3.1 Introduction

In this section, we discuss another method for certified tracing of geometric constructions.

We have already successfully applied Algorithm 2.3.2 to continuous deformation of discrete Darboux transforms in Section 2.4.

However, we cannot directly apply the algorithms from the previous chapter to arbitrary geometric constructions. We can try to describe the lines, circles, and conics in a geometric construction as plane algebraic curves. In general, the coefficients of these curves depend on the coordinates of other elements of the construction. (Consider for example a circle through three points.) In particular, the coefficients of these curves may depend on the coordinates of two or more points that are moving simultaneously when we move a free element. Therefore, we need more than two variables in the equations describing the coordinates of intersections of such curves; in general, systems of *plane* algebraic curves do not suffice to describe a geometric construction.

In a geometric construction, every construction step can only use given elements and elements constructed in previous construction steps. Analogously, the coordinates of a new element can only depend on the coordinates of given elements and elements constructed in previous construction steps. Hence, under the assumption that the primitive operations of a construction are algebraic in the coordinates of the involved elements, we can describe the construction by a triangular system of polynomials.

A general implementation of the scheme for certified homotopy continuation of triangular systems of polynomials from Section 2.5 has remained elusive so far. However, we can attack the problem at a more fundamental level of abstraction: If we restrict primitives and geometric operations of a dynamic geometry system appropriately, then the computation of the coordinates of an element comprises a sequence of computation steps, each of which uses only arithmetic, square root or cube root as an elementary operation. For constructions with compass and straightedge, each computation step can be described using only arithmetic and square roots; if we introduce conics to our dynamic geometry system, we additionally need cube roots in order to describe the intersections of two conics. In any case, every computation step uses only finite input of the computation and intermediate results of previous computation steps. If we want to trace the computation of the coordinates of an element under variation of the input (as effected by tracing of previous construction steps or movement of a free element), we must trace each computation step. In doing so, we must select the right branch of square roots and cube roots, avoid singularities of the roots, and avoid division by zero.

We can formulate arithmetic, square root, and cube root as algebraic equations in at most three variables (one or two input variables and an output variable). The computation of the coordinates of an element can thus be described as a *special* triangular system of polynomials. Every polynomial of the special triangular system contains at most three variables. Moreover, the polynomials have only few different forms, each corresponding to one of the elementary operations. These restrictions enable us to trace the special triangular system of polynomials.

In summary, we change perspective from algebraic equations describing geometric elements to algebraic equations describing elementary operations of the computation of coordinates. This leads us from the general scheme for certified homotopy continuation of triangular systems of polynomials of Section 2.5 to Denner-Broser's approach to the tracing problem in dynamic geometry (Denner-Broser, 2008, Section 6; 2013).

3.2 Elementary operations as algebraic equations

Remark 3.2.1. Let us elaborate how we can describe arithmetic, square root and cube root using algebraic equations. If z_3 is the sum of complex numbers z_1 and z_2, we have

$$z_1 + z_2 - z_3 = 0.$$

If z_3 is the difference of complex numbers z_1 and z_2, we have

$$z_1 - z_2 - z_3 = 0.$$

If z_3 is the product of complex numbers z_1 and z_2, we have

$$z_1 z_2 - z_3 = 0.$$

If z_3 is the quotient of complex numbers z_1 and $z_2 \neq 0$, we have

$$z_1 - z_2 z_3 = 0.$$

If z_2 is the square root of z_1, we have

$$z_1 - z_2^2 = 0.$$

If z_2 is the cube root of z_1, we have

$$z_1 - z_2^3 = 0.$$

In every case, we treat the variable with the highest index as output variable and the remaining variable(s) as input variable(s).

Example 3.2.2. For example, using the above description of elementary operations, we can describe the radical expression $\sqrt{x-a}$ as a triangular system of polynomials

$$\begin{aligned} x - z_1 &= 0, \\ a - z_2 &= 0, \\ z_1 - z_2 - z_3 &= 0, \\ z_3 - z_4^2 &= 0. \end{aligned}$$

Remark 3.2.3. Addition, subtraction, and multiplication are free of singularities. Division has a pole where the divisor is 0. Arithmetic is unambiguous. Square root and cube root have a ramification point at the origin of the complex plane. The equations for square root and cube root have two and three solutions, respectively. These correspond to the different branches of square root and cube root.

3.3 Circular arithmetic

Remark 3.3.1. We want to pursue the general approach for certified homotopy continuation from Section 2.5. To that end, we need a substitute for the epsilon-delta bound of Theorem 2.2.1 that we can apply to the algebraic equations describing the elementary operations. For every elementary operation, we must bound the range of the output variable when the input variables are allowed to vary inside a certain region of the complex plane. The most natural analogue of the circular regions used in the epsilon-delta bound of Theorem 2.2.1 is to use complex circular arithmetic for the elementary operations. Some alternative choices—rectangular arithmetic, optimal circular arithmetic, and affine arithmetic—are discussed in (Denner-Broser, 2008).

In what follows, we discuss complex circular arithmetic as introduced by Gargantini and Henrici (1972). We use the n-th root operation in complex circular arithmetic due to Petković and Petković (1984).

Remark 3.3.2. We adopt the notation $(z; r)$ for a closed disk of radius $r \geq 0$ centred at $z \in \mathbb{C}$. Suppose we know that the input variables vary inside $(z_1; r_1)$ and $(z_2; r_2)$, respectively. Then we obtain the range of an elementary operation if we apply it to every possible combination of values from $(z_1; r_1)$ for the first input variable and values from $(z_2; r_2)$ for the second input variable, if any.

3.3.1 Arithmetic

Lemma 3.3.3. *The ranges of addition and subtraction of* $(z_1; r_1)$ *and* $(z_2; r_2)$ *are the disks*

$$(z_1; r_1) + (z_2; r_2) = (z_1 + z_2; r_1 + r_2),$$
$$(z_1; r_1) - (z_2; r_2) = (z_1 - z_2; r_1 + r_2).$$

In geometry, these operations are known as Minkowski sum and Minkowski difference.

Proof. We show

$$(z_1; r_1) + (z_2; r_2) \subset (z_1 + z_2; r_1 + r_2)$$

and

$$(z_1; r_1) + (z_2; r_2) \supset (z_1 + z_2; r_1 + r_2).$$

To that end, consider

$$z_k + \rho_k r_k \mathrm{e}^{\mathrm{i}\varphi_k} \in (z_k; r_k), \quad 0 \leq \rho_k \leq 1, \quad 0 \leq \varphi_k \leq 2\pi, \quad k = 1, 2.$$

Then $\left|z_1 + \rho_1 r_1 \mathrm{e}^{\mathrm{i}\varphi_1} + z_2 + \rho_2 r_2 \mathrm{e}^{\mathrm{i}\varphi_2} - (z_1 + z_2)\right| = \left|\rho_1 r_1 \mathrm{e}^{\mathrm{i}\varphi_1} + \rho_2 r_2 \mathrm{e}^{\mathrm{i}\varphi_2}\right|$ and

$$\left|\rho_1 r_1 \mathrm{e}^{\mathrm{i}\varphi_1} + \rho_2 r_2 \mathrm{e}^{\mathrm{i}\varphi_2}\right| \leq \left|\rho_1 r_1 \mathrm{e}^{\mathrm{i}\varphi_1}\right| + \left|\rho_2 r_2 \mathrm{e}^{\mathrm{i}\varphi_2}\right| = \rho_1 r_1 + \rho_2 r_2 \leq r_1 + r_2.$$

This shows $(z_1; r_1) + (z_2; r_2) \subset (z_1 + z_2; r_1 + r_2)$. Conversely, consider

$$z_1 + z_2 + \rho(r_1 + r_2)\mathrm{e}^{\mathrm{i}\varphi} \in (z_1 + z_2; r_1 + r_2), \quad 0 \leq \rho \leq 1, \quad 0 \leq \varphi \leq 2\pi.$$

Then $z_1 + z_2 + \rho(r_1 + r_2)\mathrm{e}^{\mathrm{i}\varphi} = z_1 + \rho r_1 \mathrm{e}^{\mathrm{i}\varphi} + z_2 + \rho r_2 \mathrm{e}^{\mathrm{i}\varphi} \in (z_1; r_1) + (z_2; r_2)$. We have thus proved the statement for addition. The statement for subtraction follows analogously, since $-(z_2; r_2) = (-z_2; r_2)$. □

Lemma 3.3.4. *The range of multiplication of* $(z_1; r_1)$ *and* $(z_2; r_2)$ *is not a disk. However, we can determine a disk centred at* $z_1 z_2$ *that contains the range. We obtain*

$$(z_1; r_1)(z_2; r_2) \subset (z_1 z_2; r_1 r_2 + r_1|z_2| + r_2|z_1|).$$

Proof. Consider

$$z_k + \rho_k r_k \mathrm{e}^{\mathrm{i}\varphi_k} \in (z_k; r_k), \quad 0 \le \rho_k \le 1, \quad 0 \le \varphi_k \le 2\pi, \quad k = 1, 2.$$

Then

$$\begin{aligned}(z_1 + \rho_1 r_1 \mathrm{e}^{\mathrm{i}\varphi_1})(z_2 + \rho_2 r_2 \mathrm{e}^{\mathrm{i}\varphi_2}) &= z_1 z_2 + z_1 \rho_2 r_2 \mathrm{e}^{\mathrm{i}\varphi_2} + z_2 \rho_1 r_1 \mathrm{e}^{\mathrm{i}\varphi_1} + \rho_1 r_1 \mathrm{e}^{\mathrm{i}\varphi_1} \rho_2 r_2 \mathrm{e}^{\mathrm{i}\varphi_2} \\ &= z_1 z_2 + z_1 \rho_2 r_2 \mathrm{e}^{\mathrm{i}\varphi_2} + z_2 \rho_1 r_1 \mathrm{e}^{\mathrm{i}\varphi_1} + \rho_1 \rho_2 r_1 r_2 \mathrm{e}^{\mathrm{i}(\varphi_1+\varphi_2)}\end{aligned}$$

and

$$\begin{aligned}&\left| z_1 \rho_2 r_2 \mathrm{e}^{\mathrm{i}\varphi_2} + z_2 \rho_1 r_1 \mathrm{e}^{\mathrm{i}\varphi_1} + \rho_1 \rho_2 r_1 r_2 \mathrm{e}^{\mathrm{i}(\varphi_1+\varphi_2)} \right| \\ &\le \left| z_1 \rho_2 r_2 \mathrm{e}^{\mathrm{i}\varphi_2} \right| + \left| z_2 \rho_1 r_1 \mathrm{e}^{\mathrm{i}\varphi_1} \right| + \left| \rho_1 \rho_2 r_1 r_2 \mathrm{e}^{\mathrm{i}(\varphi_1+\varphi_2)} \right| \\ &\le |z_1|\, r_2 + |z_2|\, r_1 + r_1 r_2\end{aligned}$$

□

Lemma 3.3.5. *The reciprocal of* $(z; r)$ *not containing* 0 *is the disk*

$$(z; r)^{-1} = (\bar{z}/(|z|^2 - r^2); r/(|z|^2 - r^2)),$$

where $\bar{z}$ *denotes the complex conjugate of* z.

Proof. The map $z \mapsto 1/z$ is a Möbius transformation. It maps finite disks not containing the origin to finite disks not containing the origin. Therefore, the reciprocal of $(z; r)$ not containing 0 is a finite disk. (Note that the centre of the reciprocal is not the reciprocal of the centre.) It suffices to show that every point of the boundary of $(z; r)$,

$$z + r\mathrm{e}^{\mathrm{i}\varphi}, \quad 0 \le \varphi \le 2\pi,$$

is mapped to a point at distance $r/(|z|^2 - r^2)$ from $\bar{z}/(|z|^2 - r^2)$. We have

$$\begin{aligned}\left| \frac{1}{z + r\mathrm{e}^{\mathrm{i}\varphi}} - \frac{\bar{z}}{|z|^2 - r^2} \right| &= \frac{1}{|z|^2 - r^2} \cdot \left| \frac{|z|^2 - r^2 - \bar{z}(z + r\mathrm{e}^{\mathrm{i}\varphi})}{z + r\mathrm{e}^{\mathrm{i}\varphi}} \right| \\ &= \frac{r}{|z|^2 - r^2} \cdot \left| \frac{r + \bar{z}\mathrm{e}^{\mathrm{i}\varphi}}{z + r\mathrm{e}^{\mathrm{i}\varphi}} \right| \\ &= \frac{r}{|z|^2 - r^2} \cdot \left| \mathrm{e}^{\mathrm{i}\varphi} \cdot \frac{\overline{z + r\mathrm{e}^{\mathrm{i}\varphi}}}{z + r\mathrm{e}^{\mathrm{i}\varphi}} \right| \\ &= \frac{r}{|z|^2 - r^2}.\end{aligned}$$

□

Lemma 3.3.6. *The range of division of* $(z_1; r_1)$ *by* $(z_2; r_2)$ *not containing* 0 *is the range of multiplication of* $(z_1; r_1)$ *and the reciprocal of* $(z_2; r_2)$. *It is not a disk, but contained in a disk defined analogously to that for multiplication.*

Proof. This follows directly from $(z_1; r_1)/(z_2; r_2) = (z_1; r_1)(z_2; r_2)^{-1}$. □

The above lemmas motivate the following definition:

Definition 3.3.7 (Circular arithmetic)**.** We define circular arithmetic as follows: For every $(z; r)$, $(z_1; r_1)$, $(z_2; r_2)$,

$$\begin{aligned}(z_1; r_1) + (z_2; r_2) &:= (z_1 + z_2; r_1 + r_2),\\ (z_1; r_1) - (z_2; r_2) &:= (z_1 - z_2; r_1 + r_2),\\ (z_1; r_1)(z_2; r_2) &:= (z_1 z_2; r_1 r_2 + r_1|z_2| + r_2|z_1|),\\ (z; r)^{-1} &:= (\bar{z}/(|z|^2 - r^2); r/(|z|^2 - r^2)), \quad \text{if } 0 \notin (z; r),\\ (z_1; r_1)/(z_2; r_2) &:= (z_1; r_1)(z_2; r_2)^{-1}, \quad \text{if } 0 \notin (z_2; r_2).\end{aligned}$$

Remark 3.3.8. Circular arithmetic as defined in Definition 3.3.7 is commutative and associative. Instead of the distributive law, we have

$$(z_1; r_1)\left((z_2; r_2) + (z_3; r_3)\right) \subset (z_1; r_1)(z_2; r_2) + (z_1; r_1)(z_3; r_3);$$

equality holds if $r_1 = 0$ (Gargantini and Henrici, 1972, p. 308).

3.3.2 n-th root

The range of the n-th root of $(z; r)$, $n \geq 2$, is not a disk. However, we can determine a disk centred at $\sqrt[n]{z}$ that contains the range—one such disk for every branch of the n-th root (Gargantini and Henrici, 1972, for $n = 2$; Petković and Petković, 1980, for $n \geq 2$). We obtain the following result:

Lemma 3.3.9.

$$\sqrt[n]{(z; r)} \subset (\sqrt[n]{z};\ \sqrt[n]{|z|} - \sqrt[n]{|z| - r})$$

if $(z; r)$ *does not contain* 0 *and*

$$\sqrt[n]{(z; r)} \subset (0,\ \sqrt[n]{|z| + r})$$

otherwise.

The statement is illustrated by Figure 3.3.10.

Proof. If $0 \in (z; r)$, then the point of $(z; r)$ farthest from 0 has magnitude $|z| + r$. The magnitude of the n-th root of a complex number is the n-th root of the magnitude of the complex number. The non-negative branch of the n-th root is strictly increasing over the non-negative reals. Consequently, the n-th root of any complex number in $(z; r)$ has a magnitude smaller or equal $\sqrt[n]{|z| + r}$. Hence, $\sqrt[n]{(z; r)} \subset (0,\ \sqrt[n]{|z| + r})$.

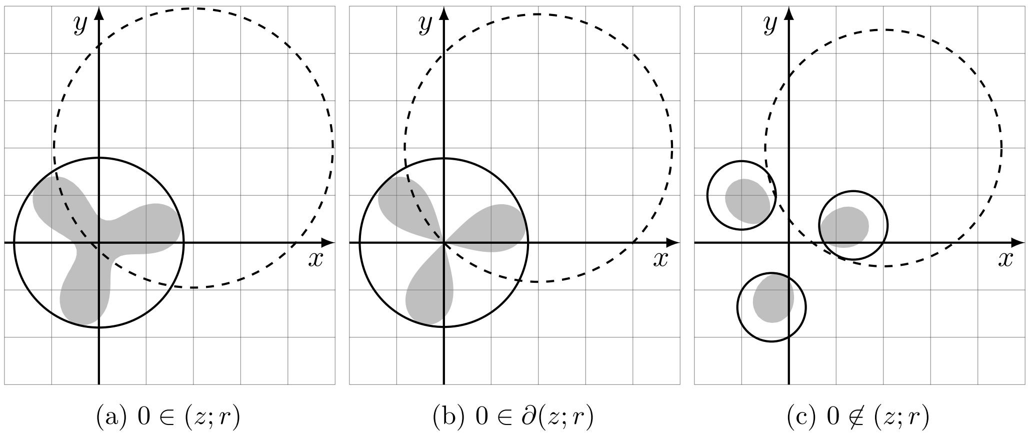


Figure 3.3.10: An illustration of Lemma 3.3.9, bounding the n-th root in complex circular arithmetic, for $n = 3$. The three branches of the cube root map the closed disk $(z; r)$ (dashed boundary) to the grey region(s) contained in the disk(s) $\sqrt[n]{(z; r)}$ (solid boundary).

If $0 \notin (z; r)$, then the map $x \mapsto \sqrt[n]{z} - \sqrt[n]{x}$ is holomorphic on $(z; r)$ for any branch of the n-th root. By the maximum modulus principle, the maximum absolute value of $\sqrt[n]{z} - \sqrt[n]{x}$ is attained on the boundary of $(z; r)$. We can write any complex number on the boundary of $(z; r)$ as $z + r\mathrm{e}^{\mathrm{i}\varphi}$. We obtain

$$\begin{aligned} \left| \sqrt[n]{z} - \sqrt[n]{z + r\mathrm{e}^{\mathrm{i}\varphi}} \right| &= \left| \sqrt[n]{z} \right| \cdot \left| 1 - \sqrt[n]{1 + r/z \cdot \mathrm{e}^{\mathrm{i}\varphi}} \right| \\ &= \sqrt[n]{|z|} \cdot \left| 1 - \sqrt[n]{1 + r/|z| \cdot \mathrm{e}^{\mathrm{i}(\varphi - \arg(z))}} \right|. \end{aligned}$$

We focus on the factor $\left| 1 - \sqrt[n]{1 + r/|z| \cdot \mathrm{e}^{\mathrm{i}(\varphi - \arg(z))}} \right|$. It is of the form

$$\left| 1 - \sqrt[n]{1 + p \cdot \mathrm{e}^{\mathrm{i}\theta}} \right|, \quad 0 \le p < 1, \quad 0 \le \theta \le 2\pi.$$

We can express $1 - \sqrt[n]{1 + p \cdot \mathrm{e}^{\mathrm{i}\theta}}$ in polar coordinates and find the real-valued roots of the θ-derivative of the square of its absolute value, $\theta = 0$ and $\theta = \pi$. Thus we obtain the extrema of $\left| 1 - \sqrt[n]{1 + p \cdot \mathrm{e}^{\mathrm{i}\theta}} \right|$, which are $1 - \sqrt[n]{1 - p}$ and $\sqrt[n]{1 + p} - 1$. Since $2 - \sqrt[n]{1 - p} - \sqrt[n]{1 + p}$ attains its minimum 0 at $p = 0$, we find that

$$1 - \sqrt[n]{1 - p} \ge \sqrt[n]{1 + p} - 1.$$

Hence,

$$\max_{\varphi - \arg(z)} \left| 1 - \sqrt[n]{1 + r/|z| \cdot \mathrm{e}^{\mathrm{i}(\varphi - \arg(z))}} \right| = 1 - \sqrt[n]{1 - r/|z|}.$$

Therefore,

$$\begin{aligned}\left|\sqrt[n]{z}-\sqrt[n]{z+r\mathrm{e}^{\mathrm{i}\varphi}}\right| &= \sqrt[n]{|z|}\cdot\left|1-\sqrt[n]{1+r/|z|\cdot\mathrm{e}^{\mathrm{i}(\varphi-\arg(z))}}\right| \\ &\leq \sqrt[n]{|z|}\cdot(1-\sqrt[n]{1-r/|z|}) \\ &= \sqrt[n]{|z|}-\sqrt[n]{|z|-r}\end{aligned}$$

and this yields the claim. □

Definition 3.3.11. We define the n-th root in circular arithmetic as follows:

$$\sqrt[n]{(z;r)} := (\sqrt[n]{z};\ \sqrt[n]{|z|}-\sqrt[n]{(|z|-r)})$$

if $0 \notin (z;r)$ and

$$\sqrt[n]{(z;r)} := (0;\ \sqrt[n]{(|z|+r)})$$

otherwise.

The algorithm of the next section relies on the following lemma (cf. (Petković and Petković, 1984, Theorem 1)):

Lemma 3.3.12 (Separation of branches). *If $(z;r)$ does not contain 0 and*

$$r/|z| < 1-(1-\sin(\pi/n))^n,$$

then the disks of Definition 3.3.11 that contain the different branches of $\sqrt[n]{(z;r)}$ do not intersect or touch each other.

Proof. It suffices to show that the disks of adjacent branches of $\sqrt[n]{(z;r)}$ do not intersect or touch each other. We denote the centres of such disks by $\sqrt[n]{z}$ and $\mathrm{e}^{2\pi\mathrm{i}/n}\cdot\sqrt[n]{z}$. The distance between the centres is

$$\begin{aligned}\left|\sqrt[n]{z}-\mathrm{e}^{2\pi\mathrm{i}/n}\sqrt[n]{z}\right| &= \left|\sqrt[n]{z}\right|\cdot\left|1-\mathrm{e}^{2\pi\mathrm{i}/n}\right| \\ &= \sqrt[n]{|z|}\sqrt{(1-\mathrm{e}^{2\pi\mathrm{i}/n})(1-\mathrm{e}^{-2\pi\mathrm{i}/n})} = \sqrt[n]{|z|}\sqrt{2-\mathrm{e}^{2\pi\mathrm{i}/n}-\mathrm{e}^{-2\pi\mathrm{i}/n}} \\ &= \sqrt[n]{|z|}\sqrt{2-2\cos(2\pi/n)} = \sqrt[n]{|z|}\sqrt{4(\sin(\pi/n))^2} \\ &= 2\sqrt[n]{|z|}\sin(\pi/n)\end{aligned}$$

According to Definition 3.3.11, the disks have radius

$$\sqrt[n]{|z|}-\sqrt[n]{|z|-r} = \sqrt[n]{|z|}\cdot(1-\sqrt[n]{1-r/|z|}).$$

The disks do not intersect or touch each other, if and only if their centres are more than twice the radius apart. Then we have

$$\begin{aligned}
& \left|\sqrt[n]{z} - \mathrm{e}^{2\pi \mathrm{i}/n} \cdot \sqrt[n]{z}\right| > 2 \cdot \sqrt[n]{|z|} \cdot \left(1 - \sqrt[n]{1 - r/|z|}\right) \\
\Leftrightarrow \quad & 2\sqrt[n]{|z|}\sin(\pi/n) > 2 \cdot \sqrt[n]{|z|} \cdot \left(1 - \sqrt[n]{1 - r/|z|}\right) \\
\Leftrightarrow \quad & \sin(\pi/n) > 1 - \sqrt[n]{1 - r/|z|} \\
\Leftrightarrow \quad & \sqrt[n]{1 - r/|z|} > 1 - \sin(\pi/n) \\
\Leftrightarrow \quad & 1 - r/|z| > (1 - \sin(\pi/n))^n \\
\Leftrightarrow \quad & r/|z| < 1 - (1 - \sin(\pi/n))^n.
\end{aligned}$$

□

3.3.3 Inclusion monotonicity

For circular arithmetic as defined in Definition 3.3.7 inclusion monotonicity holds:

Theorem 3.3.13 (Inclusion monotonicity of circular arithmetic). *Let $A_1 = (z_1; r_1)$, $A_2 = (z_2; r_2)$, $B_1 = (z_3; r_3)$, and $B_2 = (z_4; r_4)$ be closed disks such that $A_1 \subset B_1$ and $A_2 \subset B_2$. Then circular arithmetic as defined in Definition 3.3.7 satisfies*

$$\begin{aligned}
A_1 + A_2 &\subset B_1 + B_2, \\
A_1 - A_2 &\subset B_1 - B_2, \\
A_1 A_2 &\subset B_1 B_2, \\
A_1 / A_2 &\subset B_1 / B_2, \ \text{if } 0 \notin B_2.
\end{aligned}$$

For a proof, see (Alefeld and Herzberger, 1983, Theorem 9 on p. 56).

Remark 3.3.14. For division-free computations, a weaker version of inclusion monotonicity, where $z_1 = z_3$ and $z_2 = z_4$ in the definition of A_1, A_2, B_1, B_2, suffices for Algorithm 3.4.1 that we discuss in the next section: For addition, subtraction, and multiplication, the resulting disks are centred at the sum, difference, and product of the centres of the operands, respectively. The reciprocal, in terms of which we defined division, is the only operation that does not map the centre of its operand to the centre of the resulting disk.

Lemma 3.3.15 (Inclusion monotonicity of n-th root). *Let $(z_1; r_1)$ and $(z_2; r_2)$ be closed disks such that $(z_1; r_1) \subset (z_2; r_2)$. If $0 \in (z_1; r_1)$ or $0 \notin (z_2; r_2)$, then*

$$\sqrt[n]{(z_1; r_1)} \subset \sqrt[n]{(z_2; r_2)}$$

for any branch of the n-th root.

Proof. We know that $(z_1; r_1) \subset (z_2; r_2)$ if and only if $|z_2 - z_1| \leq r_2 - r_1$. By the reverse triangle inequality, we obtain

$$||z_2| - |z_1|| \leq |z_2 - z_1| \leq r_2 - r_1. \qquad (*)$$

We distinguish two cases, depending on whether 0 is contained in none or both of $(z_1; r_1)$ and $(z_2; r_2)$.

Firstly, let $0 \notin (z_1; r_1)$ and $0 \notin (z_2; r_2)$. We have

$$\begin{aligned}
& & \sqrt[n]{(z_1; r_1)} &\subset \sqrt[n]{(z_2; r_2)} \\
&\Leftrightarrow & (\sqrt[n]{z_1};\, \sqrt[n]{|z_1|} - \sqrt[n]{|z_1| - r_1}) &\subset (\sqrt[n]{z_2};\, \sqrt[n]{|z_2|} - \sqrt[n]{|z_2| - r_2}) \\
&\Leftrightarrow & |\sqrt[n]{z_2} - \sqrt[n]{z_1}| &\leq \sqrt[n]{|z_2|} - \sqrt[n]{|z_2| - r_2} - \left(\sqrt[n]{|z_1|} - \sqrt[n]{|z_1| - r_1}\right) \\
&\Leftrightarrow & \sqrt[n]{|z_2| - r_2} - \sqrt[n]{|z_1| - r_1} &\leq (|\sqrt[n]{z_2}| - |\sqrt[n]{z_1}|) - |\sqrt[n]{z_2} - \sqrt[n]{z_1}| \leq 0 \\
&\Leftrightarrow & |z_2| - r_2 &\leq |z_1| - r_1 \\
&\Leftrightarrow & |z_2| - |z_1| &\leq r_2 - r_1
\end{aligned}$$

and the last inequality is satisfied by $(*)$.

Secondly, let $0 \in (z_1; r_1)$ and $0 \in (z_2; r_2)$. We have

$$\begin{aligned}
& & \sqrt[n]{(z_1; r_1)} &\subset \sqrt[n]{(z_2; r_2)} \\
&\Leftrightarrow & (0;\, \sqrt[n]{|z_1| + r_1}) &\subset (0;\, \sqrt[n]{|z_2| + r_2}) \\
&\Leftrightarrow & 0 &\leq \sqrt[n]{|z_2| + r_2} - \sqrt[n]{|z_1| + r_1} \\
&\Leftrightarrow & |z_1| + r_1 &\leq |z_2| + r_2 \\
&\Leftrightarrow & |z_1| - |z_2| &\leq r_2 - r_1
\end{aligned}$$

and the last inequality is satisfied by $(*)$. □

Corollary 3.3.16 (Inclusion monotonicity and separation of branches)**.** *Let $(z_1; r_1)$ and $(z_2; r_2)$ be closed disks such that $(z_1; r_1) \subset (z_2; r_2)$. If $(z_2; r_2)$ satisfies the conditions of Lemma 3.3.12, then so does $(z_1; r_1)$.*

Proof. If Lemma 3.3.12 is applicable to $(z_2; r_2)$, then $0 \notin (z_2; r_2)$. By Lemma 3.3.15, $\sqrt[n]{(z_1; r_1)} \subset \sqrt[n]{(z_2; r_2)}$ for any branch of the n-th root. If the disks $\sqrt[n]{(z_2; r_2)}$ for different branches of the n-th root do not intersect or touch each other, then neither do the disks $\sqrt[n]{(z_1; r_1)}$ contained therein. □

3.4 Algorithm

We obtain the following algorithm for certified tracing of a sequence of elementary operations. It is based on (Denner-Broser, 2008, Section 6.2, Algorithm 2 and Algorithm 3).

Algorithm 3.4.1. Let a sequence of elementary operations that can be described by a triangular system of polynomials be given. Without loss of generality, we assume that we know the values of all input and output variables from a previous computation of the sequence of elementary operations. (Otherwise, we perform the computation without tracing, choosing arbitrary branches for root operations, and remember the values of all input and output variables.)

The goal of the algorithm is to perform homotopy continuation of the sequence of elementary operations as some (independent) input variables move linearly from their current value to a new value. For every changing input variable, we setup a linear homotopy between current value and new value in time parameter $t \in [0, 1]$. Only input variables that do not appear as output variables in the computation are allowed to change.

We proceed as follows:

1. We set $t_0 = 0$ and $h = 1$.

2. We set $t_1 = t_0 + h$.

3. For every changing input variable, we compute the value at $t = t_1$ according to the linear homotopy for that variable. To the input variable, we assign the closed disk centred at the current value of the variable whose radius is the distance between current value and the value at $t = t_1$.

4. Consider the input variables whose value does not change and that do not appear as an output variable in the computation. To every such variable, we assign the closed disk of radius 0 centred at the current value of the variable.

5. We proceed with the computation of the sequence of elementary operations in circular arithmetic, until we reach a division, a root operation, or the end of the sequence. If we reach a division, we go to step 6. If we reach a root operation, we go to step 7. If we reach the end of the sequence, we go to step 8.

6. If the disk assigned to the input variable of the division contains 0, we might encounter a singularity. Then we set $h = h/2$ and go to step 2. Otherwise, we perform the division in circular arithmetic and go to step 5.

7. Let $(z; r)$ denote the disk assigned to the input variable of the root operation. If the operation is an n-th root, $0 \notin (z; r)$, and $r/|z| < 1 - (1 - \sin(\pi/n))^n$, then we perform the root operation in circular arithmetic. In doing so, we choose the branch of $\sqrt[n]{z}$ closest to the current value of the output variable.

   Otherwise, if $0 \in (z; r)$ or $r/|z| \geq 1 - (1 - \sin(\pi/n))^n$, we might encounter a singularity or we might not choose the right branch of $\sqrt[n]{z}$ by proximity. Then we set $h = h/2$ and go to step 2. Otherwise, we perform the root operation in circular arithmetic and go to step 5.

8. Using circular arithmetic, we have certified that if we make a step from t_0 to $t_1 = t_0 + h$, we do not encounter a singularity and we can determine the right branch of root operations by proximity. Hence, we update the changing input variables to their value at $t = t_1$. Then we perform the computation in ordinary arithmetic; in the process, for every root operation, we select the branch that yields the value closest to the current value of the corresponding output variable.

9. If $t_1 = 1$, we output the values of all variables and stop. Otherwise, we set $t_0 = t_1$, $h = 1 - t_0$, and go to step 2.

Remark 3.4.2. If homotopy continuation along the linear homotopy between current and new values of the input variables is at all possible, then no singularities lie exactly on that homotopy path nor in a small neighbourhood of the homotopy path. Therefore, after finitely many subdivisions of step width h, we can make a successful step. Eventually, after finitely many steps, Algorithm 3.4.1 terminates.

Remark 3.4.3. If the homotopy path contains a singularity, Algorithm 3.4.1 does not terminate. As we approach the singularity, we need to make smaller and smaller steps lest any disk used in circular arithmetic contains 0.

Remark 3.4.4. We can minimize the probability that we encounter a singularity, if we choose random complex detours for time parameter t in a small neighbourhood of $[0, 1]$.

Theorem 3.4.5. *If Algorithm 3.4.1 terminates, then it returns those values of the output variables that are produced by homotopy continuation along a linear homotopy between old and new values of the changing input variables.*

Proof. In step 3 of Algorithm 3.4.1, we let every changing input variable range in a closed disk that is centred at the current value of the variable and contains the new value of the variable on its boundary. In step 4, we setup closed disks of radius 0 that analogously represent the ranges of constant input variables.

In steps 5–7, we use circular arithmetic as defined in Definition 3.3.7 to perform the elementary operations of the computation. Thus we address the problem that, under homotopy continuation, the paths of the output variables are curvilinear and unknown. We may not know the actual paths of the output variables, but circular arithmetic yields closed disks that contain the paths.

In steps 6–7, we test whether a disk, in which the path of a variable used as radicand or divisor ranges, contains a singularity. If this is the case, we need to restart the computation using a smaller step width. Otherwise, if the disk is free from singularities, any two paths running in its interior between the same end points are equivalent w.r.t. homotopy continuation of the division or root operation. Circular arithmetic yields disks containing the corresponding range of the output variable of the division or root operation.

For a root operation, we additionally check whether the disks in which the different branches of the output variable range intersect or touch each other. In this case, we cannot assign the correct branch based on proximity; we need to restart the computation using a smaller step width. Otherwise, if the disks containing the branches of the output variable do not intersect or touch each other, we can assign the correct branch based on proximity: In this case, as the input variable moves away from its current position in the disk containing its path, any branch of the output variable remains closer to its initial value than to the initial value of any other branch. Consequently, the right branch for the new value of the output variable is the branch that yields the new value of the output variable closest to its current value.

Hence, if we reach step 8, we have certified that we can make a step from t_0 to t_1 without encountering a singularity and that we can choose the right branches of root operations by proximity. Therefore, we let the changing input variables make this step. Then we perform the computation in ordinary arithmetic. For every root operation, we select the right branch based on proximity. This yields the correct intermediate values of the output variables at time $t = t_1$.

The algorithm proceeds until we reach time $t_1 = 1$, when all changing input variables attain their new values. Since the intermediate values of the output variables are correct at every step, so are the output variables at time $t_1 = 1$. $\square$

3.5 Example

We consider intersection of the unit circle

$$x^2 + y^2 - 1 = 0$$

with a vertical line, initially

$$x = 0.$$

We want to trace the intersection with coordinates $x = 0$, $y = 1$ as the vertical line moves from $x = 0$ to $x = 2$ and back to $x = 0$. For the y-coordinate, we obtain the radical expression

$$y = \sqrt{1 - x^2}.$$

When the vertical line moves, we always want to select the right branch of the square root for the y-coordinate so that the y-coordinate behaves like an analytic function of time.

We apply Algorithm 3.4.1 to the computation of $y = \sqrt{1 - x^2}$, which we express by the following sequence of elementary operations:

1. $x := x(t)$,
2. $a := 1$,
3. $b := x \cdot x$,
4. $c := a - b$,
5. $y := \sqrt{c}$.

We set up the linear homotopy

$$x(t) = (1 - t) \cdot 0 + 2t = 2t.$$

Table 3.5.1 shows the first iterations of Algorithm 3.4.1. If $x = x(t)$, the abscissa of the vertical line, moves linearly from 0 to 2 along the real axis, then we encounter a singularity at $x = 1$. At this point, the two branches of $\sqrt{1 - x^2}$ coincide. Along this path of x, we therefore cannot obtain analytic behaviour of y. Algorithm 3.4.1 makes a successful step whenever 0 is not contained in the disk $(c; r_c)$ containing the range of $1 - x^2$. As we approach the singularity at $x = 1$, the width h of successful steps of Algorithm 3.4.1 becomes smaller and smaller. We never reach time $t = 1/2$, let alone time $t = 1$.

In order to obtain analytic behaviour of y, we let x circumvent the singularity at $x = 1$ on a detour through the complex plane from 0 to 2 and back to 0. To that end, we use the 'gamma trick' of (Morgan and Sommese, 1987): In the linear homotopy $x(t) = 2t$, we replace t by $t\mathrm{e}^{\mathrm{i}\theta(1-t)}$. We obtain the homotopy

$$x_1(t) = 2t\mathrm{e}^{\mathrm{i}\theta(1-t)},$$

which describes a complex detour from 0 to 2 for any angle θ. We choose $\theta = -\pi/2$. For the way back from $x = 2$ to $x = 0$, we analogously choose the homotopy

$$x_2(t) = 2(1 - t\mathrm{e}^{\mathrm{i}\theta(1-t)}).$$

Note that $x_1(t)$ and $x_2(t)$ are distance non-decreasing: For any $0 \le t_0 \le t_1 \le 1$, the disk $(x_1(t_0); |x_1(t_1) - x_1(t_0)|)$ contains all points of $x_1(t)$ with $t_0 \le t \le t_1$. Hence, in Algorithm 3.4.1, we can use such non-linear homotopies for the input variable(s) as if we used linear ones.

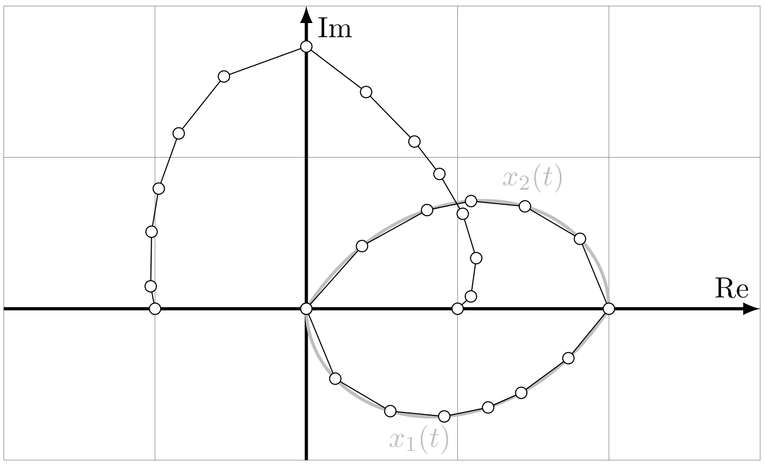


Figure 3.5.2: Homotopies $x_1(t)$ and $x_2(t)$, their approximations by successful steps of Algorithm 3.4.1, and approximate corresponding motion of y. All end points of successful steps (white points) lie on the exact paths of x and y, up to floating point errors.

When x moves from 0 to 2 along $x_1(t)$, y moves from 1 to $\sqrt{3}\mathrm{i}$ (see Figure 3.5.2); when x moves from 2 back to 0 along $x_2(t)$, y moves from $\sqrt{3}\mathrm{i}$ to -1. Effectively, x returns to its original position and y moves from one branch of the square root to the other.

3.6 Comparison with Denner-Broser's approach

The main differences between Denner-Broser's approach and our approach are the following:

Instead of geometric straight-line programs, we use special triangular systems of polynomials to model computations. Instead of the Radicand Lemma (Denner-Broser, 2008, Lemma 6.1.1, Lemma 6.6.1), we use Lemma 3.3.12 (cf. (Petković and Petković, 1984, Theorem 1)) to ensure separation of branches of n-th root operations.

We separate certification of a feasible step width and computation of the result of a step. In order to determine the feasibility of a candidate step, it is not necessary to compute the actual result of the step; for infeasible steps, we thus save some computational effort.

Denner-Broser manages to generalize her algorithm so that it finds a feasible step width for a computation with at most one infeasible guess per root or division operation (Denner-Broser, 2008, Algorithm 4). To that end, she uses a derivative of geometric straight-line programs and the Cone Lemma (Denner-Broser, 2008, Lemma 6.3.1), 'a reformulation of the mean value theorem' (Denner-Broser, 2008, p. 79). This is a very nice result from a theoretical viewpoint. Denner-Broser uses it to bound the time complexity of her algorithm w.r.t. the number of division and root operations of a computation (Denner-Broser, 2008, Theorem 6.3.4). However, it is uncertain whether in practice this approach yields a significant performance improvement compared to bisection, despite the additional computational effort needed for the computation of the derivative of the geometric straight-line program and for the computation of the feasible step width using the Cone Lemma.

Denner-Broser analyzes her algorithms in great detail, for example with respect to numerical robustness, overestimation, and other choices of complex interval arithmetic—beyond what we can cover here.

round	1	2	3	4	5	6
t_0	0	0	0	1/4	1/4	1/4
h	1	1/2	1/4	3/4	3/8	3/16
t_1	1	1/2	1/4	1	5/8	7/16
$x(t_0)$	0	0	0	1/2	1/2	1/2
$x(t_1)$	2	1	1/2	2	5/4	7/8
$y(t_0)$	1	1	1	$\sqrt{3}/2$	$\sqrt{3}/2$	$\sqrt{3}/2$
$(x; r_x)$	$(0; 2)$	$(0; 1)$	$(0; 1/2)$	$(1/2; 3/2)$	$(1/2; 3/4)$	$(1/2; 3/8)$
$(a; r_a)$	$(1; 0)$	$(1; 0)$	$(1; 0)$	$(1; 0)$	$(1; 0)$	$(1; 0)$
$(b; r_b)$	$(0; 4)$	$(0; 1)$	$(0; 1/4)$	$(1/4; 15/4)$	$(1/4; 21/16)$	$(1/4; 33/64)$
$(c; r_c)$	$(1; 4)$	$(1; 1)$	$(1; 1/4)$	$(3/4; 15/4)$	$(3/4; 21/16)$	$(3/4; 33/64)$
$(y; r_y)$	$0 \in (1; 4)$!	$0 \in (1; 1)$!	$(1; 1 - \sqrt{3}/2)$	$0 \in (3/4; 15/4)$!	$0 \in (3/4; 21/16)$!	$(\sqrt{3}/2; \sqrt{3}/2 - \sqrt{15}/8)$
x	-	-	1/2	-	-	7/8
a	-	-	1	-	-	1
b	-	-	1/4	-	-	49/64
c	-	-	3/4	-	-	11/64
y	-	-	$\sqrt{3}/2$	-	-	$\sqrt{11}/16$

Table 3.5.1: First iterations of the computation of $y = \sqrt{1 - x^2}$ by Algorithm 3.4.1 for $x = x(t) = 2t$, $0 \leq t \leq 1$

4 Generation of Real Algebraic Loci via Complex Detours

4.1 Introduction

A locus is a set of points in the plane with a common geometric property. For example, a circle is the set of points whose distance from the centre of the circle equals the radius of the circle. Here we focus on loci that are closed curves generated by dynamic geometric constructions.

The generation of loci has a very long history. We roughly follow the exposition of (Brieskorn and Knörrer, 1986, Chapter 1). Many ancient Greek mathematicians studied geometric constructions. Some classical problems, e.g. squaring the circle, duplication of the cube (Delian problem), and angle trisection, resisted any attempt of solution using only compass and straightedge. Unlike the ancient Greeks, we know that these problems cannot be solved by compass and straightedge constructions.[1]

After many failed attempts using only compass and straightedge, the ancient Greeks sought other means of solving these problems. Menaechmus (c. 350 BC) found that duplication of the cube could be performed using compass, straightedge and conic sections. Other plane algebraic curves classically used as devices for duplication of the cube and angle trisection include the cissoid of Diocles (c. 180 BC) and the conchoid of Nicomedes (c. 180 BC). Diocles and Nicomedes solved duplication of the cube and angle trisection by intersecting these curves with other geometric elements. In order to find the intersections, it is essential that we can construct not only some points but whole arcs of these curves in a continuous manner.

To that end, we can use dynamic compass and straightedge constructions. We move one element of the construction, e.g. we rotate a line about a point or slide a point on a line/circle, and follow a point constructed from the moving element as it traces a curve. If the construction steps can be described algebraically, then the resulting curve is a real connected component of a plane algebraic curve. Such constructions for various curves were known to the ancient Greeks. Newton, who investigated this technique, called it 'organic generation'.

We can use dynamic geometry software to carry out such constructions. Some applications allow us to select a line through a point or a point on a line/circle, which shall be the moving element (mover), and a dependent point, which shall be followed as it traces a curve (tracer). The software then attempts to automatically generate the real connected component of the real plane algebraic curve that is the locus of the tracer under movement of the mover.

Depending on the underlying model of geometry (and depending on the algorithm used), the software may fail to generate the entire real connected component of

[1] Wantzel (1837) showed that using compass and straightedge, we can only construct points with coordinates in a field extension of $\mathbb{Q}$ obtained by adjoining all roots of degree a power of two of rational numbers. He proved that duplication of the cube and angle trisection reduce to solving irreducible cubic equations, which is therefore impossible using only compass and straightedge.

Squaring the circle reduces to constructing π. In the nineteenth century, it was known that π is irrational. Mathematicians wondered whether it could be a solution of an equation of degree a power of two that was potentially solvable using only compass and straightedge by iterative construction of square roots. Lindemann (1882) settled the case, showing that π is transcendental, i.e. not a root of any algebraic equation.

the real algebraic curve. Kortenkamp (1999, Section 6.2, esp. Theorem 6.8) shows that dynamic geometry systems with geometric primitives like ‘intersection of a circle with a line’ or ‘angle bisector of two lines’ cannot be both determined and continuous: If we require that we can determine a unique instance of a construction for every possible position of its free elements (movable elements), then we cannot expect that its dependent elements always move continuously.

The reason behind this is that ‘intersection of a circle and a line’ or ‘angle bisector of two lines’ are ambiguous geometric operations. In general, a circle and a line have two (possibly complex) intersections. In general, two lines have two angle bisectors, which are perpendicular to each other. Consider an angle bisector of two (unoriented) lines a, b through a common point P. If we rotate a about P and the angle bisector moves continuously, then the angle bisector rotates at half the angular velocity. When a has rotated by an angle of π, it has reached its initial position. At the same time, the angle bisector has rotated by an angle of $\pi/2$ and has become perpendicular to its initial position. We have moved continuously from one possible instance of the construction to the other. Therefore the dynamic geometric system cannot be determined.

For that reason, we may sometimes observe dependent elements jump unexpectedly in most dynamic geometry software. If a tracer jumps during locus generation, the algorithm may miss part of its real algebraic locus.

As we have seen, if we want to avoid jumping elements, we need to take all possible output elements of ambiguous geometric operations into account. We parameterize the motion of the mover using a time parameter t. There may be some points in time when part of the construction degenerates or when two possible choices of a dependent element coincide and become indistinguishable. We call such points in time singularities. For example, the two intersections of a circle and a line coincide when we move the centre of the circle such that it merely touches the line. If we can somehow avoid singularities, then we can always distinguish all possible choices. Hence, we may be able to determine the instance that produces a continuous evolution of the construction.

In order to avoid jumping elements in their dynamic geometry software Cinderella (Kortenkamp and Richter-Gebert, 2006), Kortenkamp and Richter-Gebert introduced the paradigm of ‘complex detours’ (Kortenkamp, 1999, esp. Chapter 7; Kortenkamp and Richter-Gebert, 2001b; 2002): We do not let the time parameter run along the real time axis where we often encounter singularities. Instead, we embed the real time axis as the real axis of the complex plane. Between two points on the real time axis, we let the time parameter take a detour through the complex plane to circumvent singularities. Thus we can avoid singularities with high probability.

The locus generation algorithm of Cinderella exploits this principle; it has been working very well in practice for many years, but from a theoretical perspective, the algorithm has not been examined in more detail yet (Richter-Gebert, 2014). In what follows, we analyze the algorithm and some assumptions on which it is based.

Moreover, we introduce a variant of the algorithm that might always generate an entire real connected component of a real algebraic locus.

4.2 A locus generation algorithm

The locus generation algorithm addresses the following problem: Consider a geometric construction. Choose an element of the construction whose movement is constrained to one dimension, e.g. a line through a point, a point on a line, or a point on a circle. We call this element ‘mover’. Choose a point of the construction whose position depends on the position of the mover. We call this point ‘tracer’. Suppose the tracer can be constructed from the mover (and possibly further elements) using only geometric operations that have an algebraic representation. Then movement of the mover causes the tracer to move on a real plane algebraic curve. The goal of the locus generation algorithm is to automatically produce the locus of the tracer under movement of the mover.

The following locus generation algorithm has two variants. Variant A is essentially equivalent to the locus generation algorithm implemented in Cinderella. Variant B possibly generates an entire real connected component of a real algebraic locus (see Conjecture 4.5.1).

Algorithm 4.2.1 (locus generation algorithm)**.** We rationally parameterize the motion of the mover using a time parameter t. We assume that we start at a non-singular initial time $t_0 \in \mathbb{R}$ when the position of the tracer is real-valued.

1. Let $s := 1$, $t := t_0$.

2. Let $t' := t + s \cdot \varepsilon$ for some small step size $\varepsilon > 0$.

Consider the circle c in the complex plane that has the segment between t and t' as its diameter. We let the time parameter take a complex detour on this circle.

3. Let t take a small step on circle c in anticlockwise direction.

4. We trace the geometric construction as the mover moves according to its parameterization along the complex detour up to the new position of t.

5. **Variant A** If t is real-valued and the tracer has real-valued coordinates, the tracer has reached a point of the real plane algebraic locus. It may happen that t ends up in its initial position on circle c. In this case, we invert the direction of movement of the mover. To that end, we set $s := -s$. (Note that this affects only the choice of sampling points of time parameter t; the anticlockwise orientation of complex detours remains the same.) If $t = t_0$ and the tracer has reached its initial position again, we stop. Otherwise, we go to step 2.

   If t is not real-valued or the tracer does not have real-valued coordinates, we go to step 3.

Variant B If the tracer has real-valued coordinates, it has reached a point of the real algebraic locus. It may happen that t is not real-valued or that it ends up in its initial position on circle c. In any case, we update the direction of movement of the mover. To that end, we set

$$s = \frac{t-a}{|t-a|},$$

where a denotes the centre of circle c. If $t = t_0$ and the tracer has reached its initial position again, we stop. Otherwise, we go to step 2.

If the tracer does not have real-valued coordinates, we go to step 3.

Remark 4.2.2. Algorithm 4.2.1 evaluates the geometric construction only at discrete points in time along a complex detour. At every sampling point, the algorithm should select the right instance of the construction, i.e. the instance that yields a continuous evolution of the construction. More precisely, we want that the coordinates of the elements of the construction are locally analytic functions of time; the algorithm should select the instance that results from analytic continuation of the coordinate functions along the complex detours. To that end, we can use Algorithm 3.4.1 to trace the geometric construction along the complex detour.

Besides the assumption that we start at a non-singular initial time when the position of the tracer is real-valued, Algorithm 4.2.1 is based on the following assumptions. (This list of assumptions may not be complete; Algorithm 4.2.1 may be based on further implicit assumptions.)

Assumption 4.2.3. We assume that we always choose ε small enough so that the complex detours wind around at most one ramification point of a coordinate of the tracer, and only around ramification points at which the position of the tracer is real-valued. Thus we preclude that the tracer jumps from one real arc of the real algebraic locus to another due to a complex detour around a ramification point of a coordinate at which the tracer has a complex position close to the real algebraic locus.

Assumption 4.2.4. We assume that the steps we take on the complex detours are small enough so that we do not miss a real position of the tracer.

Remark 4.2.5. While we usually satisfy Assumption 4.2.3 in practice, it is not clear how to guarantee that it is satisfied in general.

Remark 4.2.6. For Variant A of Algorithm 4.2.1, we can satisfy Assumption 4.2.4, if we ensure that time parameter t always attains the real values on the complex detours. For Variant B of Algorithm 4.2.1, we must additionally determine complex points in time on the complex detours when the coordinates of the tracer are real-valued. We can try to approximate these points in time by bisection if the imaginary part of a coordinate of the tracer changes sign or has small absolute value; this may be really difficult in practice.

Remark 4.2.7. Since a real algebraic locus can contain points at infinity, it may sometimes be advantageous to describe the plane algebraic curve containing the real algebraic locus by a homogeneous equation $f(x, y, x) = 0$. Thus we can express points of the locus at infinity using finite coordinates $(x, y, z)^\top$. We can homogenize an affine equation $f(x, y) = 0$ of total degree n by plugging in $x = x/z$, $y = y/z$, and multiplying by z^n. If we set $z = 1$, we return to the affine equation.

Remark 4.2.8. If we use homogeneous coordinates $(x, y, z)^\top$ to describe the position of the tracer, we require that none of x, y, and z become infinite; if necessary, we constantly normalize $(x, y, z)^\top$ to avoid that one of the coordinates grows too much.

4.3 Orientation of complex detours

In Step 3 of Algorithm 4.2.1, we specify that time parameter t takes complex detours along circles in the complex plane in anticlockwise direction. In principle, t can also take complex detours in clockwise direction. Are there constructions where the generated locus changes depending on the clockwise or anticlockwise orientation of the complex detours?

In order to answer this question, we make some assumptions on the real algebraic loci to which we apply our locus generation algorithm, without loss of generality. Let

$$\mathcal{C}\colon\ f(x, y) = 0$$

denote a plane algebraic curve containing such a locus.

Assumption 4.3.1. We assume that $f(x, y)$ is irreducible. An analytic transition from one irreducible component of $f(x, y)$ to another is impossible. Therefore, if $f(x, y)$ is reducible, we can without loss of generality consider the irreducible component containing the starting point. (Note that the assumption that we start Algorithm 4.2.1 at a non-singular initial time precludes that we start at an intersection of irreducible components.)

Assumption 4.3.2. We assume that $\mathcal{C}$ has a real connected component. Otherwise, the locus is not a real connected component of a real plane algebraic curve and the locus generation algorithm is not applicable.

Lemma 4.3.3. *Under Assumption 4.3.1 and Assumption 4.3.2, without loss of generality, $f(x, y)$ has only real coefficients.*

Proof. Conversely, suppose $f(x, y)$ has complex coefficients. Then we can write $f(x, y)$ in the form $f(x, y) = f_\Re(x, y) + \mathrm{i} f_\Im(x, y)$ such that the real part polynomial $f_\Re(x, y)$ and the imaginary part polynomial $f_\Im(x, y)$ possess only real coefficients. By the assumption that $f(x, y)$ has complex coefficients, $f_\Im(x, y)$ does not vanish identically. If $f_\Re(x, y)$ vanishes identically, then $f(x, y) = \mathrm{i} f_\Im(x, y)$ and we can cancel the unit i from the equation $f(x, y) = 0$ to obtain an equation for $\mathcal{C}$ with only real coefficients. Hence, suppose that both real part polynomial and imaginary part polynomial do not vanish identically. By Assumption 4.3.2, $\mathcal{C}$ has a real connected component, i.e. over a real interval of x-values, $f(x, y)$ vanishes for real y-values. This can only be the case if $f_{\Re}(x, y)$ and $f_{\Im}(x, y)$ vanish there, i.e. if $f_{\Re}(x, y)$ and $f_{\Im}(x, y)$ have infinitely many common zeros. If $f_{\Re}(x, y)$ and $f_{\Im}(x, y)$ have infinitely many common zeros, then they must have a common component or $f_{\Im}(x, y)$ must be a non-zero real multiple of $f_{\Re}(x, y)$. By Assumption 4.3.1, $f_{\Re}(x, y)$ and $f_{\Im}(x, y)$ are irreducible; they cannot have a common component. Therefore, $f_{\Im}(x, y)$ must be a non-zero real multiple of $f_{\Im}(x, y)$. Hence, we can write $f_{\Im}(x, y) = \lambda \cdot f_{\Re}(x, y)$ for some $\lambda \in \mathbb{R}$. Therefore, $f(x, y) = (1 + \mathrm{i}\lambda) f_{\Re}(x, y)$, i.e. $f(x, y)$ has only real coefficients up to multiplication by a unit in $\mathbb{C}$, which we can cancel from the equation $f(x, y) = 0$. □

Lemma 4.3.4. *Consider a plane algebraic curve $\mathcal{K}$: $p(x, y) = 0$, where $p(x, y)$ is a polynomial with only real-valued coefficients. The paths of the y-values on the complex plane algebraic curve $\mathcal{K}$ under complex conjugate movement of x are complex conjugate.*

Proof. $\mathcal{K}$ is the zero set of a polynomial $p(x, y)$ with only real coefficients. Real coefficients are invariant under complex conjugation. Hence, if we consider the complex conjugate of the defining equation of $\mathcal{K}$, we find that

$$0 = \overline{0} = \overline{p(x, y)} = p(\overline{x}, \overline{y}),$$

and $p(\overline{x}, \overline{y})$ vanishes if and only if $p(x, y)$ vanishes.

Let $x(t)$ be a parameterization of the movement of x through the complex plane. Let $y(t)$ be a parameterization of the corresponding analytic motion of a y-value so that $p(x(t), y(t)) = 0$, for all t. Then to complex conjugate movement of x, parameterized by $\overline{x(t)}$, corresponds complex conjugate motion of the y-value, parameterized by $\overline{y(t)}$, since

$$p(\overline{x(t)}, \overline{y(t)}) = 0 \Leftrightarrow p(x(t), y(t)) = 0.$$

(The y-values must be the same in both cases since they must agree at all real points of the complex plane algebraic curve $\mathcal{K}$, where complex conjugation has no effect.) □

Remark 4.3.5. Suppose we can construct the tracer from the mover (and possibly further elements) using only geometric operations that have an algebraic representation. Moreover, suppose that the motion of the mover is rationally parameterized in time parameter t. Then the x-coordinate and the y-coordinate of the tracer satisfy algebraic equations $g(t, x) = 0$ and $h(t, y) = 0$. In practice, it may often be too expensive to work these equations out symbolically. But in principle, we can determine an equation $f(x, y) = 0$ for the locus of the tracer by taking the t-resultant of $g(t, x)$ and $h(t, y)$ to eliminate t from these equations.

Theorem 4.3.6. *The locus generation algorithm is independent of the clockwise or anticlockwise orientation of its complex detours.*

Proof. By Remark 4.3.5, the x-coordinate of the tracer is determined by an algebraic equation $g(t, x) = 0$. Analogously to Assumption 4.3.1, Assumption 4.3.2, and Lemma 4.3.3, we may assume that $g(t, x)$ has only real coefficients, without loss of generality. If we reverse the orientation of the complex detours, we obtain complex conjugate movement of time parameter t. By Lemma 4.3.4, the motion of x such that $g(t, x) = 0$ under complex conjugate movement of t is complex conjugate. Particularly, the real values of x resulting from either movement agree. The same argument applies to the algebraic equation $h(t, y) = 0$ that governs the motion of y under movement of t. Consequently, Algorithm 4.2.1 produces the same real algebraic locus, independent of the clockwise or anticlockwise orientation of its complex detours. □

4.4 Termination

Does the locus generation algorithm always terminate? Or can we get lost on algebraic Riemann surfaces?
Analogously to the previous section, we assume that $f(x, y)$ is an irreducible polynomial (cf. Assumption 4.3.1) with real coefficients (cf. Assumption 4.3.2 and Lemma 4.3.3).

Besides, recall the assumptions on which the locus generation algorithm (Algorithm 4.2.1) is based: Firstly, we assume that the complex detours of Algorithm 4.2.1 are small enough so that they wind around at most one ramification point of a coordinate of the tracer, and only around ramification points at which the position of the tracer is real-valued (Assumption 4.2.3). Thus we preclude that the tracer jumps from one real arc of the real algebraic locus to another due to a complex detour around a ramification point of a coordinate at which the tracer has a complex position close to the real algebraic locus. Secondly, we assume that the steps we take on the complex detours are small enough so that we do not miss a real position of the tracer (Assumption 4.2.4).

A proof of termination of Algorithm 4.2.1 has remained elusive so far. However, the successful application of Variant A of Algorithm 4.2.1 in Cinderella supports the following conjecture:

Conjecture 4.4.1. *The locus generation algorithm terminates if it takes small enough complex detours and small enough steps on every complex detour.*

4.5 Generation of real connected components

Conjecture 4.5.1. *Variant B of Algorithm 4.2.1 generates an entire real connected component of a real algebraic locus if it takes small enough complex detours and small enough steps on every complex detour.*

Remark 4.5.2. Variant A of Algorithm 4.2.1 need not generate an entire real connected component of a real algebraic locus. It may miss real arcs of a locus that correspond to non-real complex values of time parameter t. For an example, see Section 4.6.2.

Remark 4.5.3. A real connected component of a real algebraic locus need not be an algebraic curve by itself. For example, consider a four-bar linkage with bars of lengths 4, 1, 4, and 2. (We discuss how we can express a mechanical linkage in terms of dynamic geometry in Section 4.6.4.) We leave the first bar of the linkage fixed and trace the midpoint of the third bar under continuous movement of the linkage. Then the four-bar linkage generates one of the two real connected components of the plane algebraic sextic

$$\begin{aligned}\mathcal{C}\colon f(x,y) &= (6+5x-2x^3)^2 + 3(-45+4x(-2+2x+x^3))y^2 \\ &\quad + 4(11+3x^2)y^4 + 4y^6 = 0.\end{aligned}$$

The algebraic curve $\mathcal{C}$ is irreducible (over the complex numbers). Thus, each of the two real connected components by itself cannot be an algebraic curve.

4.6 Examples

4.6.1 Conic through five points in general position

Consider Pascal's theorem (also known as 'hexagrammum mysticum'):

Theorem 4.6.1 (Pascal's theorem)**.** *If A, B, C, D, E, F are six points on a conic, then opposite sides of the hexagon $ABCDEF$ (extended to lines, if necessary) meet in three collinear points.*

For more details and proofs, see (Richter-Gebert, 2011, esp. Section 1.4 and Section 10.6).

The converse of the theorem is also true, which gives rise to organic generation of a conic through five points, by the following construction (see Figure 4.6.2).

Construction 4.6.3. Let A, B, C, D, E be five points of the real projective plane, in general position.

1. Let a be the line through A and B.
2. Let b be the line through D and E.
3. Let F be the intersection of a and b.
4. Let c be a line through F.
5. Let d be the line through B and C.
6. Let G be the intersection of c and d.
7. Let e be the line through C and D.

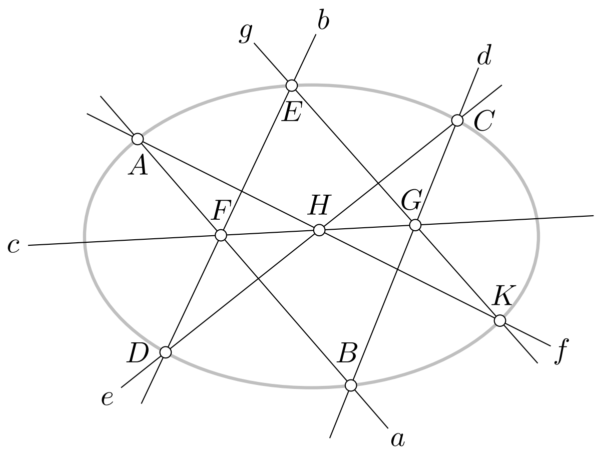


Figure 4.6.2: An instance of Construction 4.6.3. When line c rotates about point F, point K traces the conic through points A, B, C, D, E.

8. Let H be the intersection of c and e.
9. Let f be the line through A and H.
10. Let g be the line through E and G.
11. Let K be the intersection of f and g.

When line c rotates about point F, point K traces the conic through points A, B, C, D, E.

Remark 4.6.4. Construction 4.6.3 does not distinguish the orientation of line c. Point K returns to its initial position after half a turn of line c about point F. If line c makes a full turn, point K traces the conic through points A, B, C, D, E twice.

4.6.2 Orthogonal projection of a circle onto a line

We consider the following (seemingly simple) construction (see Figure 4.6.5), because it highlights the difference between Variant A and Variant B of Algorithm 4.2.1.

Construction 4.6.6. Let a line b, a circle c_0, and a point A on circle c_0 be given.

1. Let a be the line through A perpendicular to b.
2. Let B be the intersection of a and b.

When point A moves around on circle c_0, point B traces a segment of line b.

Remark 4.6.7. For simplicity, we use the geometric primitive ‘perpendicular to a line through a point‘. It can be easily constructed with compass and straightedge (see Book I, Proposition 12 of Euclid’s Elements).

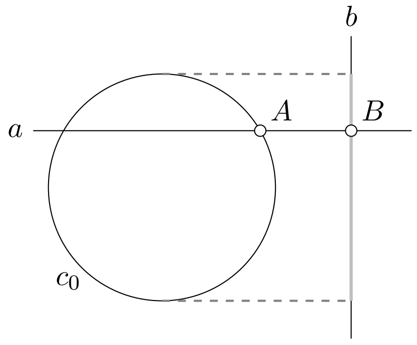


Figure 4.6.5: An instance of Construction 4.6.6. When point A moves around on circle c_0, point B traces a segment of line b.

Remark 4.6.8. Line a intersects circle c_0 in two points (counted with multiplicity). Hence there are two positions of point A (counted with multiplicity) for every position of point B on the segment that point B traces on line b. In other words, point B covers the segment twice as point A makes a full turn on circle c_0.

Let us work out the real algebraic locus of point B under movement of point A on circle c_0 algebraically. Without loss of generality, let c_0 be the unit circle and b the line parallel to the y-axis intersecting the x-axis at $x = 2$. Let O denote the origin. We need to parameterize the motion of point A on circle c_0. To that end, we can use trigonometric functions, as follows:

$$A = (\cos\varphi, \sin\varphi)^\top, \quad -\pi \leq \varphi \leq \pi.$$

However, this parameterization is not rational. We use tangent half-angle substitution,

$$t = \tan\frac{\varphi}{2}, \quad \cos\varphi = \frac{1-t^2}{1+t^2}, \quad \sin\varphi = \frac{2t}{1+t^2},$$

and homogeneous coordinates to derive the rational parameterization

$$A = (1-t^2, 2t, 1+t^2)^\top, \quad t \in \mathbb{R}.$$

Line b has homogeneous coordinates

$$b = (1, 0, -2)^\top.$$

Line a is perpendicular to line b and therefore has homogeneous coordinates of the form

$$a = (0, 1, z(t))^\top,$$

where $z(t)$ has to be determined so that point A lies on line a, i.e. so that

$$\langle a, A\rangle = 2t + z(t)(1+t^2) = 0.$$

Hence, homogeneous coordinates of line a are

$$a = (0, 1 + t^2, -2t)^\top.$$

We obtain homogeneous coordinates of point B by taking the cross product of line a and line b,

$$B = a \times b = (2(1 + t^2), 2t, 1 + t^2)^\top \sim \left(2, \frac{2t}{1 + t^2}, 1\right)^\top.$$

The real algebraic locus of point B under movement of point A on circle c_0 is described by the implicit equations

$$\begin{gathered} x - 2 = 0 \\ t^2 y - 2t + y = 0. \end{gathered}$$

If we take the t-resultant of these equations, we arrive at a single equation for the real algebraic locus,

$$(x - 2)^2 = 0.$$

The construction has singularities at $t = \pm 1$, where the y-coordinate of point B equals ± 1. If we solve the equation between t and the y-coordinate of point B,

$$t^2 y - 2t + y = 0,$$

for t, we find that

$$t = \frac{1 \pm \sqrt{1 - y^2}}{y}.$$

For real y-values of absolute values greater than 1, t becomes complex. Point B moves higher or lower than the (real-valued) extreme positions of point A on circle c_0 if and only if we allow point A to become complex.

Variant A of Algorithm 4.2.1 does not allow this to happen and skips the branch of the real algebraic locus where point B has real-valued coordinates, but point A (and thus t) is complex-valued. It generates that part of the real algebraic locus where both mover and tracer have real-valued coordinates.

Variant B of Algorithm 4.2.1 does not skip the branch of the real algebraic locus where point B has real-valued coordinates, but point A (and thus t) is complex-valued. It generates the entire real algebraic locus.

Variant A seems more appropriate from the perspective of real projective geometry; Variant B seems more appropriate from the algebraic perspective.

4.6.3 Conchoid of Nicomedes

The conchoids of Nicomedes are a family of quartic plane algebraic curves

$$\mathcal{C}\colon f(x, y) = (y + a)^2(x^2 + y^2) - b^2 y^2 = 0, \quad a, b > 0.$$

For organic generation of a conchoid, we can use the following construction.

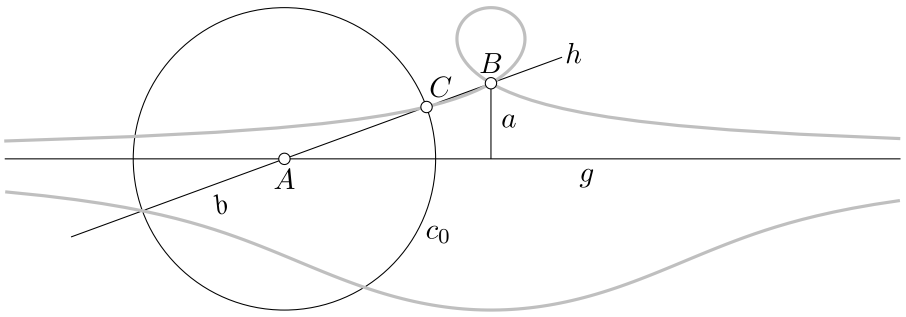


Figure 4.6.9: An instance of Construction 4.6.10. When point A moves along line g, point C traces the conchoid with *pole* B, *base* g and *distance* b.

Construction 4.6.10 (conchoid of Nicomedes)**.** Let A be a point on a line g. Let B be a point at distance $a > 0$ from g. Let c_0 be a circle of radius b centred at A.

1. Let h be the line through A and B.

2. Let C be an intersection of c_0 and h.

When point A moves along line g, point C traces the conchoid with *pole* B, *base* g, and *distance* b.

As mentioned in the introduction, part of the original motivation to study the conchoid of Nicomedes was that it can be used for angle trisection. The following construction is based on (Ferréol and Mandonnet, 2005, Conchoïde de Nicomède, `http://www.mathcurve.com/courbes2d/conchoiddenicomede/conchoiddenicomede.shtml`).

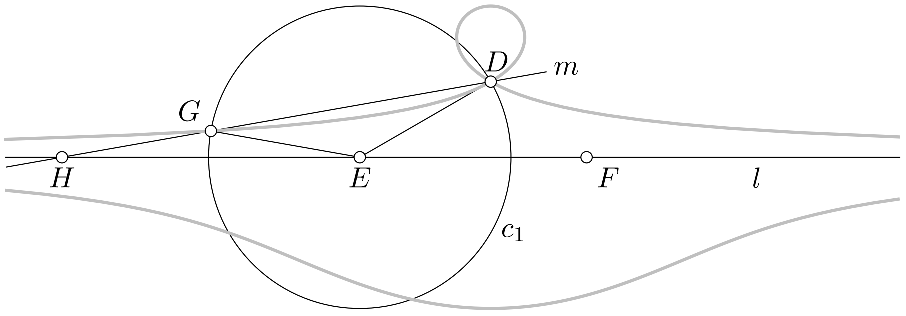


Figure 4.6.11: An instance of Construction 4.6.12. By construction, angle $\angle HEG$ trisects angle $\angle DEF$.

Construction 4.6.12 (angle trisection)**.** Let $\angle DEF$ be the angle to be trisected. We can trisect a right angle by constructing an equilateral triangle. Hence, without loss of generality, let angle $\angle DEF$ be acute.

1. Let l be the line through E and F.

2. Use Construction 4.6.10 to generate the conchoid with pole D, base l, and distance $|DE|$.
3. Let c_1 be the circle centred at E, through F.
4. Let G be the intersection of c_1 with the conchoid that lies on the same side of l as F.
5. Let m be the line through D and G.
6. Let H be the intersection of l and m.

Then angle $\angle HEG$ trisects angle $\angle DEF$.

Proof. By construction of the conchoid, points H and E are equidistant to point G. Therefore, triangle $\triangle EGH$ is equilateral and

$$\angle HEG = \angle GHE.$$

We apply the exterior angle theorem to triangle $\triangle EGH$ and find that

$$\angle DGE = \angle GHE + \angle HEG = 2 \cdot \angle HEG.$$

Triangle $\triangle GED$ is equilateral by construction, and thus

$$\angle EDG = \angle DGE = 2 \cdot \angle HEG.$$

We apply the exterior angle theorem to triangle $\triangle HED$ and conclude

$$\begin{aligned}\angle DEF &= \angle GHE + \angle EDH = \angle GHE + \angle EDG = \angle HEG + 2 \cdot \angle HEG \\ &= 3 \cdot \angle HEG.\end{aligned}$$ □

4.6.4 Watt curves

We can use Algorithm 4.2.1 to simulate mechanical linkages. For example, the following construction uses a four-bar linkage to generate a Watt curve, a plane algebraic sextic

$$\mathcal{C}\colon f(x,y) = (x^2+y^2)(x^2+y^2-a^2-b^2+c^2)^2 + 4a^2y^2(x^2+y^2-b^2) = 0$$

with parameters $a, b, c > 0$.

Construction 4.6.14. Let A and B be two points in the plane at distance $2a$ from each other. Let c_0 be a circle with centre A and radius b. Let c_1 be a circle with centre B and radius b.

1. Let C be a point on c_0.
2. Let c_2 be a circle with centre C and radius $2c$.

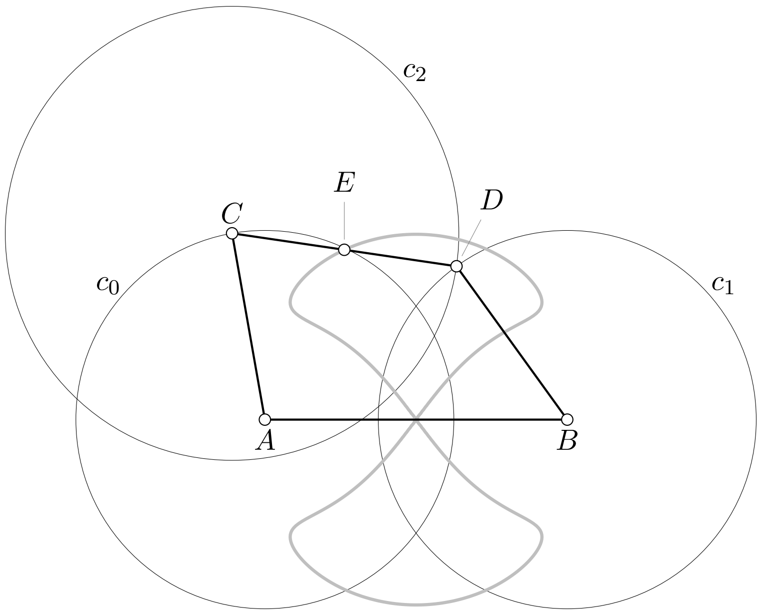


Figure 4.6.13: An instance of Construction 4.6.14. When point C moves on circle c_0 according to Algorithm 4.2.1, point E traces a Watt curve with parameters $a = |AB|/2$, $b = |AC| = |BD|$, and $c = |CD|/2$.

3. Let D be an intersection of c_1 and c_2.
4. Let E be the midpoint of the segment between C and D.

When point C moves on circle c_0 according to Algorithm 4.2.1, point E traces a Watt curve with parameters a, b, c.

Remark 4.6.15. The first bar of the four-bar linkage, segment AB, has length $2a$. Circles c_0, c_1, and c_2 have fixed radii. Hence, they prescribe the lengths $|AC| = b$, $|CD| = 2c$, and $|DB| = b$ of the remaining bars of the four-bar linkage.

Remark 4.6.16. Depending on parameters a, b, c, Watt curves have a wide variety of different shapes.

Remark 4.6.17. In Figure 4.6.13, we choose parameters a, b, c so that $a > c$. Therefore, the movement of point C on circle c_0 is constrained. If point C moved too far to the left, then circles c_1 and c_2 would move apart and would no longer have real-valued intersections. Such movement is not possible without breaking the linkage. Algorithm 4.2.1 resolves the singularities when circles c_1 and c_2 merely touch each other, by taking complex detours around them. At every such singularity, point C reverses its direction of movement on circle c_0. Apparently, Algorithm 4.2.1 produces a physically reasonable motion of the four-bar linkage.

5 Identification of Real Algebraic Loci

5.1 Introduction

We would like to determine the equation of the real plane algebraic curve containing the real algebraic locus generated by our locus generation algorithm (Algorithm 4.2.1). This may help us to draw the curve more efficiently or to work out characteristic features of the curve (see (Lebmeir, 2009)). Knowledge of the equation of the plane algebraic curve is also essential for the visualization technique discussed in the next chapter.

The locus generation algorithm produces a point cloud of points on the real algebraic locus. Lebmeir and Richter-Gebert (2007) use an approximate least squares approach for fitting a real plane algebraic curve to such a point cloud. For that, they need to determine the degree of the plane algebraic curve. To that end, they successively test degrees $1, 2, 3, \ldots$; reaching the right degree correlates with a significant drop of the singular value with minimal absolute value of a certain matrix (Lebmeir and Richter-Gebert, 2007, Section 3).

We follow a slightly different, randomized approach. This allows us to address two of the open questions of (Lebmeir and Richter-Gebert, 2007, Section 6), 'To what extent can randomization techniques be used to speed up the calculations?' and 'Can similar approaches be used within the more special class of rational algebraic functions?'.

5.2 Algorithms

Lemma 5.2.1. *A plane algebraic curve of degree n is uniquely determined by $n(n+3)/2$ points in the plane, in general position.*

Proof. Let $\mathcal{C}\colon f(x,y) = 0$ be a plane algebraic curve of degree n. We plug in $x = X/Z$, $y = Y/Z$ into $f(x,y) = 0$ and multiply by Z^n to obtain a homogeneous equation for $\mathcal{C}$. It has the general form

$$f(X,Y,Z) = \sum_{\substack{0 \le j,k,l \le n \\ j+k+l=n}} a_{jkl} X^j Y^k Z^l = 0.$$

Every summand has total degree n and a degree between 0 and n in each of the variables X, Y, Z. Hence, the number of terms equals the number of partitions of n into three non-negative integers,

$$\binom{n+2}{2} = \frac{(n+2)(n+1)}{2}.$$

Besides, we have one degree of freedom in choosing a non-zero scalar multiple of the defining equation of $\mathcal{C}$. There remain

$$\frac{(n+2)(n+1)}{2} - 1 = \frac{n(n+3)}{2}$$

degrees of freedom. If we prescribe a point of the plane algebraic curve, in general position, the degrees of freedom decrease by one. Consequently, $n(n+3)/2$ points in general position determine the equation of a plane algebraic curve of degree n, up to a non-zero scalar multiple. If we plug in $Z = 1$ into $f(X, Y, Z) = 0$, we return to the affine equation $f(x, y) = 0$. This dehomogenization does not change the number of terms. Therefore, the argument applies independent of whether we describe $\mathcal{C}$ by an affine equation, $f(x, y) = 0$, or by a projective equation, $f(X, Y, Z) = 0$. $\square$

Algorithm 5.2.2 (locus identification algorithm)**.** Let a point cloud generated by Algorithm 4.2.1 be given. Let N denote a maximum degree. Let TOL be a tolerance.

1. For $n = 1, 2, \ldots, N$:
   a) From the point cloud, we randomly choose $m = n(n+3)/2$ distinct points,
   $$(x_k, y_k)^\top, \quad k = 1, 2, \ldots, m.$$
   b) We solve the underdetermined homogeneous linear system of equations
   $$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{0},$$
   where $\boldsymbol{x}$ is the coefficient vector and $\boldsymbol{A}$ is the $m \times (m+1)$ matrix defined as follows: The k-th row of $\boldsymbol{A}$ contains the monomials of a general polynomial in x and y of degree n, in a fixed order, evaluated at $x = x_k$, $y = y_k$.
   c) The coefficient vector yields a candidate for the equation defining the plane algebraic curve containing the point cloud. In order to normalize the candidate equation, we divide its coefficients by the coefficient with maximum absolute value. We thus obtain a normalized candidate equation $f(x, y) = 0$. If all points $(x, y)^\top$ of the point cloud satisfy $|f(x, y)| < \mathsf{TOL}$, we output the normalized coefficient vector and stop.
2. If we reach this point, then we could not identify an equation of a plane algebraic curve of degree $\leq N$ containing the point cloud up to a tolerance of TOL. We print an error message and stop.

Remark 5.2.3. The idea behind Algorithm 5.2.2 is the following: By Lemma 5.2.1, $n(n+3)/2$ points in the plane, in general position, determine a unique plane algebraic curve of degree n. We randomize the choice of points from the point cloud in order to obtain points in general position with high probability.

The k-th row of the underdetermined homogeneous linear system of equations $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{0}$ represents the condition that point $(x_k, y_k)^\top$ lies on the plane algebraic curve with coefficient vector $\boldsymbol{x}$.

We take rounding errors introduced by floating point arithmetic into account and use a tolerance of TOL to test whether all points of the point cloud lie on the candidate plane algebraic curve.

We stop once all points of the point cloud lie on the plane algebraic curve up to a tolerance of TOL. Thus we try to guard against degree overestimation. If we overestimated the degree of the plane algebraic curves, we would obtain additional reducible components as artefacts (see (Lebmeir and Richter-Gebert, 2007, Figure 3)).

Remark 5.2.4. We may use the following approach to solve the underdetermined homogeneous linear system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{0}$: We remove a column $\boldsymbol{b}$ of matrix $\boldsymbol{A}$ so that we obtain the $m \times m$ matrix $\boldsymbol{A}'$ with minimal condition number among all matrices obtained by removing one column from $\boldsymbol{A}$. Then we solve the linear system $\boldsymbol{A}'\boldsymbol{x}' = -\boldsymbol{b}$ and set the coefficient corresponding to column $\boldsymbol{b}$ to 1.

If we choose points in general position, the coefficient vector $\boldsymbol{x}$ is uniquely determined up to a non-zero scalar factor. Hence, we may choose one non-zero coefficient to be 1. Equivalently, we may remove one column $\boldsymbol{b}$ from $\boldsymbol{A}$ to obtain the linear system $\boldsymbol{A}'\boldsymbol{x}' = -\boldsymbol{b}$. In practice, matrix $\boldsymbol{A}'$ is often nearly singular. Therefore, we remove the column $\boldsymbol{b}$ of $\boldsymbol{A}$ that yields the best-conditioned matrix $\boldsymbol{A}'$.

Remark 5.2.5. We can increase the numerical stability of the algorithm if we use singular value decomposition to solve the underdetermined homogeneous linear system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{0}$. Singular value decomposition is probably also more efficient than the above approach for solving the linear system.

Remark 5.2.6. We can increase the reliability of Algorithm 5.2.2 if we repeat steps a) to c) a couple of times for every degree n.

Remark 5.2.7. We can easily adapt Algorithm 5.2.2 to homogeneous coordinates.

Remark 5.2.8. Does the randomization of Algorithm 5.2.2 increase performance compared to the approach of (Lebmeir and Richter-Gebert, 2007)?

Suppose we use singular value decomposition to solve the underdetermined homogeneous linear system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{0}$ in step b) of Algorithm 5.2.2. The algorithm of (Lebmeir and Richter-Gebert, 2007) performs singular value decomposition on a matrix defined analogously to matrix $\boldsymbol{A}$, but with a row for every point of the point cloud. The point cloud generated by Algorithm 4.2.1 usually consists of several hundreds of points. Hence, the algorithm of (Lebmeir and Richter-Gebert, 2007) performs singular value decomposition on a usually much larger matrix.

Algorithm 5.2.2 needs to test whether all points of the point cloud lie on the candidate plane algebraic curve up to a tolerance of TOL. The algorithm of (Lebmeir and Richter-Gebert, 2007) simply measures the drop of the singular value of minimal absolute value.

For large point clouds, the decrease in complexity of the singular value decomposition likely surpasses the additional cost of randomization and the approximate incidence test.

Remark 5.2.9. Analogously to Algorithm 5.2.2, we can determine the coefficients of plane algebraic equations $g(t, x) = 0$ and $h(t, y) = 0$ that describe how the x-coordinate and the y-coordinate depend on time parameter t.

If the plane algebraic curve is rational, equations $g(t, x) = 0$ and $h(t, y) = 0$ are linear in x respectively y (or powers of such equations). We can solve for x respectively y to obtain a rational parameterization of the plane algebraic curve (see Section 5.3.2).

We obtain the following algorithm:

Algorithm 5.2.10 (rational parameterization)**.** Let a point cloud generated by Algorithm 4.2.1 be given. Let the motion of the mover be rationally parameterized in time parameter t. Together with every point $(x, y)^\top$ of the point cloud, we store the corresponding parameter value t. Let N denote a maximum degree. Let TOL be a tolerance.

1. For $n = 1, 2, \ldots, N$:
   a) From the point cloud, we randomly choose $m = 2(n + 1) - 1$ distinct points,
   $$(t_k, x_k, y_k)^\top, \quad k = 1, 2, \ldots, m.$$
   b) We solve the underdetermined homogeneous linear system of equations
   $$\boldsymbol{A}_g \boldsymbol{x}_g = \boldsymbol{0},$$
   where $\boldsymbol{x}_g$ is the coefficient vector and $\boldsymbol{A}_g$ is the $m \times (m + 1)$ matrix defined as follows: The k-th row of $\boldsymbol{A}_g$ contains the monomials of a general polynomial in t and x of total degree $n + 1$ that has degree n in t and is linear in x, in a fixed order, evaluated at $t = t_k$, $x = x_k$.
   c) The coefficient vector yields a candidate for equation $g(t, x) = 0$ describing the relation between time parameter t and the x-coordinate of the plane algebraic curve containing the point cloud. In order to normalize the candidate equation, we divide its coefficients by the coefficient with maximum absolute value. We thus obtain a normalized candidate equation $g(t, x) = 0$. Unless all points $(t, x, y)^\top$ of the point cloud satisfy $|g(t, x)| < \mathsf{TOL}$, we increment n and go to step 1.
   d) We solve the underdetermined homogeneous linear system of equations
   $$\boldsymbol{A}_h \boldsymbol{x}_h = \boldsymbol{0},$$
   where $\boldsymbol{x}_h$ is the coefficient vector and $\boldsymbol{A}_h$ is the $m \times (m + 1)$ matrix defined as follows: The k-th row of $\boldsymbol{A}_h$ contains the monomials of a general polynomial in t and y of total degree $n + 1$ that has degree n in t and is linear in y, in a fixed order, evaluated at $t = t_k$, $y = y_k$.
   e) The coefficient vector yields a candidate for equation $h(t, y) = 0$ describing the relation between time parameter t and the y-coordinate of the plane algebraic curve containing the point cloud. In order to normalize the candidate equation, we divide its coefficients by the coefficient

with maximum absolute value. We thus obtain a normalized candidate equation $h(t, y) = 0$. Unless all points $(t, x, y)^\top$ of the point cloud satisfy $|h(t, y)| < \mathsf{TOL}$, we increment n and go to step 1.

f) We solve equation $g(t, x) = 0$, which is linear in x, for x. Thus we obtain an approximate rational parameterization $x(t)$ for the x-coordinate of the plane algebraic curve containing the point cloud.

g) We solve equation $h(t, y) = 0$, which is linear in y, for y. Thus we obtain an approximate rational parameterization $y(t)$ for the x-coordinate of the plane algebraic curve containing the point cloud.

h) We output $x(t), y(t)$ and stop.

2. If we reach this point, then we could not identify an approximate rational parameterization $x(t), y(t)$ of the plane algebraic curve containing the point cloud up to a tolerance of TOL. We print an error message and stop.

Remark 5.2.11. We can easily adapt Algorithm 5.2.10 to homogeneous coordinates.

5.3 Examples

5.3.1 Limaçon (implicit equation)

A limaçon is a quartic plane algebraic curve

$$\mathcal{C}\colon f(x, y) = (x^2 + y^2 - ax)^2 - b^2(x^2 + y^2) = 0$$

with parameters $a, b > 0$.

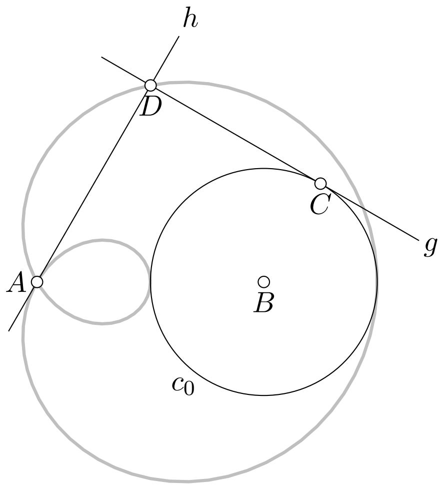


Figure 5.3.1: An instance of Construction 5.3.2

Construction 5.3.2. Let A be a point at the origin. Let B be the point with coordinates $(a,0)^\top$. Let c_0 be a circle with centre B and radius b. Let C be a point on circle c_0.

1. Let g be the tangent to c_0 through C.
2. Let h be the line through A perpendicular to g.
3. Let D be the intersection of g and h.

When point C moves around circle c_0, point D traces a limaçon with parameters a, b.

We apply Algorithm 5.2.2 with $N = 5$ and $\mathsf{TOL} = 10^{-8}$ to a point cloud P of 100 points for Construction 5.3.2 with $a = 2$ and $b = 1$ generated by Algorithm 4.2.1:

$n = 1$**:** We randomly choose $m = 2$ points from point cloud P:

k	x_k	y_k
1	0.9837075567601544	0.10313000277768827
2	0.5542546231506568	−0.36845625340204335

We solve the underdetermined homogeneous linear system of equations

$$\begin{pmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \end{pmatrix} \boldsymbol{x} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

for coefficient vector $\boldsymbol{x}$. We obtain

$$\boldsymbol{x} \approx (1, -0.9107, -0.8898)^\top.$$

Not all points of the point cloud lie on the line

$$\mathcal{C}_1 \colon f_1(x,y) = x - 0.9107y - 0.8898 = 0$$

up to a tolerance of $\mathsf{TOL} = 10^{-8}$ (see Figure 5.3.3a); we have

$$\max_{(x,y)^\top \in P} |f_1(x,y)| \approx 2.741 > \mathsf{TOL}.$$

$n = 2$**:** We randomly choose $m = 5$ points from point cloud P:

k	x_k	y_k
1	1.000684030023719	1.7321822891433003
2	0.4140819900035717	1.4815690462555489
3	0.8390765389044432	−0.29244507461933433
4	0.2008179994763873	0.23252610465687165
5	0.04507584020146185	−1.0822869715963503

We solve the underdetermined homogeneous linear system of equations

$$\begin{pmatrix} x_1^2 & x_1y_1 & y_1^2 & x_1 & y_1 & 1 \\ x_2^2 & x_2y_2 & y_2^2 & x_2 & y_2 & 1 \\ x_3^2 & x_3y_3 & y_3^2 & x_3 & y_3 & 1 \\ x_4^2 & x_4y_4 & y_4^2 & x_4 & y_4 & 1 \\ x_5^2 & x_5y_5 & y_5^2 & x_5 & y_5 & 1 \end{pmatrix} \boldsymbol{x} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

for coefficient vector $\boldsymbol{x}$. We obtain

$$\boldsymbol{x} \approx (-0.9598, 0.2362, -0.02711, 1, -0.1307, -0.1413)^\top.$$

Not all points of the point cloud lie on the quadric

$$\mathcal{C}_2\colon\ f_2(x,y) = -0.9598x^2 + 0.2362xy - 0.02711y^2 + x - 0.1307y - 0.1413 = 0$$

up to a tolerance of $\mathsf{TOL} = 10^{-8}$ (see Figure 5.3.3b); we have

$$\max_{(x,y)^\top \in P} |f_2(x,y)| \approx 5.842 > \mathsf{TOL}.$$

$n = 3$**:** We randomly choose $m = 9$ points from point cloud P:

k	x_k	y_k
1	−0.10351064449890579	−0.6996980293839725
2	0.04507584020146185	−1.0822869715963503
3	0.958509489360396	0.16186019754438224
4	−0.020306267659636154	−0.9573734228502432
5	2.3192631886089474	1.403761324155898
6	1.000684030023719	1.7321822891433003
7	0.7558692942720825	0.33629628143593643
8	0.9894372968119147	−0.0833476962694887
9	0.7085849638314798	−0.35197717330088585

We solve the underdetermined homogeneous linear system of equations

$$\begin{pmatrix} x_1^3 & x_1^2y_1 & x_1y_1^2 & y_1^3 & x_1^2 & x_1y_1 & y_1^2 & x_1 & y_1 & 1 \\ x_2^3 & x_2^2y_2 & x_2y_2^2 & y_2^3 & x_2^2 & x_2y_2 & y_2^2 & x_2 & y_2 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_9^3 & x_9^2y_9 & x_9y_9^2 & y_9^3 & x_9^2 & x_9y_9 & y_9^2 & x_9 & y_9 & 1 \end{pmatrix} \boldsymbol{x} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

for coefficient vector $\boldsymbol{x}$. We obtain

$$\boldsymbol{x} \approx (0.1361, -0.6368, 0.6121, -0.1269, -0.3765, 1, -0.3812, 0.3657, \\ -0.3816, -0.1280)^\top.$$

Not all points of the point cloud lie on the cubic

$$\begin{aligned}\mathcal{C}_3\colon f_3(x,y) &= 0.1361x^3 - 0.6368x^2y + 0.6121xy^2 - 0.1269y^3 \\ &\qquad - 0.3765x^2 + xy - 0.3812y^2 + 0.3657x - 0.3816y - 0.1280 \\ &= 0\end{aligned}$$

up to a tolerance of $\mathsf{TOL} = 10^{-8}$ (see Figure 5.3.3c); we have

$$\max_{(x,y)^\top \in P} |f_3(x,y)| \approx 4.905 > \mathsf{TOL}.$$

$n = 4$: We randomly choose $m = 14$ points from point cloud P:

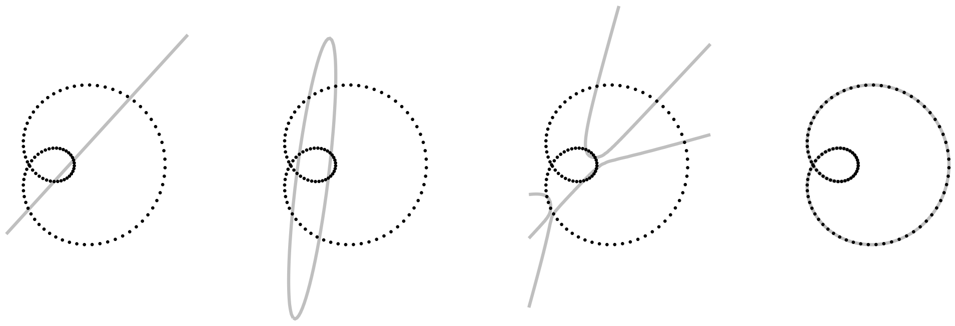
k	x_k	y_k
1	0.6336502032010924	−0.36606057164931516
2	0.8919996499197558	−0.24930393291812158
3	0.18917628065488942	1.2787696513841564
4	2.9988757784209676	−0.06360733730558996
5	−0.038149399652683026	0.9159806679983081
6	0.22195976803923528	−1.3144092689217712
7	0.9223067627678522	0.21606894691608589
8	1.5080577401686142	1.7461721201951905
9	0.5909862893052821	−1.5903956586621792
10	0.7775259536846537	−0.32708270617380014
11	0.8199652707774099	0.30469665211710856
12	0.018435613975493490	−0.03045891072577873
13	0.4140819900035717	1.4815690462555489
14	0.7369743269307526	−1.6564449122519318

We solve the underdetermined homogeneous linear system of equations

$$\begin{pmatrix} x_1^4 & x_1^3y_1 & x_1^2y_1^2 & x_1y_1^3 & y_1^4 & \cdots & x_1 & y_1 & 1 \\ x_2^4 & x_2^3y_2 & x_2^2y_2^2 & x_2y_2^3 & y_2^4 & \cdots & x_2 & y_2 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ldots & \vdots & \vdots & \vdots \\ x_{14}^4 & x_{14}^3y_{14} & x_{14}^2y_{14}^2 & x_{14}y_{14}^3 & y_{14}^4 & \cdots & x_{14} & y_{14} & 1 \end{pmatrix} \boldsymbol{x} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

for coefficient vector $\boldsymbol{x}$. We obtain

$$\begin{aligned}\boldsymbol{x} \approx (&-0.2500, 1.624 \times 10^{-15}, -0.5000, -6.581 \times 10^{-15}, -0.2500, 1, 1.880 \times 10^{-15}, \\ &1, 5.528 \times 10^{-15}, -0.7500, 5.353 \times 10^{-15}, 0.2500, -1.651 \times 10^{-14}, \\ &-7.741 \times 10^{-15}, 5.480 \times 10^{-17})^\top.\end{aligned}$$



(a) $n = 1$ (b) $n = 2$ (c) $n = 3$ (d) $n = 4$

Figure 5.3.3: limaçon point cloud and candidate real algebraic loci

All points of the point cloud lie on the quartic

$$\begin{aligned}\mathcal{C}_4\colon f_4(x,y) &= -0.2500x^4 + 1.624\times 10^{-15}x^3y - 0.5000x^2y^2 - 6.581\times 10^{-15}xy^3\\ &\qquad - 0.2500y^4 + x^3 + 1.880\times 10^{-15}x^2y + xy^2 + 5.528\times 10^{-15}y^3\\ &\qquad - 0.7500x^2 + 5.353\times 10^{-15}xy + 0.2500y^2\\ &\qquad - 1.651\times 10^{-14}x - 7.741\times 10^{-15}y + 5.480\times 10^{-17}\\ &= 0\end{aligned}$$

up to a tolerance of $\mathsf{TOL} = 10^{-8}$ (see Figure 5.3.3d); we have

$$\max_{(x,y)^\top \in P} |f_4(x,y)| \approx 4.517\times 10^{-14} < \mathsf{TOL}.$$

Therefore, we stop. Indeed, $f_4(x,y) = 0$ is reasonably close to the exact equation

$$\left(x^2+y^2-2x\right)^2 - (x^2+y^2) = x^4 + 2x^2y^2 + y^4 - 4x^3 - 4xy^2 + 3x^2 - y^2 = 0,$$

divided by 4.

Remark 5.3.4. In contrast to the algorithm of (Lebmeir and Richter-Gebert, 2007), Algorithm 5.2.2 does not attempt to find curves of degree $n = 1, 2, 3$ that best approximate *all* points of the point cloud (compare Figure 5.3.3 and (Lebmeir and Richter-Gebert, 2007, Figure 3)). When we reach the right degree, Algorithm 5.2.2 approximates the curve much better than before. If we use only $m = n(n+3)/2$ points to determine approximate curves, we artificially increase the difference of approximation quality between the approximate curve of the right degree and its predecessor. Consequently, we could also use a significant drop of $\max_{(x,y)^\top\in P} |f(x,y)|$ as a stopping criterion.

5.3.2 Limaçon (rational parameterization)

Using Algorithm 5.2.10, we can (approximately) determine a rational parameterization for the limaçon from the previous subsection (Construction 5.3.2 with $a = 2$

and $b = 1$): In Construction 5.3.2, we could parameterize the motion of point C along circle c_0 by

$$C(\varphi) = (a, 0)^\top + b \cdot (\cos\varphi, \sin\varphi)^\top, \quad -\pi \leq \varphi \leq \pi.$$

However, this parameterization is not rational. We use tangent half-angle substitution,

$$t = \tan\frac{\varphi}{2}, \quad \cos\varphi = \frac{1-t^2}{1+t^2}, \quad \sin\varphi = \frac{2t}{1+t^2},$$

to derive the rational parameterization

$$C(t) = \left(a + b \cdot (1-t^2)/(1+t^2), b \cdot 2t/(1+t^2)\right)^\top, \quad t \in \mathbb{R}.$$

Together with every point $(x, y)^\top$ of the point cloud, we store the corresponding parameter value t. We use Algorithm 5.2.10 to determine algebraic equations $g(t, x) = 0$ and $h(t, y) = 0$ that describe the relation between t and x respectively t and y. Since the limaçon is a rational plane algebraic curve, we can find algebraic equations $g(t, x)$, $h(t, y)$ that are linear in x respectively y. For the degree in t, we would generally test degrees $n = 1, 2, \ldots$ successively; here, we elaborate only the decisive step of the procedure:

$n = 4$: We randomly choose $m = 2(n+1) - 1 = 9$ points from point cloud P:

k	t_k	x_k	y_k
1	4.260998791303504	0.7085849638314798	−0.35197717330088585
2	3.730086748535029	0.6336502032010924	−0.36606057164931516
3	7.293038268798594	0.8919996499197558	−0.24930393291812158
4	0.8629471643344457	0.18917628065488942	1.2787696513841564
5	−0.010604004580936474	2.9988757784209676	−0.06360733730558996
6	1.042515626323568	−0.038149399652683026	0.9159806679983081
7	−0.8452911790931055	0.22195976803923528	−1.3144092689217712
8	−8.652722941075382	0.9223067627678522	0.21606894691608589
9	0.45767653215895854	1.5080577401686142	1.7461721201951905

We solve the underdetermined homogeneous linear systems of equations

$$\begin{pmatrix} t_1^4 x_1 & t_1^3 x_1 & t_1^2 x_1 & t_1 x_1 & x_1 & -t_1^4 & -t_1^3 & -t_1^2 & -t_1 & -1 \\ t_2^4 x_2 & t_2^3 x_2 & t_2^2 x_2 & t_2 x_2 & x_2 & -t_2^4 & -t_2^3 & -t_2^2 & -t_2 & -1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \cdots & \vdots & \vdots & \vdots & \vdots \\ t_9^4 x_9 & t_9^3 x_9 & t_9^2 x_9 & t_9 x_9 & x_9 & -t_9^4 & -t_9^3 & -t_9^2 & -t_9 & -1 \end{pmatrix} \boldsymbol{x}_g = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix},$$

$$\begin{pmatrix} t_1^4 y_1 & t_1^3 y_1 & t_1^2 y_1 & t_1 y_1 & y_1 & -t_1^4 & -t_1^3 & -t_1^2 & -t_1 & -1 \\ t_2^4 y_2 & t_2^3 y_2 & t_2^2 y_2 & t_2 y_2 & y_2 & -t_2^4 & -t_2^3 & -t_2^2 & -t_2 & -1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \cdots & \vdots & \vdots & \vdots & \vdots \\ t_9^4 y_9 & t_9^3 y_9 & t_9^2 y_9 & t_9 y_9 & y_9 & -t_9^4 & -t_9^3 & -t_9^2 & -t_9 & -1 \end{pmatrix} \boldsymbol{x}_h = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

for coefficient vectors $\boldsymbol{x}_g, \boldsymbol{x}_h$. We obtain

$$\begin{aligned}\boldsymbol{x}_g &\approx (-0.2500, 7.109 \times 10^{-15}, -0.5000, 3.652 \times 10^{-15}, -0.2500, -0.2500,\\ &\qquad 6.254 \times 10^{-15}, 1, -2.449 \times 10^{-15}, -0.7500)^\top,\\ \boldsymbol{x}_h &\approx 0.1667, -1.332 \times 10^{-15}, 0.3333, -2.374 \times 10^{-15}, 0.1667, 1.011 \times 10^{-16},\\ &\qquad -0.3333, -5.664 \times 10^{-15}, 1, 1.497 \times 10^{-17}{}^\top.\end{aligned}$$

All points $(t, x, y)^\top$ of the point cloud lie on the quintics

$$\begin{aligned}\mathcal{C}_g\colon g(t,x) &= -0.2500t^4x + 7.109 \times 10^{-15}t^3x - 0.5000t^2x + 3.652 \times 10^{-15}tx\\ &\qquad - 0.2500x - 0.2500t^4 + 6.254 \times 10^{-15}t^3 + t^2 - 2.449 \times 10^{-15}t\\ &\qquad - 0.7500\\ &= 0,\end{aligned}$$

$$\begin{aligned}\mathcal{C}_h\colon h(t,y) &= 0.1667t^4y - 1.332 \times 10^{-15}t^3y + 0.3333t^2y - 2.374 \times 10^{-15}ty\\ &\qquad + 0.1667y - 1.011 \times 10^{-16}t^4 + 0.3333t^3 + 5.664 \times 10^{-15}t^2 - t\\ &\qquad - 1.497 \times 10^{-17}\\ &= 0\end{aligned}$$

up to a tolerance of $\mathsf{TOL} = 10^{-8}$; we have

$$\max_{(t,x,y)^\top \in P} |g(t,x)| = 1.035 \times 10^{-9} < \mathsf{TOL},$$

$$\max_{(t,x,y)^\top \in P} |h(t,y)| = 7.490 \times 10^{-9} < \mathsf{TOL},$$

Therefore, we stop. Indeed, $g(t,x) = 0$ and $h(t,y) = 0$ are reasonably close to the exact equations

$$\begin{aligned}(t^4x + 2t^2x + x - t^4 + 4t^2 - 3)/4 = 0,\\ (t^4y + 2t^2y + y + 2t^3 - 6t)/6 = 0.\end{aligned}$$

We solve $g(t,x) = 0$ for x and $h(t,y) = 0$ for y. This yields a rational parameterization that is a close approximation of the exact rational parameterization of the limaçon with parameters $a = 2$, $b = 1$ from Construction 5.3.2,

$$x(t) = \frac{t^4 - 4t^2 + 3}{t^4 + 2t^2 + 1}, \quad y(t) = \frac{-2t^3 + 6t}{t^4 + 2t^2 + 1}, \quad t \in \mathbb{R}.$$

6 GPU-based Visualization of Algebraic Riemann Surfaces

6.1 Introduction

Suppose we generate a real algebraic locus by Algorithm 4.2.1 and approximate the equation of the real plane algebraic curve that contains the locus by Algorithm 5.2.2. Then we may want to visualize the corresponding *complex* plane algebraic curve as a Riemann surface on which the tracer moves along the complex detours of Algorithm 4.2.1. Ideally, such a visualization can help improve our understanding of the complex structure of algebraic loci.

6.1.1 Mathematical background

The following basic example illustrates what we would like to visualize.

Example 6.1.1. Let y be the square root of x,

$$y = \sqrt{x}.$$

If x is a non-negative real number, we typically define y as the non-negative real number whose square equals x, i.e. we always choose the non-negative solution of the equation

$$y^2 - x = 0 \tag{6.1}$$

as $y = \sqrt{x}$. For negative real numbers x, no real number y solves Equation 6.1. However, if we define the imaginary unit i as a number with the property that $\mathrm{i}^2 = -1$ then the square root of x becomes the purely imaginary number

$$y = \mathrm{i}\sqrt{|x|}.$$

Together, these two conventions yield a continuous square root function

$$\sqrt{\cdot}\colon \mathbb{R} \to \mathbb{C}.$$

For complex numbers x, Equation 6.1 has exactly two complex solutions (counted with multiplicity), the square roots of x.

We have seen that, for real numbers x, we can choose one solution of Equation 6.1 for the square root and obtain a square root function that is continuous over the real numbers. In contrast, we cannot for every complex number x choose one solution of Equation 6.1 so that we obtain a square root function that is continuous over the complex numbers: If we plot the two solutions of Equation 6.1 as x runs along a circle centred at the origin of the complex plane, we observe that y moves at half the angular velocity of x (see Figure 6.1.2). When x completes one full circle and reaches its initial position again, the square roots have interchanged signs. Therefore, a discontinuity occurs when x returns to its initial position after one full turn and the square root jumps back to its initial position.

Note that by choosing the values of the square root in a different manner, or, equivalently, letting x start at at a different position, we can move the discontinuity

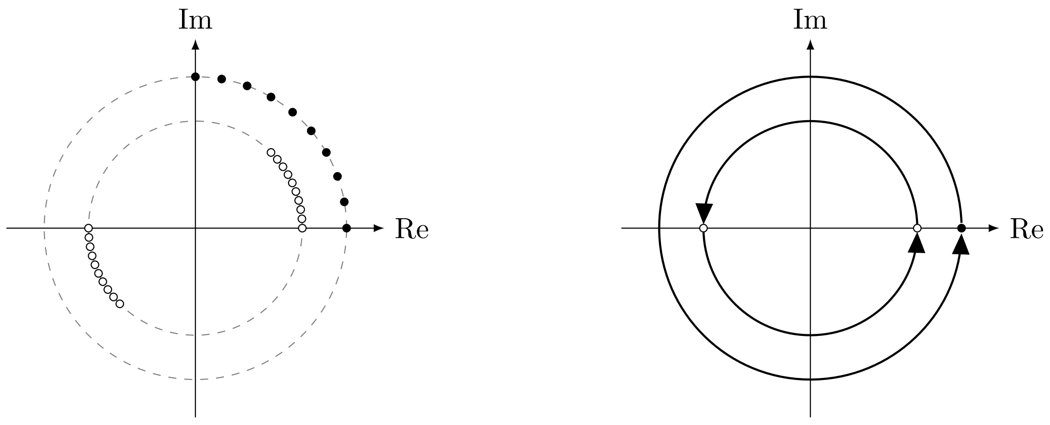


Figure 6.1.2: When a complex number (black points) runs along a circle centred at the origin of the complex plane, its square roots (white points) move at half the angular velocity (left image). After a full turn of x, the square roots have interchanged positions. The real part of the square root that initially had positive real part has become negative, and vice versa (right image).

to an arbitrary position on the circle. Moreover, note that there is (at least) one discontinuity on any circle of any radius centred at the origin.

In order to define the principal branch of the complex square root function, we usually align the discontinuities along the negative real axis, the canonical branch cut of the complex square root function, and choose those values that on the real axis agree with the square root over the real numbers.

Alternatively, we can extend the domain of the complex square root to make it a single-valued and continuous function. To that end, we take two copies of the extended complex plane and slit them along the negative real axis. On the first copy, we choose the solution of Equation 6.1 with non-negative real part as the complex square root of x; on the second copy, we choose the other solution. We glue the upper side (lower side) of the slit of the first copy to the lower side (upper side) of the slit of the second copy and obtain a Riemann surface of the complex square root. (In three dimensions, this is not possible without self-intersections.)

Remark 6.1.3. On the Riemann surface, the complex square root is single-valued and continuous. It is even analytic except at the origin and at infinity, which are exactly the points where the two solutions of Equation 6.1 coincide.

Remark 6.1.4. The branch cut is not a special curve of the Riemann surface. When we glue the Riemann surface together, the branch cut becomes a curve like every other curve on the Riemann surface. If we had used a different curve between the origin and infinity as branch cut, we would have obtained the same result.

Remark 6.1.5. Equation 6.1 describes a parabola. We can proceed analogously to obtain Riemann surfaces for other plane algebraic curves.

6.1.2 Previous work

Probably the most common approach for visualization of functions is to plot a function graph. However, for a complex function

$$g\colon \mathbb{C} \to \mathbb{C},$$

the function graph

$$\{(z, g(z)) \mid z \in \mathbb{C}\} \subset \mathbb{C} \times \mathbb{C} \simeq \mathbb{R}^2 \times \mathbb{R}^2$$

is a (real) two-dimensional surface in (real) four-dimensional space.

One way to visualize a four-dimensional object is to plot several two- or three-dimensional slices. This approach seems less useful for understanding the overall structure of the object.

Another traditional method to visualize complex functions is domain colouring. The principle of domain colouring is to colour every point in the domain of a function with the colour of its function value in a reference image. If we choose the reference image wisely, a lot of information about the complex function can be read off from the resulting two-dimensional image (see e.g. (Poelke and Polthier, 2012) and (Wegert, 2012)). The idea of lifting domain colouring to Riemann surfaces is due to Poelke and Polthier (2009).

We can interpret a Riemann surface of a plane algebraic curve

$$\mathcal{C}\colon f(x, y) = 0 \tag{6.2}$$

as a function graph of a multivalued complex function, which maps every x to multiple values of y. If $f(x, y)$ is a polynomial of degree n in y, there are exactly n values of y for every value of x that satisfy $f(x, y) = 0$ (counted with multiplicity). Every such pair (x, y) corresponds to a point on the Riemann surface. In other words, the Riemann surface is an n-fold cover of the complex plane.

Let $\pi\colon (x, y) \mapsto x$ denote a projection function on the Riemann surface. Then the values of y at x correspond to the elements of the fibre $\pi^{-1}(x)$. The situation is analogous to function graphs of single-valued functions from the real numbers (or the real plane) to the real numbers, where one function value lies above every point in the domain.

We can transfer the Riemann surface from (real) four-dimensional space into (real) three-dimensional space by introducing a height function $H\colon \mathbb{C} \to \mathbb{R}$. We typically use the real part as a height function. We plot the surface

$$\{(\operatorname{Re} x, \operatorname{Im} x, H(y)) \mid x, y \in \mathbb{C}\colon f(x, y) = 0\}$$

and use domain colouring to represent the value of y at every point of the surface.

In practice, we want to generate a triangle mesh that approximates the Riemann surface as the graph of a multivalued function over a triangulated domain in the complex plane. The Riemann surface mesh approximates the continuous Riemann surface in the following sense: The y-values at the vertices of a triangle of the Riemann surface mesh result from each other under analytic continuation along the edges of the underlying triangle in the triangulated domain. If $f(x, y) = 0$ is a polynomial of degree n in y there are n values of y above every vertex x of the triangulated domain. Hence, we have to determine which of the $3n$ values of y above a triangle in the triangulated domain form triangles of the Riemann surface mesh. A wrong combination of values of y to triangles might for example occur due to discontinuity if we used the principal branch of the square root function for the computation of y. This would produce artefacts in the visualization for which there is no mathematical justification.

For the generation of such a Riemann surface mesh, previous algorithms have solved systems of differential equations (Trott, 2008; Nieser et al., 2010) or explicitly identified and analyzed branch cuts to remove discontinuities (Kranich, 2012; Wegert, 2012, Section 7.6).

In the next section, we discuss an algorithm based on a different idea: We can exploit that $y(x)$ is continuous almost everywhere on the Riemann surface and therefore, if x changes little, so does $y(x)$.

6.2 Algorithms

In this section, we describe algorithms for generating and visualizing domain-coloured Riemann surface meshes of plane algebraic curves. Let

$$\mathcal{C}\colon f(x, y) = 0 \tag{6.3}$$

be a complex plane algebraic curve. In particular, let f be a polynomial with complex coefficients of degree n in y. Moreover let $U \subset \mathbb{C}$ be a triangulated domain in the complex plane. (In practice, U is typically rectangular.)

We want to generate a Riemann surface mesh of $\mathcal{C}$. The mesh discretizes a part of a (real) two-dimensional surface in (real) four-dimensional space. We can visualize it using a height function and domain colouring, as described in the previous section.

We obtain a Riemann surface mesh of $\mathcal{C}$ as a graph of the multivalued function induced by Equation 6.3, which maps every value of x in U to n values of y such that $f(x, y) = 0$.

For every triangle in U, we thus obtain n values of y at each of its three vertices. The problem is to determine whether, and if so, how, the $3n$ values of y can be combined to form triangles of the Riemann surface mesh. The resulting triangles should be consistent with the fact that y as a function of x is analytic almost everywhere on the Riemann surface. This is impossible if the triangle in U contains a ramification point of $y(x)$. In this case, we subdivide the triangle to obtain smaller triangles mostly free of ramification points. Otherwise, the triangles of the Riemann surface mesh are uniquely determined by analytic continuation of $y(x)$ along the edges of the triangle in U.

In order to find these triangles of the Riemann surface mesh, we use the following idea: Consider a triangle $\triangle x_1x_2x_3$ in U that is free of ramification points of $y(x)$. Under this assumption, $y(x)$ is continuous on those parts of the Riemann surface that lie above $\triangle x_1x_2x_3$. Hence, for every $\varepsilon > 0$ there exists $\delta > 0$ such that $|y(x_1) - y(x_2)| < \varepsilon$ for all x_1, x_2 with $|x_1 - x_2| < \delta$. If ε is half the minimum distance between the n values of $y(x)$ at x_1 and $|x_1 - x_2|$ is smaller than the corresponding δ, then the values of $y(x)$ at x_2 are closer to the corresponding values of $y(x)$ at x_1 than to any other value of $y(x)$ at x_1.

In other words, if triangle $\triangle x_1x_2x_3$ is small enough, we can combine the values of y at its vertices to triangles of the Riemann surface mesh based on proximity: Among the $3n$ values of y at the vertices of triangle $\triangle x_1x_2x_3$, every three values of y closest to each other form a triangle of the Riemann surface mesh.

We can algorithmically compute a $\delta > 0$ as above using the epsilon-delta bound for plane algebraic curves of Theorem 2.2.1. Theorem 2.2.1 is of essential importance for our approach. Our approach only works because Theorem 2.2.1 provides us with a reliable bound computable as a function of x that depends only on a few constants derived from the coefficients of $f(x, y)$.

If triangle $\triangle x_1x_2x_3$ is not small enough to correctly combine the values of y at its vertices based on proximity, we subdivide the triangle.

In summary, we obtain the following algorithm:

Algorithm 6.2.1 (Generation of a Riemann surface mesh)**.** Let $U \subset \mathbb{C}$ be a triangulated domain in the complex plane. Let

$$\mathcal{C}\colon\ f(x, y) = 0$$

be a complex plane algebraic curve and $f(x, y)$ a polynomial of degree n in y. We prescribe a maximal subdivision depth (as a maximal number of iterations or as a minimal edge length).

1. Compute the global ingredients of the epsilon-delta bound of Theorem 2.2.1 for $y(x)$.
2. For every triangle $\triangle x_1x_2x_3$ in U:
   a) Compute the $3n$ values of $y(x)$ at x_1, x_2, x_3,
   $$\{y_k(x_j) \mid f(x_j, y_k(x_j)) = 0,\ j = 1, 2, 3,\ k = 1, 2, \dots, n\}.$$
   b) Compute half the minimum distance between the values of $y(x)$ at each of the vertices of $\triangle x_1x_2x_3$,
   $$\varepsilon(x_j) = \tfrac{1}{2} \min_{k \neq l} |y_k(x_j) - y_l(x_j)|, \quad j = 1, 2, 3.$$
   c) Compute $\delta(x_j)$ by the epsilon-delta bound of Theorem 2.2.1 so that
   $$|y(x_j) - y(x)| < \varepsilon(x_j),\ \text{if } |x_j - x| < \delta(x_j), \quad j = 1, 2, 3.$$

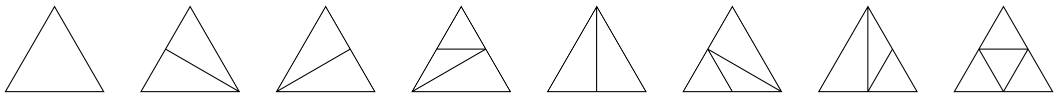


Figure 6.2.2: Adaptive refinement patterns used in Algorithm 6.2.1

   d) Determine which of the edges of $\triangle x_1x_2x_3$ are longer than the minimum of the $\delta(x_j)$ at their endpoints and must be subdivided.

   e) Select the right adaptive refinement pattern (see Figure 6.2.2) and subdivide $\triangle x_1x_2x_3$ accordingly.

3. Repeat step 2 until the maximal subdivision depth is reached.

4. Discard every triangle in U with an edge longer than the minimum of the $\delta(x_j)$ at its endpoints.

5. For every triangle $\triangle x_1x_2x_3$ in U, combine the values of $y(x)$ at its vertices to triangles of the Riemann surface mesh based on proximity. More formally, the triangles added to the Riemann surface mesh comprise the vertices

$$\begin{gathered}(x_1, y_k(x_1)),\\ (x_2, \operatorname*{argmin}_{y_l(x_2)} |y_k(x_1) - y_l(x_2)|),\\ (x_3, \operatorname*{argmin}_{y_l(x_3)} |y_k(x_1) - y_l(x_3)|),\end{gathered}$$

   for $k = 1, 2, \ldots, n$.

6. Output the Riemann surface mesh and stop.

Remark 6.2.3. By construction, Algorithm 6.2.1 generates a Riemann surface mesh that is consistent with the analytic structure of the Riemann surface of $\mathcal{C}$.

Remark 6.2.4. The adaptive refinement patterns used for the subdivision of triangles, whose edges are too long, produce a watertight subdivision.

Remark 6.2.5. Step 4 of Algorithm 6.2.1 produces holes around the ramification points of $y(x)$. We can make these holes very small if we choose the maximal subdivision depth appropriately.

For the visualization of a Riemann surface mesh, we use the following algorithm:

Algorithm 6.2.6 (Visualization of a Riemann surface mesh)**.** Let a Riemann surface mesh and a domain colouring reference image be given. We choose a height function $H\colon \mathbb{C} \to \mathbb{R}$ to transform a point on the Riemann surface mesh from (real) four-dimensional space to a point in (real) three-dimensional space,

$$(x, y(x)) \mapsto (\operatorname{Re} x, \operatorname{Im} x, H(y(x))).$$

1. Draw the mesh that results from transforming every vertex of the Riemann surface mesh as above.
2. Interpolate the value of $y(x)$ on the transformed mesh.
3. Assign to every point on the transformed mesh the colour in the reference image of the value that $y(x)$ attains at that point on the transformed mesh.

Remark 6.2.7. If we choose the real (or imaginary) part of $y(x)$ as a height function, the transformation from (real) four-dimensional to (real) three-dimensional space becomes a projection.

Remark 6.2.8. Using the real part of $y(x)$ as a height function has the advantage that the visualization then contains the image of $\mathcal{C}$ interpreted as a real plane algebraic curve. It is the intersection of the visualization of the Riemann surface mesh in (real) three-dimensional space with the $\operatorname{Re} x$-$\operatorname{Re} y$-plane (the xz-plane, if we label the coordinate axes of real three-dimensional space such that the x-axis points to the right and the z-axis points upwards).

Remark 6.2.9. The computation of the Riemann surface mesh by Algorithm 6.2.1 is independent of the choice of height function used for its visualization.

6.3 Implementation

In this section, we discuss how Algorithm 6.2.1 and Algorithm 6.2.6 can be implemented using OpenGL and WebGL. Since WebGL targets a much wider range of devices, its API is more limited than that of OpenGL. Consequently, our implementation using WebGL differs substantially from our implementation using OpenGL. Before we discuss each setup separately, let us talk about what they have in common.

The main part of our programs is written in shading language (GLSL for OpenGL and ESSL for WebGL) and runs on the GPU.[1] We use the CPU to compute the global ingredients for the epsilon-delta bound, to generate shading language code that computes the epsilon-delta bound for $\mathcal{C}\colon f(x,y)=0$ as a function of x, and to generate a coarse triangulation of the input domain.

The implementations in OpenGL and WebGL share some shading language code. Since there is no native type for complex numbers, we represent them using two-dimensional floating point vectors. Common routines include complex arithmetic, numerical root-finding algorithms, the computation of the epsilon-delta bound, and domain colouring.

The implementation of complex arithmetic is straightforward and we shall not go into detail about it.

[1]A shading language program consists of different shaders (vertex, geometry, or fragment shaders) that operate on different primitives (vertices, triangles, or fragments of a pixel). In a shader, we specify how a single primitive should be processed; the GPU automatically executes each shader for many primitives in parallel. This parallelization makes GPU-based algorithms really fast. Therefore, we try to maximize the amount of code that runs on the GPU, and try to avoid using the CPU.

We need numerical root-finding algorithms to approximate roots of polynomials in order to compute values of $y(x)$ (and to compute the global ingredients of the epsilon-delta bound). For instance, Laguerre's method (Press et al., 2007, Section 9.5.1) and deflation (Press et al., 2007, Section 9.5.3) or Weierstraß–Durand–Kerner method (Weierstraß, 1891; Durand, 1960; Kerner, 1966) are well-suited. The latter may be a little easier to implement in shading language (due to the absence of variable-length arrays).

For the computation of the epsilon-delta bound, see Theorem 2.2.1. Note that the computation is parallelizable since the epsilon-delta bound can be implemented as a function of x that depends on only a few constants derived from the coefficients of $f(x, y)$.

Instead of computing texture coordinates, which would depend on the range of $y(x)$ on the input domain, we generate the domain colouring procedurally on-the-fly. To that end, we use a variation of the enhanced phase portrait colour scheme of (Wegert, 2012, Section 2.5). The reference image is shown in Figure 6.4.1. We discuss the colour scheme in Section 6.4.1.

The main difference between the implementations in OpenGL and WebGL is how the common routines can be combined to realize Algorithm 6.2.1 and Algorithm 6.2.6.

6.3.1 An implementation in OpenGL

Implementation of Algorithm 6.2.1 in OpenGL

Our implementation of Algorithm 6.2.1 in OpenGL comprises three GLSL programs, for initialization, subdivision, and assembly of the Riemann surface mesh. We cache the output of each program using transform feedback[2] and feed it back to the next program.

The initialization program consists only of a vertex shader, which operates on the vertices of the triangulated input domain. For every vertex x, we compute $y_k(x)$, $k = 1, 2, \ldots, n$, and $\delta(x)$.

After initialization, we run the subdivision program. The program consists of a pass-through vertex shader and a geometry shader. The geometry shader operates on the triangles of the triangulated input domain or of its last subdivision, respectively. We have access to the values of x_j, $\delta(x_j)$, and $y_k(x_j)$, $k = 1, 2, \ldots, n$, $j = 1, 2, 3$, at the vertices of each triangle $\triangle x_1 x_2 x_3$. We determine which edges of triangle $\triangle x_1 x_2 x_3$ are longer than the minimum of the $\delta(x_j)$ at their endpoints. In order to subdivide these edges, we compute their midpoints x, and $\delta(x)$ and $y_k(x)$, $k = 1, 2, \ldots, n$, at the midpoints. We use the appropriate adaptive refine pattern of Figure 6.2.2 and output between one and four triangles for every input triangle. In doing so, we reuse previously computed values rather than recomputing them.

[2]Transform feedback is a mechanism that allows us to store the output of a GLSL program in GPU memory and to reuse it as the input to another GLSL program. The mechanism is more efficient than some other approaches because it reduces expensive data transfer between GPU and CPU memory.

We run the subdivision program iteratively until we reach the prescribed maximal subdivision depth.

The assembly program consists of a pass-through vertex shader and a geometry shader. The geometry shader operates on the triangles of the adaptively subdivided input domain. We again have access to the values of x_j, $\delta(x_j)$, and $y_k(x_j)$, $k = 1, 2, \ldots, n$, $j = 1, 2, 3$, at the vertices of each triangle $\triangle x_1 x_2 x_3$. For every triangle $\triangle x_1 x_2 x_3$, we test whether one of its edges is longer than the minimum of the $\delta(x_j)$ at its endpoints. In this case, we discard the triangle. Otherwise, we determine the triangles of the Riemann surface mesh by proximity (see Algorithm 6.2.1, step 5) and output these n triangles.

We also cache the assembled Riemann surface mesh using transform feedback so that we can pass it as input to our implementation of the visualization algorithm (Algorithm 6.2.6).

Implementation of Algorithm 6.2.6 in OpenGL

Our implementation of Algorithm 6.2.6 in OpenGL consists of one GLSL program with a vertex and a fragment shader.

The vertex shader operates on the vertices of a Riemann surface mesh generated by our implementation of Algorithm 6.2.1. We apply height function $H : \mathbb{C} \to \mathbb{R}$ to map each (real) four-dimensional vertex

$$(\operatorname{Re} x, \operatorname{Im} x, \operatorname{Re} y(x), \operatorname{Im} y(x))$$

to a (real) three-dimensional vertex

$$(\operatorname{Re} x, \operatorname{Im} x, H(y(x))).$$

We homogenize the coordinates of this vertex and transform them using the model-view-projection matrix. We pass $y(x)$ as a varying variable to the fragment shader.

The fragment shader operates on the interpolated value of $y(x)$ at a fragment of a pixel of the output device. We compute the colour of $y(x)$ according to our domain colouring reference image.

Remarks

Using our implementation, the generation of a Riemann surface mesh takes little but noticeable time. The bottlenecks of the implementation are numerical root-finding and iterative subdivision. However, if we use transform feedback to cache the Riemann surface mesh and pass it to the implementation of the visualization algorithm, we obtain interactive performance.

Another advantage of using transform feedback to cache the Riemann surface mesh is that we can easily export the data. If we additionally compute texture coordinates and a high-resolution reference image, we can even print our visualization using a full colour 3D printer (see Figure 6.3.1).

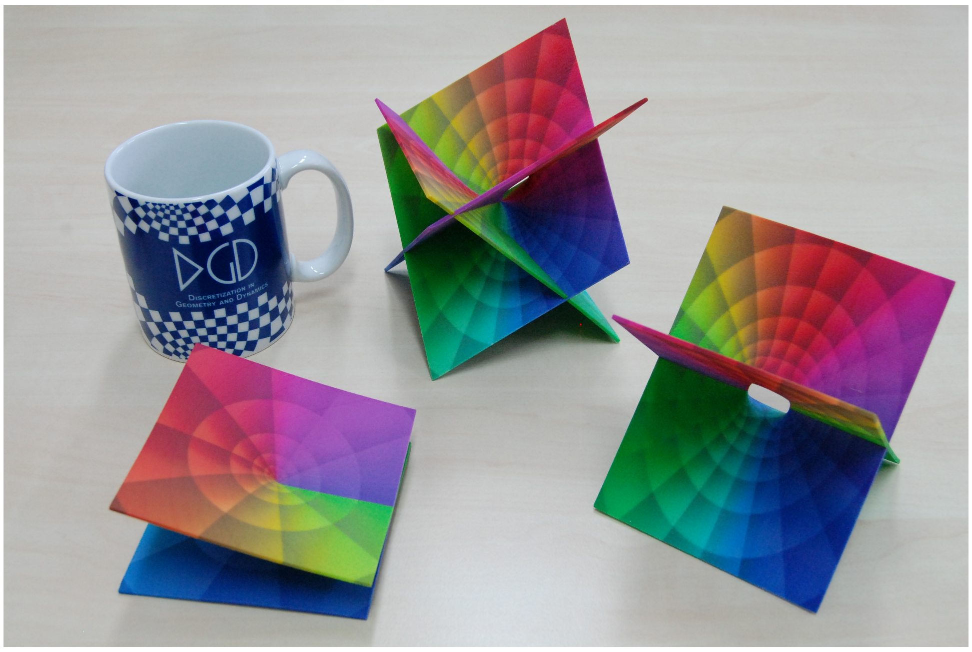


Figure 6.3.1: 3D-printed models of domain-coloured Riemann surfaces of square root, folium of Descartes, and unit circle (clockwise). The merchandise coffee mug of DFG Collaborative Research Center TRR 109, "Discretization in Geometry and Dynamics", is included in the picture as an indicator for the size of the models.

6.3.2 An implementation in WebGL

In order to support a wider range of devices, the WebGL API is much more limited than the OpenGL API. Particularly, in WebGL, geometry shaders and transform feedback are currently unavailable. (The WebGL 2 draft includes transform feedback and compute shaders.) Therefore our implementation in WebGL differs substantially from our implementation in OpenGL.

How to replace transform feedback

Instead of transform feedback, our implementation in WebGL uses floating point textures (specified in the `OES_texture_float` extension) and multiple render targets (specified in the `WEBGL_draw_buffers` extension). I do not claim originality of this approach. It is commonly used for running simulations on the GPU. The original idea may be due to (Crane et al., 2007). We number the vertices of every mesh consecutively and pass this number (index) to the vertex shaders along with the other attributes. In particular, vertices that are shared among several triangles must be duplicated and numbered separately. Hence, we assume that every triangle appears as three consecutive vertices in array buffer storage (triangle soup). We use floating point textures essentially as we would arrays of floats, indexed by vertex number. We store values corresponding to the k-th vertex in the k-th pixel of a texture. We can store up to four floats per pixel of a floating point texture, namely one float each in the red, green, blue, and alpha channel. If we need to store more than four floats, we use multiple render targets which allows us to colour the same pixel of several textures simultaneously.

We want to store values we compute for a vertex in textures ('transform' in transform feedback). To that end, we bind the array buffers and draw the contents as points (as opposed to triangles). In the vertex shader, we compute the positions of the point with index k (in normalized device coordinates) so that it is rasterized as the k-th pixel of the render target textures. Recall that normalized device coordinates range in $[-1,1]^3$. For example, if h and w denote the height and width of the textures in pixels, we assign to the point with index k the position

$$\left(\frac{2\cdot(k \bmod w)+1}{w}-1, \frac{2\cdot\lfloor k/w\rfloor+1}{h}-1, 0\right).$$

In the fragment shader, we compute the values to be stored and assign them as output colours in a specific order (as we later want to retrieve them).

We want to read stored values for a vertex from textures ('feedback' in transform feedback). To that end, we bind the textures and an array buffer containing a range of vertex numbers (indices). We draw the contents of the array buffer as points or triangles (depending on whether we want to send the output to different textures or to the screen). In the vertex shader, we compute texture coordinates for the point with index k which allow us to lookup the k-th pixel from the textures. Recall that texture coordinates range in $[0,1]^2$. For a texture of height h and width w, we compute the texture coordinates

$$\left(\frac{(k \bmod w)+0.5}{w}, \frac{(k \bmod w)+0.5}{h}\right).$$

Adding 0.5 in the numerators accounts for the fact that we want to obtain coordinates for the centre of a pixel in order to avoid interpolation with adjacent pixels. We pass the texture coordinates to the fragment shader, where we can use them to perform a texture lookup.

In order to access data of a whole triangle (as in geometry shaders), we can, in the vertex shader, determine the indices of the other vertices of the triangle. For example, the point with index k is part of the triangle whose vertices have indices

$$k-(k \bmod 3),\ k-(k \bmod 3)+1, \text{ and } k-(k \bmod 3)+2.$$

We compute normalized device coordinates or texture coordinates for all three indices and pass them to the fragment shader, together with the index of the triangle vertex currently under consideration.

How to replace geometry shaders

We replace the geometry shader of the subdivision program of our implementation in OpenGL using a variation of a method proposed by Boubekeur and Schlick (2008). The method works as follows: We precompute all adaptive refinement patterns up to a certain subdivision depth, in our case eight adaptive refinement patterns up to depth one (see Figure 6.2.2). We use barycentric coordinates to store the positions of the triangle vertices of each refinement pattern in an array buffer. Using array buffers of different lengths allows us to achieve variable-length output, as with geometry shaders. For every triangle of a coarse input mesh, we draw the triangles in the array buffer of the appropriate adaptive refinement pattern. We use the vertex positions of the input triangle (read from a texture or from uniform variables) and the barycentric coordinates of the triangles of the adaptive refinement pattern to compute the vertex positions of the output triangle.

We can combine this method with floating point texture and multiple render targets as outlined above, if we number the vertices of each adaptive refinement pattern consecutively and store those indices together with the barycentric coordinates. We pass an offset as a uniform variable to the vertex shader that needs to be added to the indices. We draw the adaptive refinement pattern and increment the offset by the number of vertices in the adaptive refinement pattern.

The geometry shader of the assembly program has fixed-length output. It generates exactly n triangles of the Riemann surface mesh per triangle of the (subdivided) input mesh. We can replace it with n invocations of a vertex shader, one for every sheet of the Riemann surface mesh. We pass the number of the current sheet to the vertex shader as a uniform variable.

Remarks

We cannot expect our WebGL implementation to reach the same performance as our OpenGL implementation. In the subdivision program, since we draw a different adaptive refinement pattern for every triangle of the input mesh, we lose parallelism. Consequently, subdivision in WebGL is much slower than its OpenGL counterpart. However, if we cache the assembled Riemann surface mesh (in textures) and pass it to our implementation of the visualization algorithm, we can still achieve interactive performance.

6.4 Examples

In this section, we discuss domain-coloured Riemann surface meshes for the complex square root function and for the folium of Descartes. Before that, let us explain our domain colouring reference image so that we can interpret the domain-coloured Riemann surface meshes.

6.4.1 Domain colouring reference image

Recall that the basic idea of domain colouring is the following: If we want to visualize a complex function

$$f\colon K \subset \mathbb{C} \to \mathbb{C},$$

we face the problem that its graph is real four-dimensional. However, we can visualize the behaviour of the function by colouring every point in its domain with the colour of the function value at that point in a reference image. The reference image is the domain colouring of the complex identity function.

Depending on what reference image we choose, we can read off various properties of a function from its domain colouring. For an overview of different colour schemes, we refer to (Poelke and Polthier, 2012; Wegert, 2012).

As our reference image, we use a variation of the enhanced phase portrait colour scheme of (Wegert, 2012, Section 2.5). The reference image is best described using polar coordinates

$$r\mathrm{e}^{\mathrm{i}\varphi} = r(\cos\varphi + \mathrm{i}\sin\varphi)$$

of a complex number with *modulus* $r > 0$ and *phase* $\varphi \in [0, 2\pi)$.

Firstly, we encode the phase at any point in the domain as the hue of its colour (in HSI colour space). In a square with side length 10 centred at the origin, we thus obtain the colour wheel shown in Figure 6.4.1a. As the phase changes from 0 to 2π, we obtain every colour of the rainbow. Positive real numbers, which have phase 0, are coloured in pure red. Negative real numbers, which have phase π, are coloured in cyan. Purely imaginary numbers do not have such distinctive colours. (This can be fixed using the NIST continuous phase mapping, which scales the phase piecewise linearly so that purely imaginary numbers with positive imaginary part become yellow and purely imaginary numbers with negative imaginary part become blue. See (NIST, 2014, `http://dlmf.nist.gov/help/vrml/aboutcolor#S2.SS2`). For simplicity, we do not follow this approach here.)

Secondly, we add contour lines of complex numbers of the same phase at integer multiples of 15 degrees (see Figure 6.4.1b). To that end, we change the intensity of the colour by multiplying it with a sawtooth function

$$0.7 + (1.0 - 0.7) \cdot (\varphi/(\pi/12) - \lfloor \varphi/(\pi/12) \rfloor)\,.$$

Because phase corresponds to hue, the points of such a contour line are all of the same colour.

Finally, we add contour lines of complex numbers of the same modulus on a log-scale (see Figure 6.4.1c). To that end, we change the intensity of the colour by multiplying it with a sawtooth function

$$0.7 + (1.0 - 0.7) \cdot (\log(r)/(\pi/12) - \lfloor \log(r)/(\pi/12) \rfloor)\,.$$

Note that the contour lines of phase and modulus intersect each other orthogonally. The scaling factor $1/(\pi/12)$ in the sawtooth function for the modulus contour lines

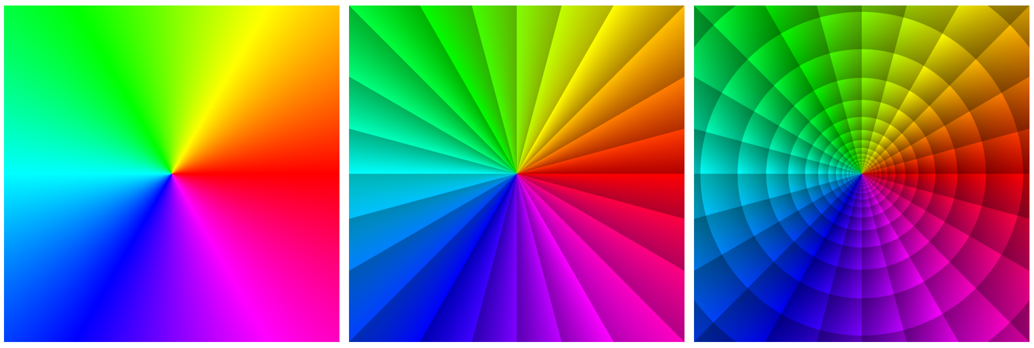


Figure 6.4.1: Composition of domain colouring reference image: If we (a) represent phase by hue, (b) add contour lines of phase and (c) add contour lines of modulus, we obtain our domain colouring reference image, a variation of the enhanced phase portrait colour scheme of (Wegert, 2012, Section 2.5).

deliberately matches the scaling factor used in the sawtooth function for the phase contour lines. Consequently, the regions enclosed by the contour lines of phase and modulus are squarish in appearance.

6.4.2 Complex square root

Recall the construction of a Riemann surface of the complex square root from Section 6.1.1 where we glued together its two branches at a branch cut along the negative real axis.

The domain colouring of the two branches of the complex square root over a square of side length 10 centred at the origin is shown in Figure 6.4.2.

On the sheet shown in Figure 6.4.2a, the complex square root takes values with negative real part (coloured green to blue). On the sheet shown in Figure 6.4.2b, it takes values with positive real part (coloured purple to yellow). (The sheet shown in Figure 6.4.2b corresponds to the principal branch of the complex square root.)

On both sheets, twelve contour lines of phase are visible, half as many as in the reference image. We can see that the phase of the complex square root function changes at half the angular velocity of its argument.

Moreover, the discontinuity at the branch cut along the negative real axis is clearly visible. We also see that there is a smooth transition between the second (third) quadrant of Figure 6.4.2a and the third (second) quadrant of Figure 6.4.2b.

If we cut the two sheets along the negative real axis and glue the upper side of the cut of one sheet to the lower side of the cut of the other sheet, and vice versa, we obtain a Riemann surface of the complex square root. The resulting Riemann surface, produced with Algorithm 6.2.1 and Algorithm 6.2.6 using real part as

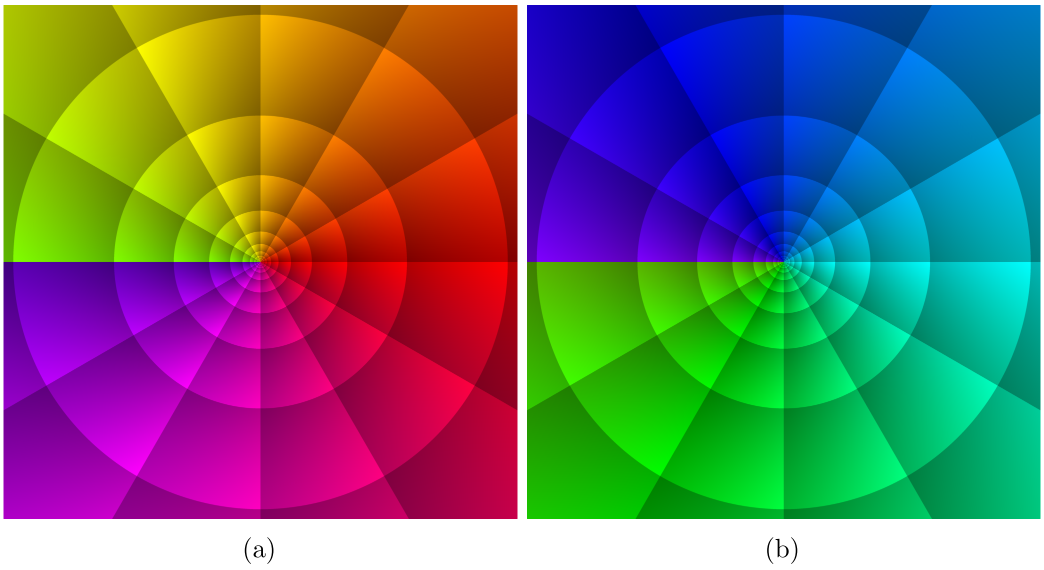


Figure 6.4.2: Domain colouring of the two sheets of the complex square root

height function, is shown in Figure 6.4.3 (perspective) and Figure 6.4.4 (multiview orthogonal).

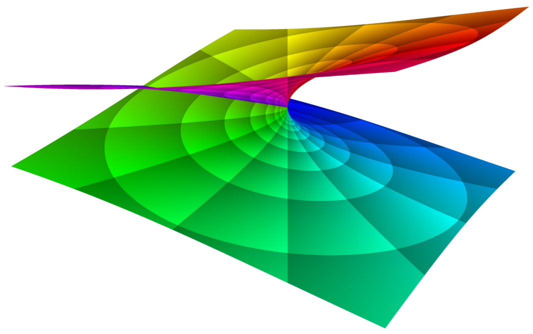


Figure 6.4.3: Domain-coloured Riemann surface of the complex square root in perspective projection

Note that the self-intersection of the surface in Figure 6.4.3 is only an artefact of using a height function to map the Riemann surface mesh from real four-dimensional to real three-dimensional space. Evidently, the two values of the complex square root at each point of the self-intersection do not agree: they are coloured differently, in green and purple, respectively.

In Figure 6.4.4b, we see the parabola that the real parts of the y-values describe according to the equation $y^2 - x = 0$ when x takes values on the non-negative real axis.

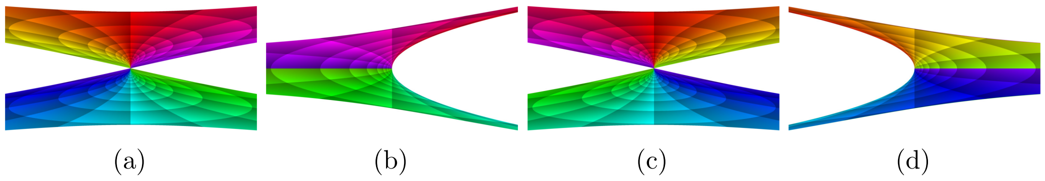


Figure 6.4.4: Domain-coloured Riemann surface of the complex square root in orthogonal projection from left, front, right, and back (from left to right)

6.4.3 Folium of Descartes

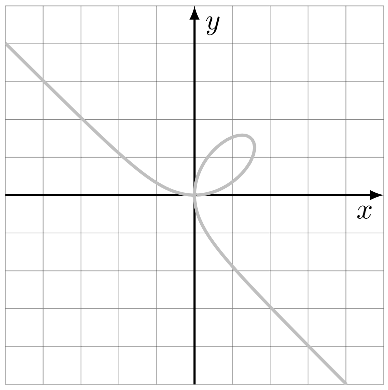
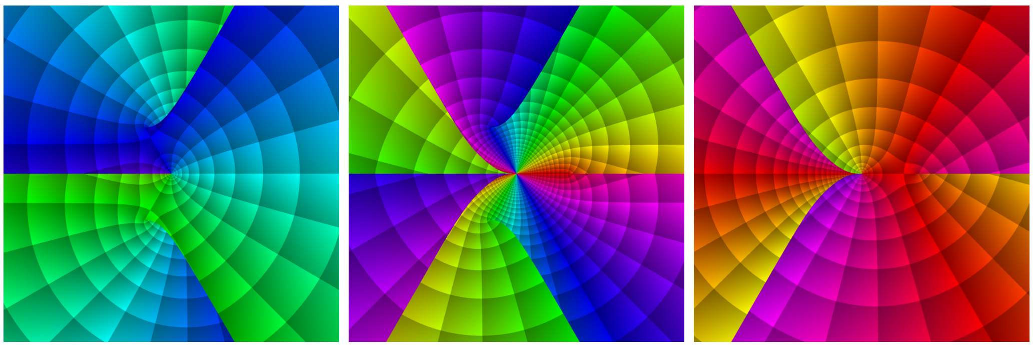


Figure 6.4.5: The folium of Descartes as a real plane algebraic curve

The folium of Descartes is a classical plane algebraic curve of order three,

$$\mathcal{C}\colon\, f(x,y) = x^3 + y^3 - 3xy = 0. \tag{6.4}$$

The cubic curve is nowadays called 'folium' after the leaf-shaped loop that it describes in the first quadrant of the real plane (see Figure 6.4.5). It is named in honour of the French geometer René Descartes (1596–1650), who was among the first mathematicians to introduce coordinates into geometry. Originally, the curve was called *fleur de jasmin* since Descartes and some of his contemporaries, who were working out the principles of dealing with negative and infinite coordinates, initially wrongly believed that the leaf-shaped loop repeated itself in the other quadrants and therefore resembled a jasmine flower (Loria, 1910, p. 53).
Figure 6.4.6 shows three domain-coloured sheets of the folium of Descartes over a square of side length 10 centred at the origin of the complex plane. We can generate these sheets by sorting the y-values that satisfy Equation 6.4 at every point x of the domain according to their real part. The sheet shown in Figure 6.4.6a uses the y-value with the smallest real part, the sheet shown in Figure 6.4.6b the y-value with the second-smallest real part, and the sheet shown in Figure 6.4.6c the y-value with the largest real part.



(a) (b) (c)

Figure 6.4.6: Domain colouring of three sheets of the folium of Descartes

We see that the first sheet carries y-values with negative real part (coloured green to blue). At the centre of the second sheet, we identify a zero of order two, which we can recognize from the fact that the colours of the colour wheel used in our reference image wind around it twice in the same order as in the reference image. It is the node of the leaf-shaped loop. The third sheet carries y-values with positive real part (coloured purple to yellow).

There are three branch cuts (discontinuities of hue) on the first sheet, six on the second sheet and three on the third sheets. We can see how the sheets of the Riemann surface are connected to each other along the branch cuts: First and second sheet are connected at the branch cuts of the first sheet. Second and third sheet are connected at the branch cuts of the third sheet. First and third sheet are not connected directly with each other. (Imagine how much harder it would be to read this off from Equation 6.4.)

Apart from the branch cuts, the map from x to $y(x)$ is conformal (angle-preserving) on every sheet. We can see that the contour lines of phase and modulus intersect each other orthogonally on every sheet, as in our reference image.

If we cut the sheets along the branch cuts and glue them together correctly, we obtain a Riemann surface for the folium of Descartes.

The resulting Riemann surface, produced with Algorithm 6.2.1 and Algorithm 6.2.6 using real part as height function, is shown in Figure 6.4.7 (perspective) and Figure 6.4.8 (multiview orthogonal).

Again, the self-intersections of the surface in Figure 6.4.7 are only an artefact of using a height function to map the Riemann surface mesh from (real) four-dimensional to (real) three-dimensional space.

Figure 6.4.7 makes it obvious that cutting a Riemann surface into sheets by sorting y-values by real part may be the most straightforward but not necessarily the geometrically most appropriate method. Our Riemann surface of the folium of Descartes in large part appears to be composed of three copies of the complex plane (which looks like our reference image). Complications seem to arise only near the origin.

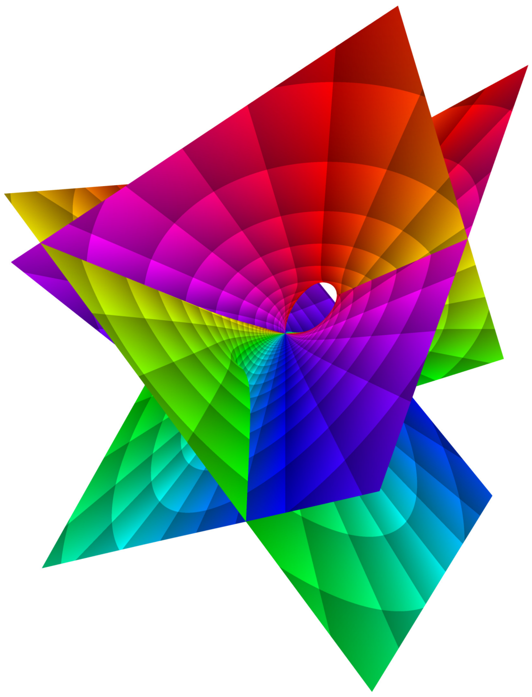


Figure 6.4.7: Domain-coloured Riemann surface of the folium of Descartes in perspective projection

If we look closely at Figure 6.4.8b, we may see how we obtain the real folium of Descartes (as a real plane algebraic curve) as the intersection of our Riemann surface mesh with the $\operatorname{Re} x$-$\operatorname{Re} y$-plane. The leaf-shaped loop is clearly visible as a hole in our visualization. One of the ‘complex planes of which the Riemann surface is composed’ is so thin that it is barely visible from this perspective. It is almost asymptotic to the ‘wings’ of the folium of Descartes (as a real plane algebraic curve) in the second and fourth quadrant of the real xy-plane.

Right below the centre of Figure 6.4.8c, we see two leaf-shaped loops in complex directions. Perhaps Descartes and his contemporaries were not entirely wrong after all to believe that the folium of Descartes has more than one leaf. Indeed, if we let $x' = \mathrm{e}^{\pm \mathrm{i}\pi/3} x$, we discover that in the $\operatorname{Re} x'$-$\operatorname{Re} y$-plane the curve describes a leaf-shaped loop, which is exactly half as high as that in the $\operatorname{Re} x$-$\operatorname{Re} y$-plane (this also holds for the ‘wings’) and rotated into a different quadrant (see Figure 6.4.9).

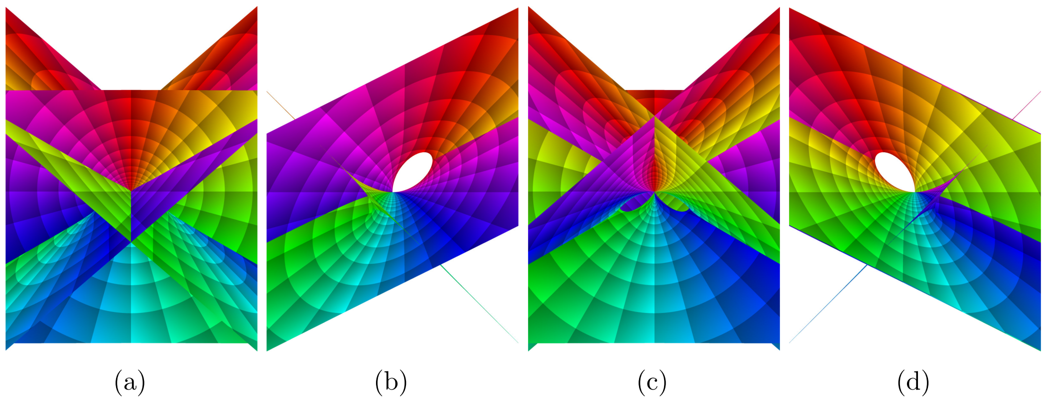


Figure 6.4.8: Domain-coloured Riemann surface of the folium of Descartes in orthogonal projection from left, front, right, and back (from left to right)

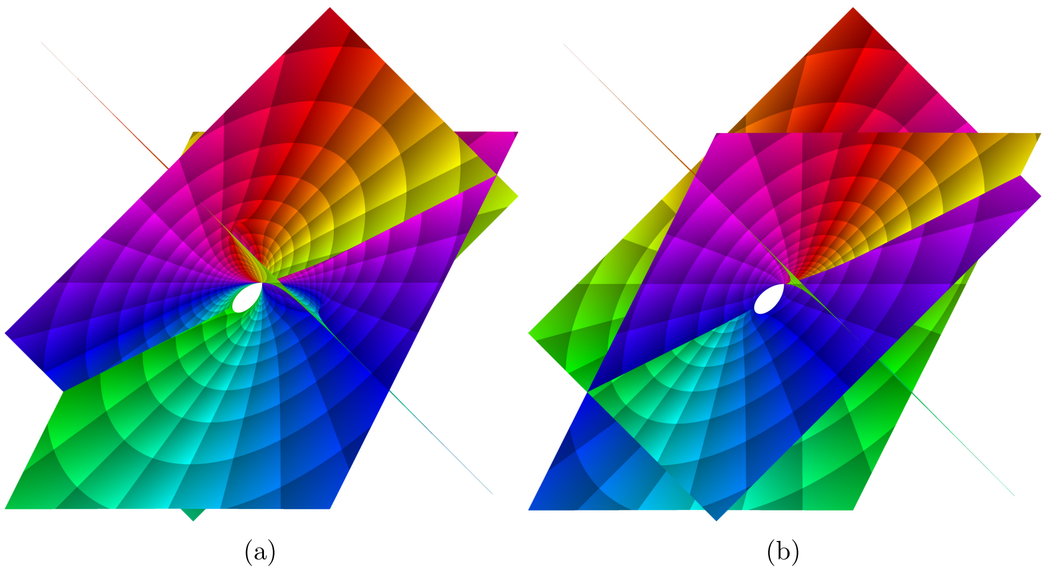


Figure 6.4.9: Leaf-shaped loops of the folium of Descartes in complex directions.

7 A Survey of Complex Loci

7 A Survey of Complex Loci

This chapter gives a survey of some plane algebraic curves from the *Famous Curves Index* (O'Connor and Robertson, 2006) and the *Encyclopédie des formes mathématiques remarquables* (Ferréol and Mandonnet, 2005). We examine how these curves can be constructed as loci. Moreover, we visualize the loci as domain-coloured Riemann surfaces using the algorithms from Chapter 6.

Since the chapter is intended as a handy reference, it contains not only new material, but also some constructions and figures from other parts of the thesis.

7.1 Quadrics

Quadrics and conics are synonymous. They comprise circles, ellipses, hyperbolas, and parabolas. A conic is uniquely defined by five points in general position. We obtain a conic through five points in general position by the following construction.

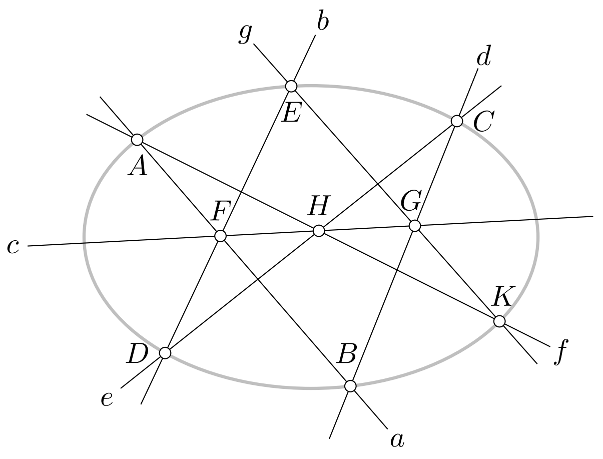


Figure 7.1.1: An instance of Construction 7.1.2. When line c rotates about point F, point K traces the conic through points A, B, C, D, E.

Construction 7.1.2. Let A, B, C, D, E be five points of the real projective plane, in general position.

1. Let a be the line through A and B.
2. Let b be the line through D and E.
3. Let F be the intersection of a and b.
4. Let c be a line through F.
5. Let d be the line through B and C.
6. Let G be the intersection of c and d.
7. Let e be the line through C and D.
8. Let H be the intersection of c and e.
9. Let f be the line through A and H.
10. Let g be the line through E and G.
11. Let K be the intersection of f and g.

By Pascal's theorem (Theorem 4.6.1), when line c rotates about point F, point K traces the conic through points A, B, C, D, E.

7.1.1 Circle

The plane algebraic curve

$$\mathcal{C}\colon f(x,y) = (x-a)^2 + (y-b)^2 - r^2 = 0$$

describes a circle with centre $(a,b)^\top$ and radius r.

There are various possible ways to define a circle. Each yields a different construction of a circle as a locus.

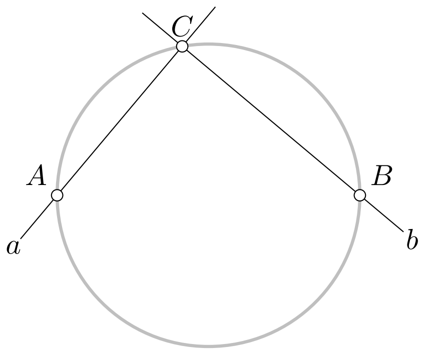


Figure 7.1.3: An instance of Construction 7.1.4. By Thales' theorem, when line a rotates around point A, point C traces the circle with diameter AB.

Construction 7.1.4 (circle by diameter)**.** Let A and B be two points in the plane.

1. Let a be a line through A.
2. Let b be the line through B perpendicular to a.
3. Let C be the intersection of a and b.

By Thales' theorem, when a rotates around A, then C traces the circle with segment AB as diameter.

Construction 7.1.6 (circle with given centre through given point)**.** Let A and B be two points in the plane.

1. Let a be a line through A.
2. Let b be the line through A and B.
3. Let c be an angle bisector of a and b.
4. Let d be the line through B perpendicular to c.
5. Let C be the intersection of a and d.

When a rotates around A, then C traces the circle through B centred at A.

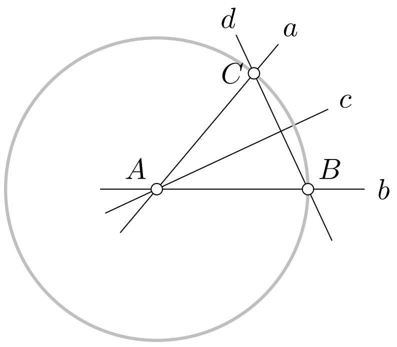


Figure 7.1.5: An instance of Construction 7.1.6. When line a rotates around point A, point C traces the circle through B centred at A.

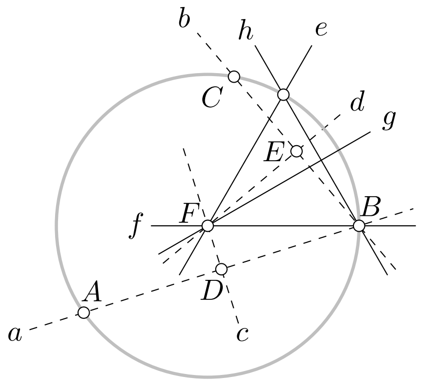


Figure 7.1.7: An instance of Construction 7.1.8. When e rotates around F, then G traces the circle through A, B, and C.

Construction 7.1.8 (circle through three points)**.** Let A, B, and C be three points in the real projective plane, in general position.

1. Let a be the line through A and B.
2. Let b be the line through B and C.
3. Let D be the midpoint of the segment between A and B
4. Let E be the midpoint of the segment between B and C
5. Let c be the line through D perpendicular to a.
6. Let d be the line through E perpendicular to b.
7. Let E be the intersection of c and d.

Point F is the circumcentre of triangle $\triangle ABC$. We proceed analogously to Construction 7.1.6 to construct the circle through B centred at F.

8. Let e be a line through F.
9. Let f be the line through F and B.
10. Let g be an angle bisector of e and f.
11. Let h be the line through B perpendicular to g.
12. Let G be the intersection of e and h.

When e rotates around F, then G traces the circle through A, B, and C.

Consider the unit circle, i.e. the circle centred at the origin with radius 1,

$$\mathcal{C}\colon\, f(x,y) = x^2 + y^2 - 1 = 0.$$

We visualize the Riemann surface of its y-coordinate as a domain-coloured function graph of the multivalued function $y(x)$ over a square of side length 10 centred at the origin.

Figure 7.1.9 shows the domain colouring of two sheets of the Riemann surface of the unit circle over a square of side length 10 centred at the origin. On the sheet shown in Figure 7.1.9a, the y-coordinate takes only non-positive values. On the sheet shown in Figure 7.1.9b, the y-coordinate takes only non-negative values. On both sheets, the branch points at $x = -1$ and $x = 1$ are clearly visible. From -1, a branch cut runs along the negative real axis. From 1, a branch cut runs along the positive real axis. Note how the domain colouring extends smoothly from one side of the branch cut on one sheet to the other side of the branch cut on the other sheet.

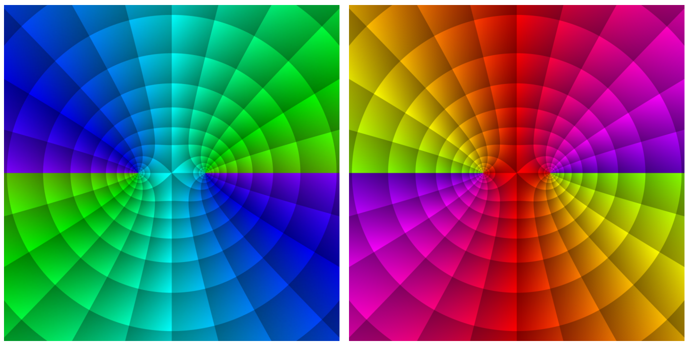

(a) (b)

Figure 7.1.9: Domain colouring of two sheets of the unit circle

Figure 7.1.10 and Figure 7.1.11 show the corresponding domain-coloured Riemann surface, using the real part of y as a height function. Note that the Riemann surface does not actually intersect itself; this is an artefact of the projection from (real) four-dimensional space into (real) three-dimensional space that results from using the real part of y as a height function.

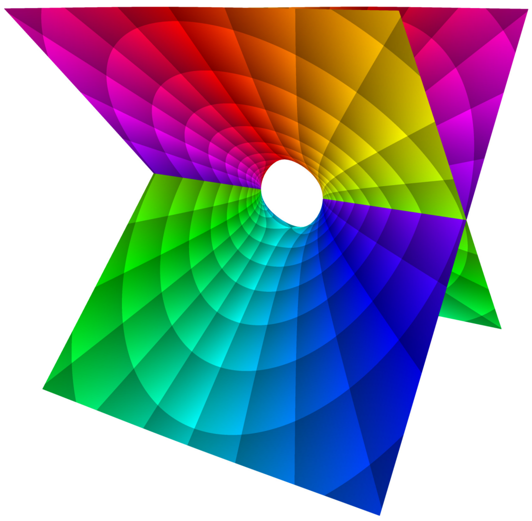


Figure 7.1.10: Domain-coloured Riemann surface of the unit circle in perspective projection

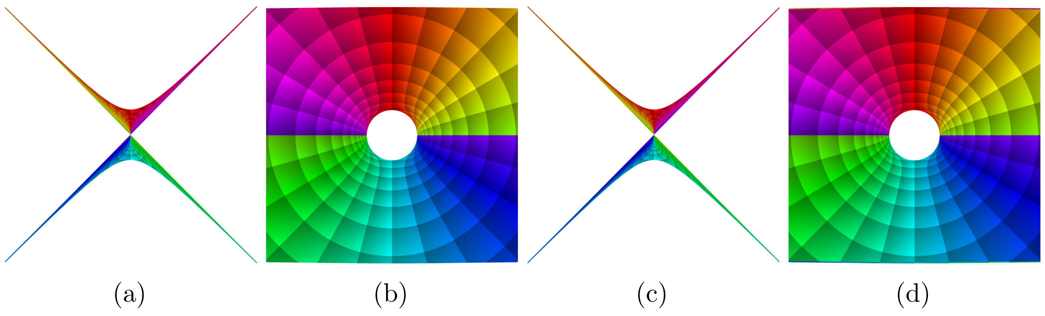


Figure 7.1.11: Domain-coloured Riemann surface of the unit circle in orthogonal projection from left, front, right, and back (from left to right)

7.1.2 Ellipse

The plane algebraic curve

$$\mathcal{C}\colon f(x,y) = (x/a)^2 + (y/b)^2 - 1 = 0$$

is an ellipse with semi-axes of length a and length b.

The following construction imitates how a gardener draws an ellipse, using the pen to pull tight a piece of string attached to two pegs at the foci of the ellipse. The construction relies on the fact that the sum of the distances between any point of the ellipse and its foci is constant.

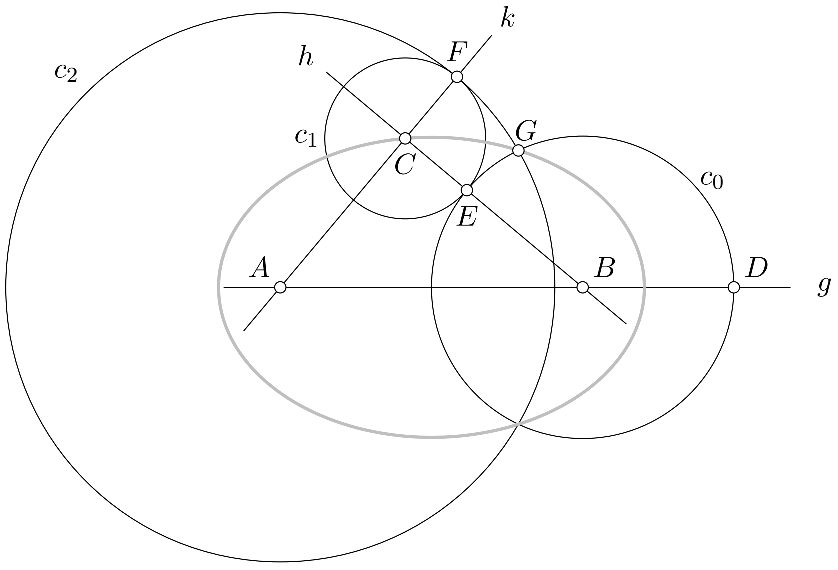


Figure 7.1.12: An instance of Construction 7.1.13. When point D moves back and forth on line g, point G traces the ellipse through C with foci A and B.

Construction 7.1.13. Let A and B be points at the foci of the ellipse. Let C be a point on the ellipse.

1. Let g be the line through A and B.
2. Let D be a point on g not further from B than C.
3. Let c_0 be the circle through D centred at B.
4. Let h be the line through B and C.
5. Let E be the intersection of c_0 and h on the same side of B as C.
6. Let c_1 be the circle through E centred at C.

7. Let k be the line through C and A.
8. Let F be the intersection of c_1 and k furthest from A.
9. Let c_2 be the circle through F centred at A.
10. Let G be an intersection of c_0 and c_2.

When D moves back and forth on g, then G traces the ellipse through C with foci A and B.

Figure 7.1.14 shows the ellipse with parameters $a = 2$, $b = 1$ as a real plane algebraic curve. Figure 7.1.15 shows the domain colouring of two sheets of its Riemann surface over a square of side length 10 centred at the origin. Figure 7.1.16 and Figure 7.1.17 show the corresponding domain-coloured Riemann surface, using the real part of y as a height function.

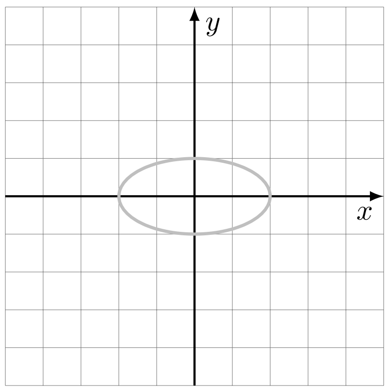


Figure 7.1.14: The ellipse with parameters $a = 2$, $b = 1$ as a real plane algebraic curve

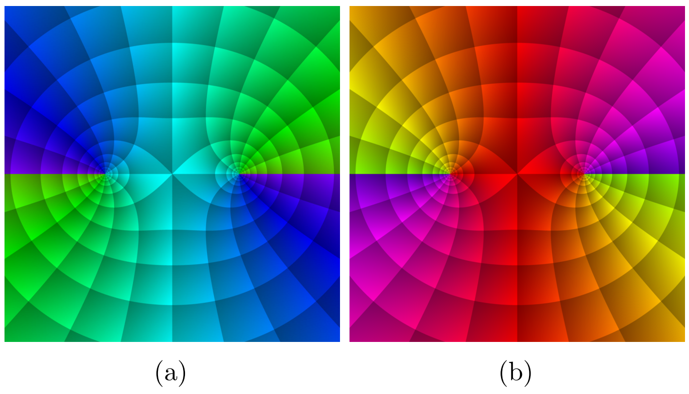


Figure 7.1.15: Domain colouring of two sheets of the ellipse with parameters $a = 2$, $b = 1$

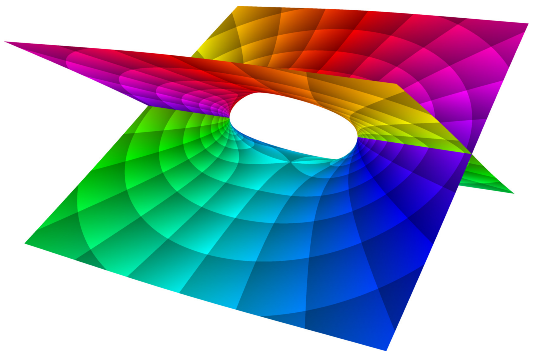


Figure 7.1.16: Domain-coloured Riemann surface of the ellipse with parameters $a = 2$, $b = 1$ in perspective projection

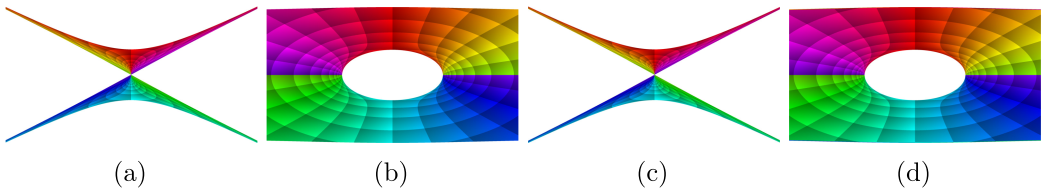


Figure 7.1.17: Domain-coloured Riemann surface of the ellipse with parameters $a = 2$, $b = 1$ in orthogonal projection from left, front, right, and back (from left to right)

7.1.3 Hyperbola

The plane algebraic curve

$$\mathcal{C}\colon f(x,y) = (x/a)^2 - (y/b)^2 - 1 = 0.$$

is a hyperbola with parameters a, b.

The following construction of a hyperbola relies on the fact that the difference of the distances between any point of the hyperbola and its foci is constant.

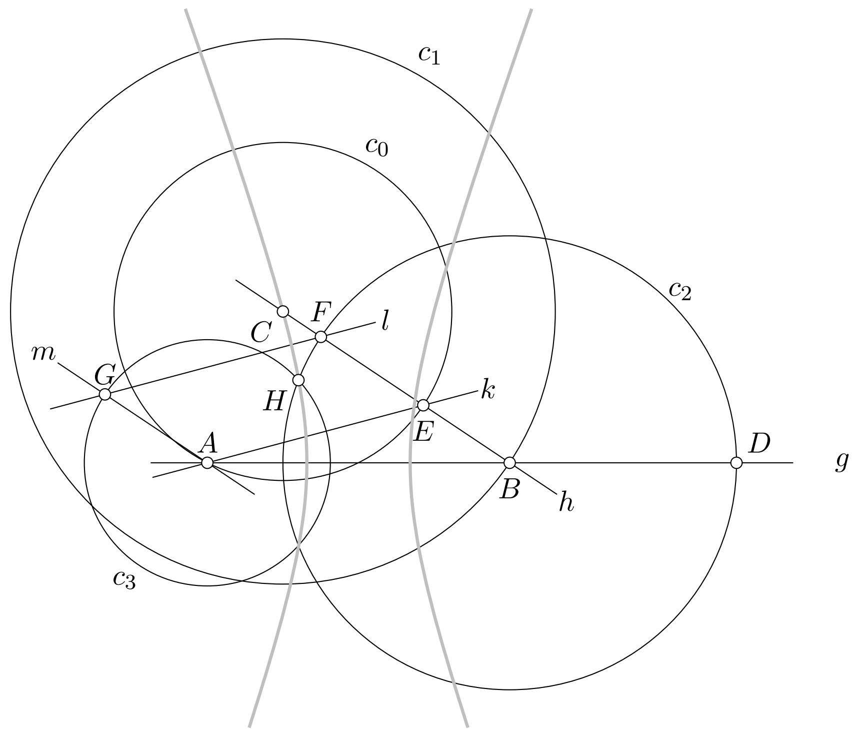


Figure 7.1.18: An instance of Construction 7.1.19. When point D moves back and forth on line g, point H traces the hyperbola through C with foci A and B.

Construction 7.1.19. Let A and B be points at the foci of the hyperbola. Let C be a point on the hyperbola.

1. Let g be the line through A and B.
2. Let h be the line through B and C.
3. Let c_0 be the circle through A centred at C.
4. Let c_1 be the circle through B centred at C.
5. Let D be a point on g.

6. Let c_2 be the circle through D centred at B.
7. Let E be the intersection of c_0 and h on the same side of C as B.
8. Let F be the intersection of c_2 and h on the same side of B as E.
9. Let k be the line through A and E.
10. Let l be the line through F parallel to k.
11. Let m be the line through A parallel to h.
12. Let G be the intersection of l and m.
13. Let c_3 be the circle through G centred at A.
14. Let H be an intersection of c_2 and c_3.

When D moves back and forth on g, H traces the hyperbola through C with foci A and B.

Figure 7.1.20 shows the hyperbola with parameters $a = b = 1$ as a real plane algebraic curve. Figure 7.1.21 shows the domain colouring of two sheets of its Riemann surface over a square of side length 10 centred at the origin. Figure 7.1.22 and Figure 7.1.23 show the corresponding domain-coloured Riemann surface, using the real part of y as a height function.

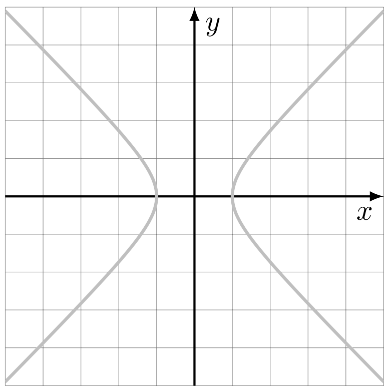


Figure 7.1.20: The hyperbola with parameters $a = b = 1$ as a real plane algebraic curve

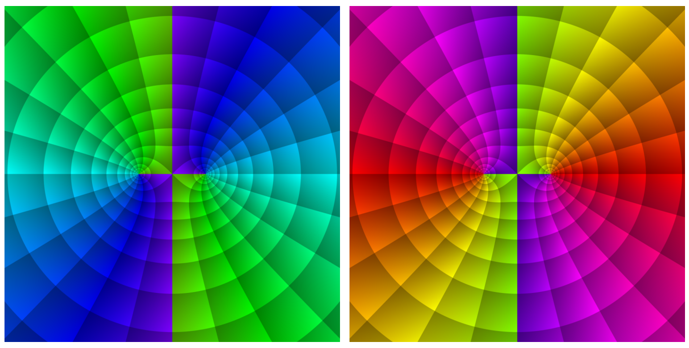

(a) (b)

Figure 7.1.21: Domain colouring of two sheets of the hyperbola with parameters $a = b = 1$

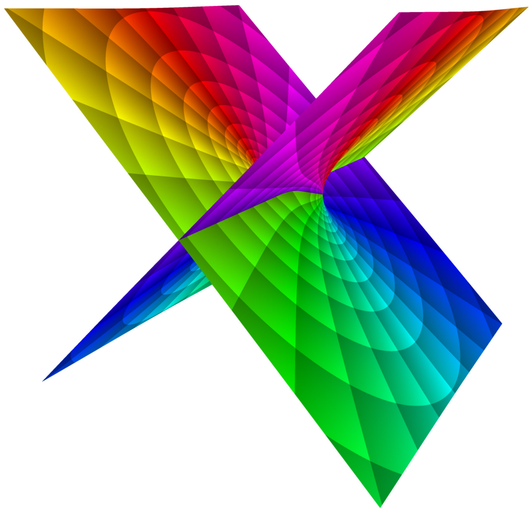


Figure 7.1.22: Domain-coloured Riemann surface of the hyperbola with parameters $a = b = 1$ in perspective projection

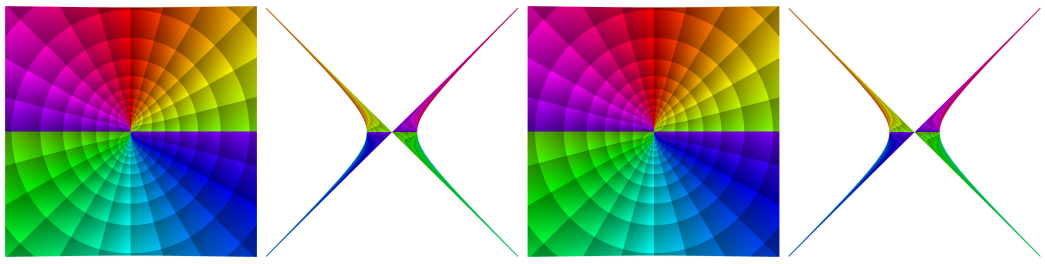

(a) (b) (c) (d)

Figure 7.1.23: Domain-coloured Riemann surface of the hyperbola with parameters $a = b = 1$ in orthogonal projection from left, front, right, and back (from left to right)

7.1.4 Parabola

The plane algebraic curve

$$\mathcal{C}\colon f(x,y) = ay^2 + by + c - x = 0$$

is a parabola with parameters a, b, c.

The following construction of a parabola relies on the fact that any point of the parabola is equidistant to a fixed point, the focus of the parabola, and to a fixed line, the directrix of the parabola.

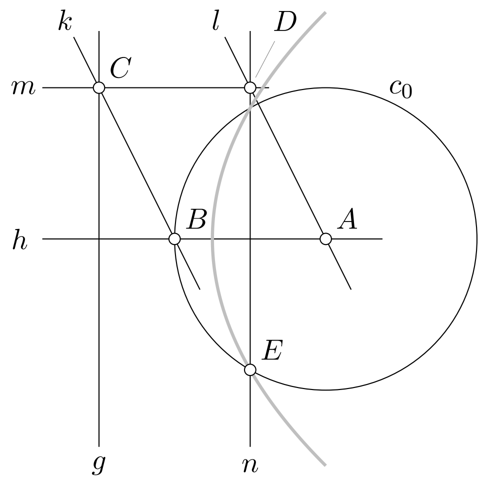


Figure 7.1.24: An instance of Construction 7.1.25. When point B moves back and forth on line h, point E traces the parabola with focus A and directrix g.

Construction 7.1.25. Let a point A and a line g be given.

1. Let h be the line through A perpendicular to g.
2. Let B be a point on h.
3. Let c_0 be the circle through B centred at A.
4. Let C be a point on g and not on h.
5. Let k be the line through C and B.
6. Let l be the line through A parallel to k.
7. Let m be the line through C parallel to h.
8. Let D be the intersection of k and h.
9. Let n be the line through D parallel to g.

10. Let E be an intersection of c_0 and n.

When C moves back and forth on h, E traces the parabola with focus A and directrix g.

In what follows we visualize the parabola with parameters $a = 1$, $b = c = 0$,

$$\mathcal{C}\colon\ f(x,y) = y^2 - x = 0.$$

Note that in this case, $y(x)$ is the (multi-valued) complex square root function.

Figure 7.1.26 shows the parabola with parameters $a = 1$, $b = c = 0$ as a real plane algebraic curve. Figure 7.1.27 shows the domain colouring of two sheets of its Riemann surface over a square of side length 10 centred at the origin. Figure 7.1.28 and Figure 7.1.29 show the corresponding domain-coloured Riemann surface, using the real part of y as a height function. See Section 6.4.2 for an interpretation of the visualization in terms of the complex square root function.

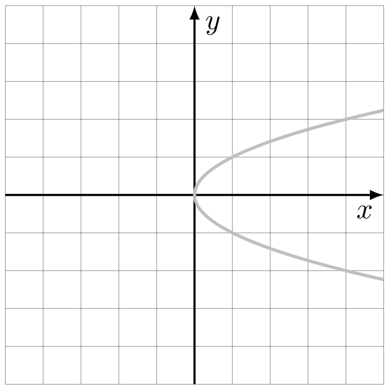


Figure 7.1.26: The parabola with parameters $a = 1$, $b = c = 0$ as a real plane algebraic curve

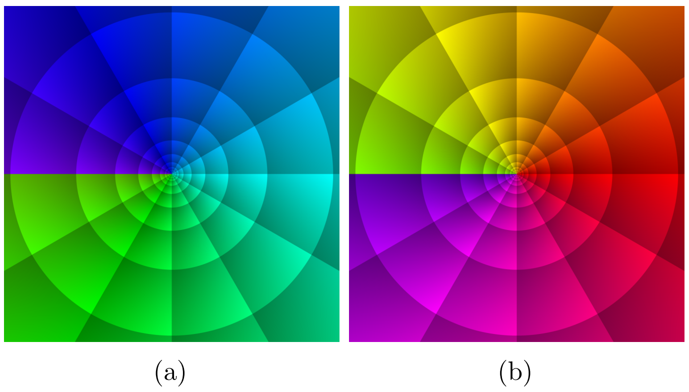


Figure 7.1.27: Domain colouring of two sheets of the parabola with parameters $a = 1$, $b = c = 0$

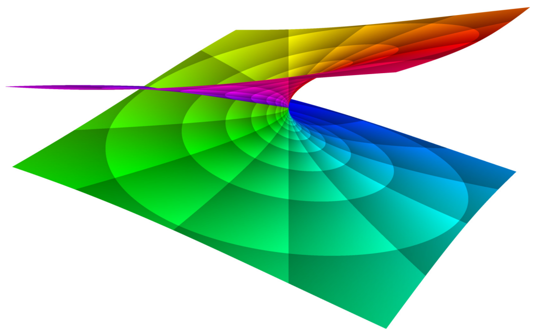


Figure 7.1.28: Domain-coloured Riemann surface of the parabola with parameters $a = 1$, $b = c = 0$ in perspective projection

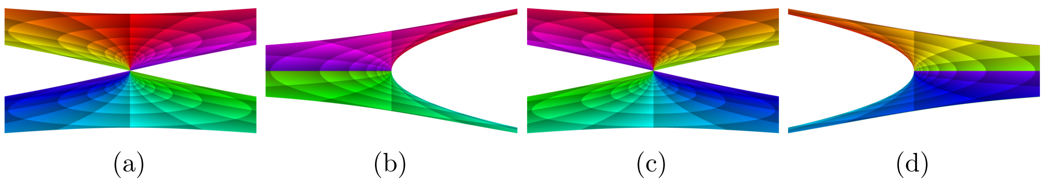


Figure 7.1.29: Domain-coloured Riemann surface of the parabola with parameters $a = 1$, $b = c = 0$ in orthogonal projection from left, front, right, and back (from left to right)

7.2 Cubics

7.2.1 Folium of Descartes

The folium of Descartes is a cubic plane algebraic curve

$$\mathcal{C}\colon f(x,y) = x^3 + y^3 - 3axy = 0$$

with parameter $a > 0$.

The following construction of the folium of Descartes is given by de Longchamps (1890, p. 104f.).

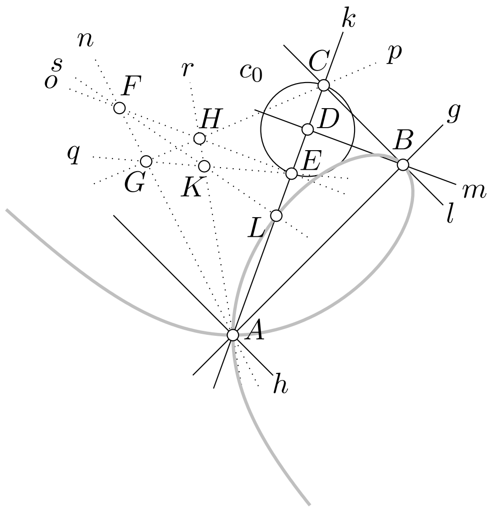


Figure 7.2.1: An instance of Construction 7.2.2. When line k rotates around point A, point L traces the folium of Descartes with apex $B = 3/2 \cdot (a,a)^\top$.

Construction 7.2.2. Let g be the angle bisector of the first and third quadrant. Let h be the angle bisector of the second and fourth quadrant. Let A be a point at the origin. Let B be a point on g.

1. Let k be a line through A.
2. Let l be the line through B parallel to h.
3. Let C be the intersection of k and l.
4. Let m be the line through B perpendicular to k.
5. Let D be the intersection of k and m.
6. Let c_0 be the circle through C centred at D.
7. Let E be the intersection of c_0 with k other than C.

We construct the harmonic conjugate L of C w.r.t. A and E:

8. Let F be a point not on k.
9. Let n be the line through A and F.
10. Let o be the line through E and F.
11. Let p be a line through C not through F.
12. Let G be the intersection of n and p.
13. Let H be the intersection of o and p.
14. Let q be the line through E and G.
15. Let r be the line through A and H.
16. Let K be the intersection of q and r.
17. Let s be the line through F and K.
18. Let L be the intersection of k and s.

When k rotates around A, L traces the folium of Descartes with apex $B = 3/2 \cdot (a, a)^\top$.

Figure 7.2.3 shows the folium of Descartes with parameter $a = 1$ as a real plane algebraic curve. Figure 7.2.4 shows the domain colouring of three sheets of its Riemann surface over a square of side length 10 centred at the origin. Figure 7.2.5 and Figure 7.2.6 show the corresponding domain-coloured Riemann surface, using the real part of y as a height function.

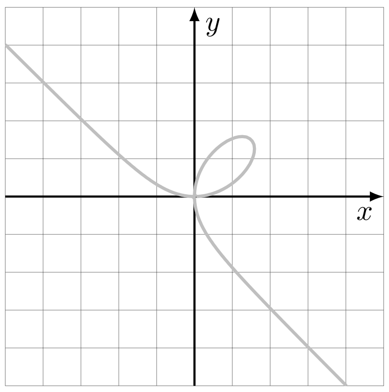


Figure 7.2.3: The folium of Descartes with parameter $a = 1$ as a real plane algebraic curve

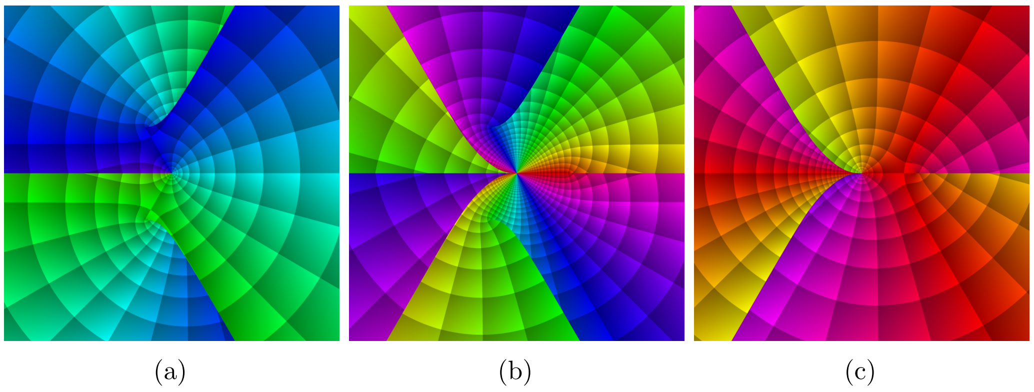


Figure 7.2.4: Domain colouring of three sheets of the folium of Descartes with parameter $a = 1$

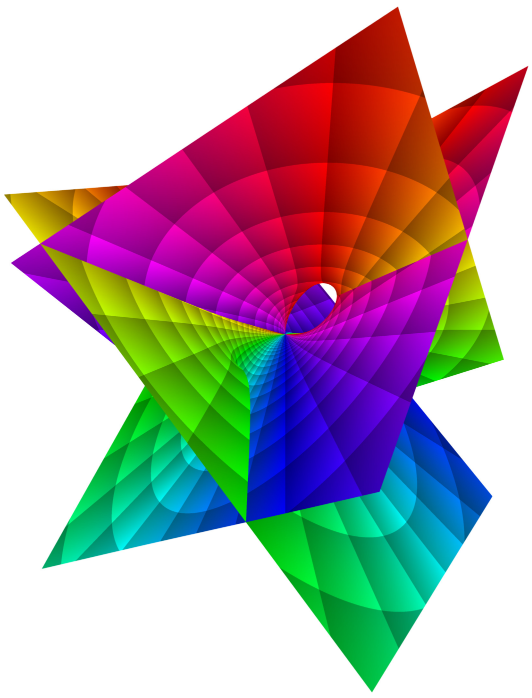


Figure 7.2.5: Domain-coloured Riemann surface of the folium of Descartes with parameter $a = 1$ in perspective projection

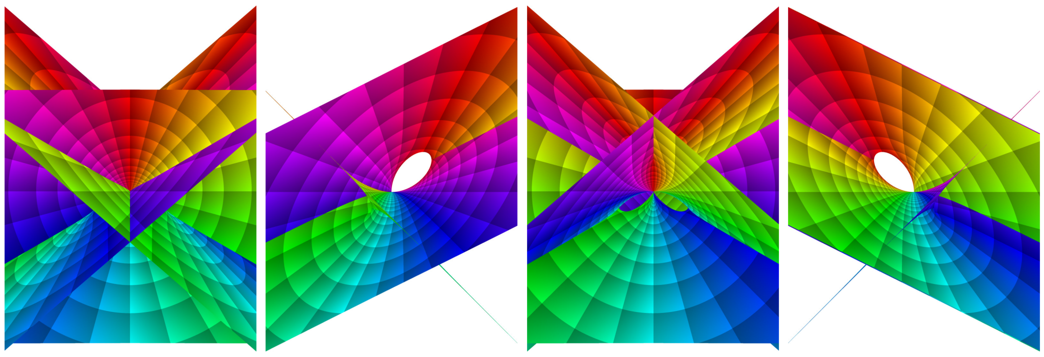


Figure 7.2.6: Domain-coloured Riemann surface of the folium of Descartes with parameter $a = 1$ in orthogonal projection from left, front, right, and back (from left to right)

7.2.2 Right Strophoid

The right strophoid is a cubic plane algebraic curve

$$\mathcal{C}\colon f(x,y) = (a+x)y^2 - (a-x)x^2 = 0$$

with parameter $a > 0$.

The following construction of the right strophoid is given in (Ferréol and Mandonnet, 2005, Strophoïde droite, `http://www.mathcurve.com/courbes2d/strophoid droite/strophoiddroite.shtml`). It is a generalization of a construction of the bicorn in (de Longchamps, 1897, p. 39f.).

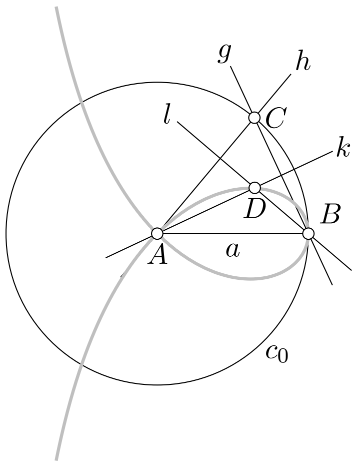


Figure 7.2.7: An instance of Construction 7.2.8. When point C moves along circle c_0, point D traces the right strophoid with parameter a.

Construction 7.2.8. Let A be a point at the origin. Let B be the point with coordinates $(a,0)^\top$.

1. Let c_0 be the circle through B centred at A.
2. Let C be a point on c_0.
3. Let g be the line through B and C.
4. Let h be the line through A and C.
5. Let k be the line through A perpendicular to g.
6. Let l be the line through B perpendicular to h.
7. Let F be the intersection of k and l.

When C moves along c_0, D traces the right strophoid with parameter a.

Figure 7.2.9 shows the right strophoid with parameter $a = 1$ as a real plane algebraic curve. Figure 7.2.10 shows the domain colouring of two sheets of its Riemann surface over a square of side length 10 centred at the origin. Figure 7.2.11 and Figure 7.2.12 show the corresponding domain-coloured Riemann surface, using the real part of y as a height function.

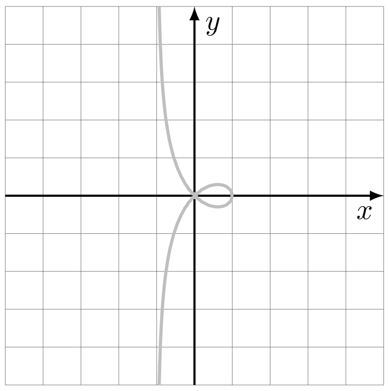


Figure 7.2.9: The right strophoid with parameter $a = 1$ as a real plane algebraic curve

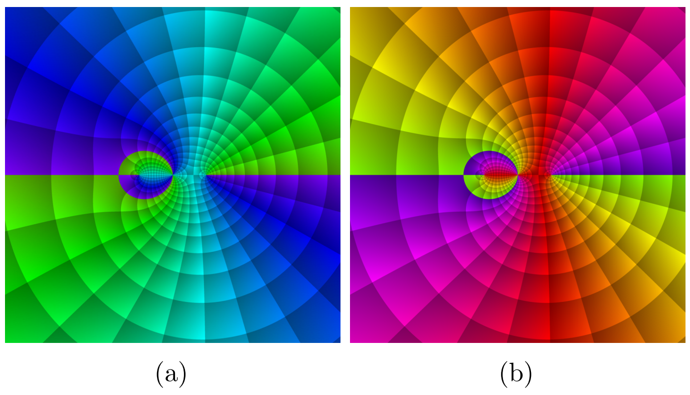


Figure 7.2.10: Domain colouring of three sheets of the right strophoid with parameter $a = 1$

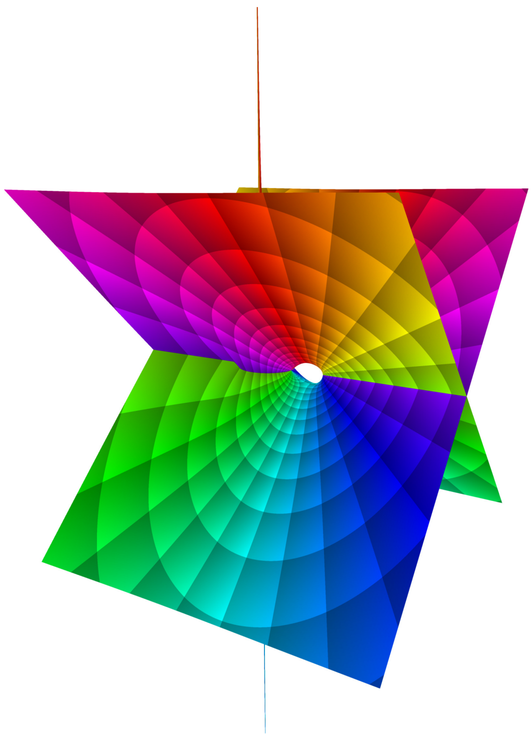


Figure 7.2.11: Domain-coloured Riemann surface of the right strophoid with parameter $a = 1$ in perspective projection

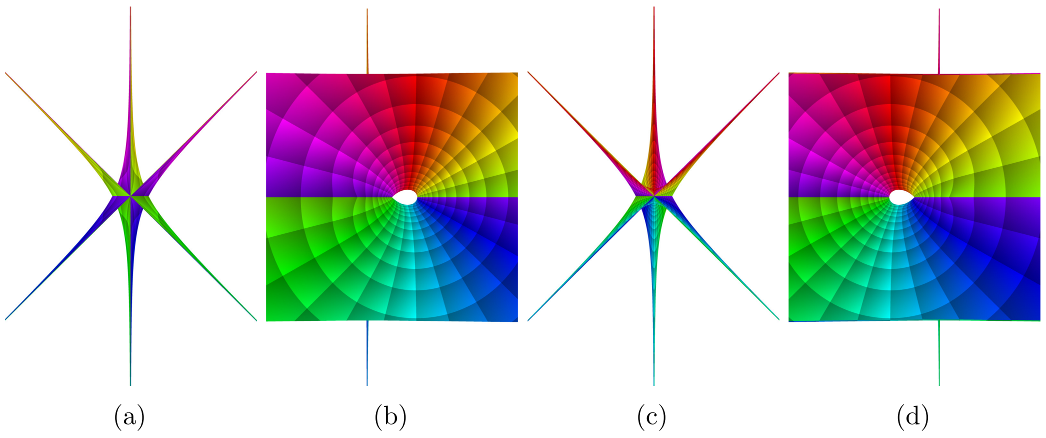


Figure 7.2.12: Domain-coloured Riemann surface of the right strophoid with parameter $a = 1$ in orthogonal projection from left, front, right, and back (from left to right)

7.3 Quartics

7.3.1 Conchoid of Nicomedes

The conchoids of Nicomedes are a family of quartic plane algebraic curves,

$$\mathcal{C}\colon f(x,y) = (y+a)^2(x^2+y^2) - b^2y^2 = 0,$$

with parameters $a, b > 0$. They can be generated as loci using the following construction.

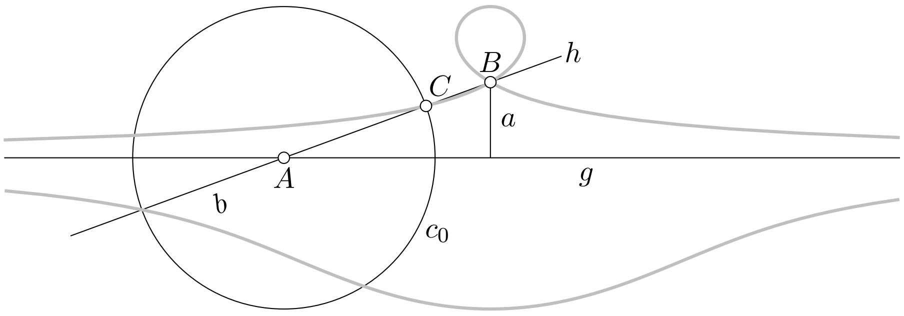


Figure 7.3.1: An instance of Construction 7.3.2. When point A moves along line g, point C traces the conchoid with *pole* B, *base* g and *distance* b.

Construction 7.3.2. Let A be a point on a line g. Let B be a point at distance $a > 0$ from g. Let c_0 be a circle of radius b centred at A.

1. Let h be the line through A and B.
2. Let C be an intersection of c_0 and h.

When point A moves along line g, point C traces the conchoid with *pole* B, *base* g, and *distance* b.

Figure 4.6.9 shows the conchoid of Nicomedes with parameters $a = 1$, $b = 2$ as a real plane algebraic curve. Figure 7.3.4 shows the domain colouring of four sheets of its Riemann surface over a square of side length 10 centred at the origin. Figure 7.3.5 and Figure 7.3.6 show the corresponding domain-coloured Riemann surface, using the real part of y as a height function.

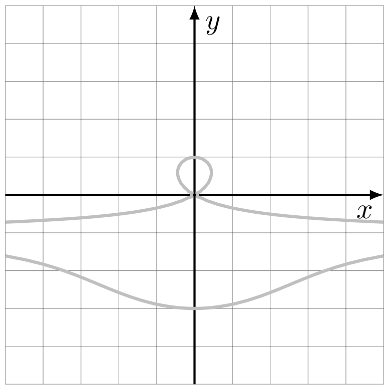


Figure 7.3.3: The conchoid of Nicomedes with parameters $a = 1$, $b = 2$ as a real plane algebraic curve

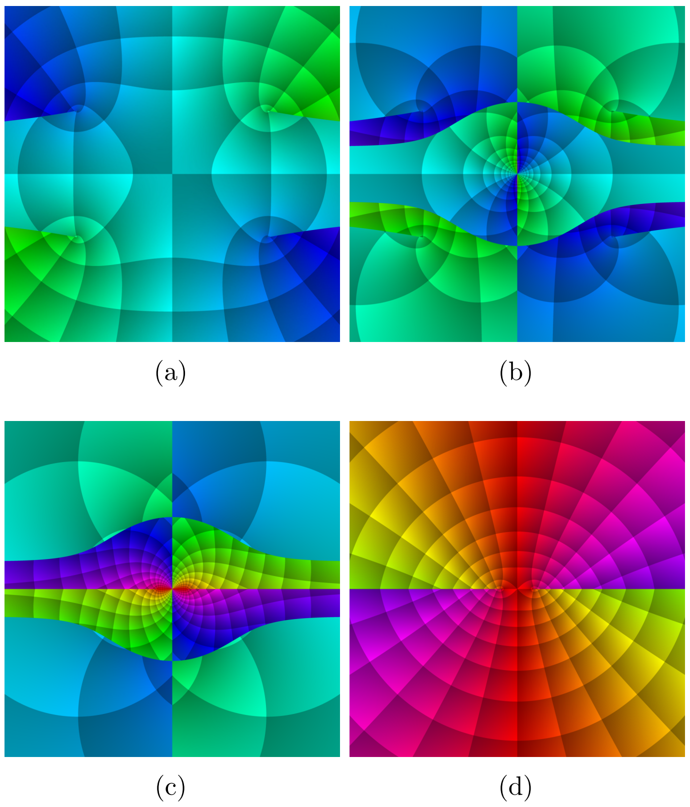


Figure 7.3.4: Domain colouring of four sheets of the conchoid of Nicomedes with parameters $a = 1$, $b = 2$

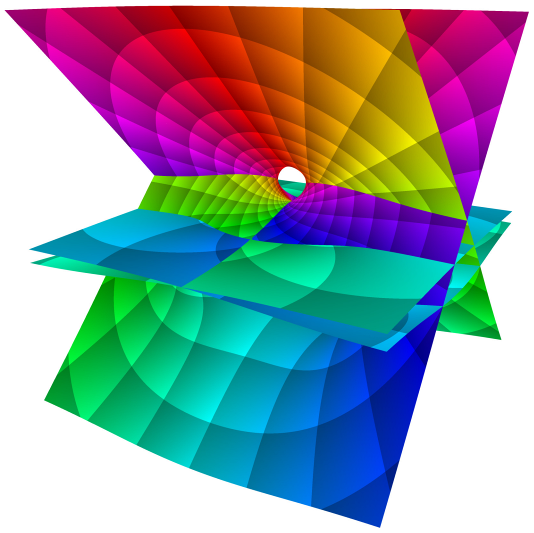


Figure 7.3.5: Domain-coloured Riemann surface of the conchoid of Nicomedes with parameters $a = 1$, $b = 2$ in perspective projection

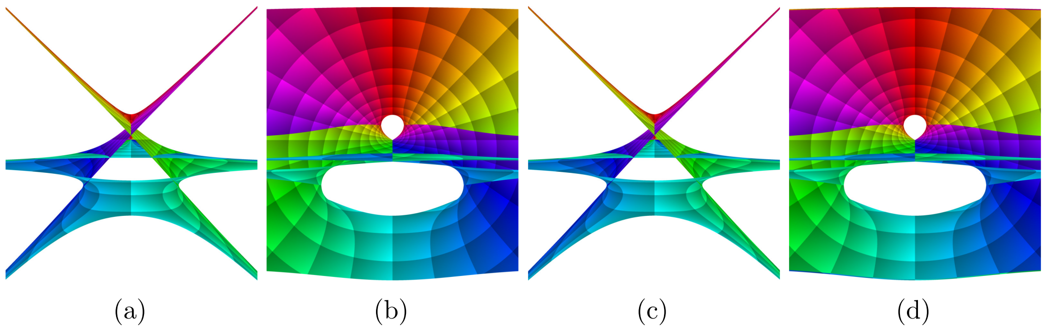


Figure 7.3.6: Domain-coloured Riemann surface of the conchoid of Nicomedes with parameters $a = 1$, $b = 2$ in orthogonal projection from left, front, right, and back (from left to right)

7.3.2 Limaçon

A limaçon is a quartic plane algebraic curve

$$\mathcal{C}\colon\ f(x,y) = (x^2+y^2-ax)^2 - b^2(x^2+y^2) = 0,$$

with parameters $a, b > 0$.

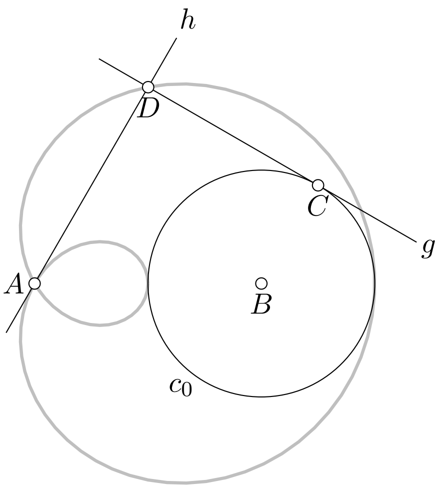


Figure 7.3.7: An instance of Construction 7.3.8

Construction 7.3.8. Let A be a point at the origin. Let B be the point with coordinates $(a,0)^\top$. Let c_0 be the circle with centre B and radius b. Let C be a point on circle c_0.

1. Let g be the tangent to c_0 through C.
2. Let h be the line through A perpendicular to g.
3. Let D be the intersection of g and h.

When point C moves around circle c_0, point D traces a limaçon with parameters a, b.

Figure 7.3.9 shows the limaçon with parameters $a = b = 1.5$ as a real plane algebraic curve. Figure 7.3.10 shows the domain colouring of four sheets of its Riemann surface over a square of side length 10 centred at the origin. Figure 7.3.11 and Figure 7.3.12 show the corresponding domain-coloured Riemann surface, using the real part of y as a height function.

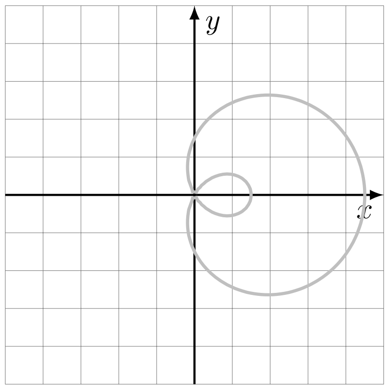


Figure 7.3.9: The limaçon with parameters $a = b = 1.5$ as a real plane algebraic curve

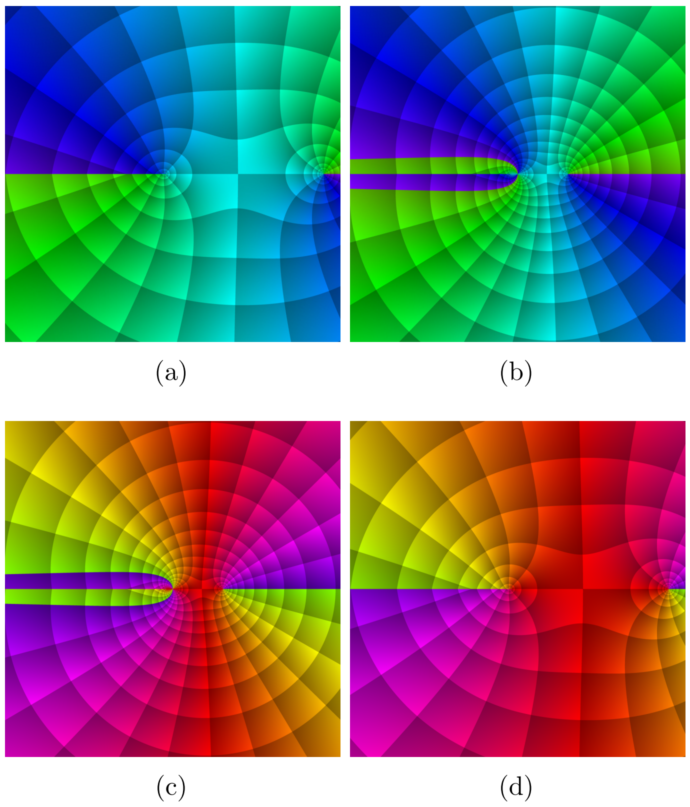


Figure 7.3.10: Domain colouring of four sheets of the limaçon with parameters $a = b = 1.5$

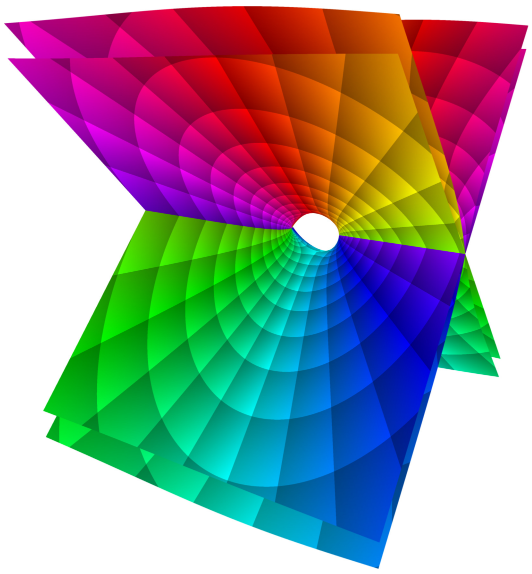


Figure 7.3.11: Domain-coloured Riemann surface of the limaçon with parameters $a = b = 1.5$ in perspective projection

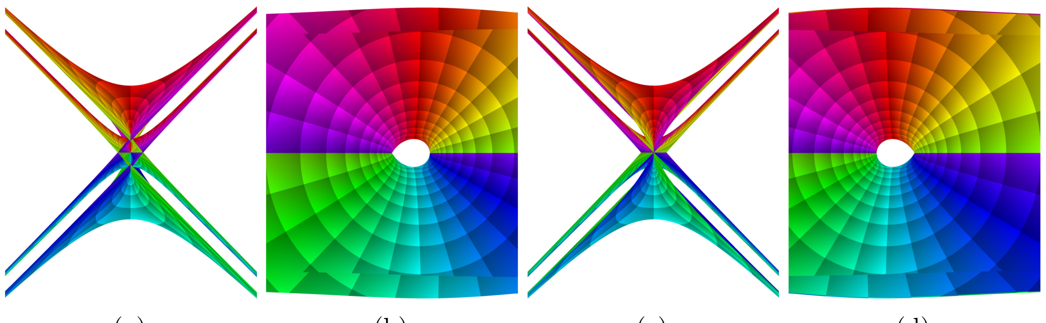


Figure 7.3.12: Domain-coloured Riemann surface of the limaçon with parameters $a = b = 1.5$ in orthogonal projection from left, front, right, and back (from left to right)

7.4 Quintics

To the best of my knowledge, there are no ‘famous’ quintic plane algebraic curves that arise as a locus in a geometric construction. Therefore, we only treat the quintic that describes the relationship between cross-ratio μ (here: x) and initial point γ_0 (here: y) of a closed discrete Darboux transform of a regular pentagon (cf. Equation 2.9 in Section 2.4).

7.4.1 Pentagon curve

Consider the quintic plane algebraic curve

$$\begin{aligned}\mathcal{C}\colon\ \Big[\Big(\Big(-3+\sqrt{5}\Big)x^2+6x-3-\sqrt{5}\Big)y^2+\Big(\Big(-2-4\sqrt{5}\Big)x+1+\sqrt{5}\Big)y\\ +\Big(-3+\sqrt{5}\Big)x^2+6x-3-\sqrt{5}\Big](1-x)=0\end{aligned}$$

(cf. Equation 2.9 in Section 2.4).

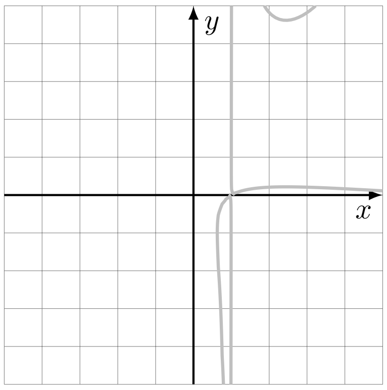


Figure 7.4.1: Pentagon curve as a real plane algebraic curve

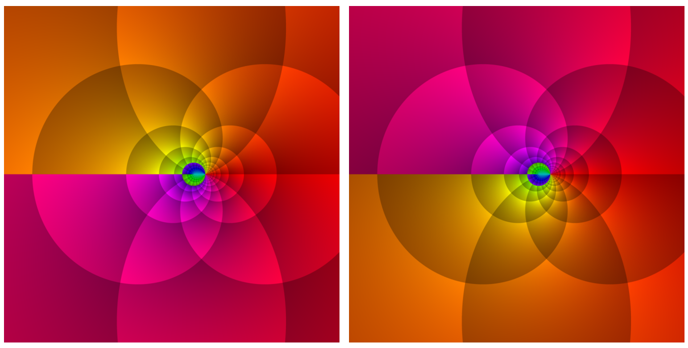


Figure 7.4.2: Domain colouring of two sheets of a Pentagon curve

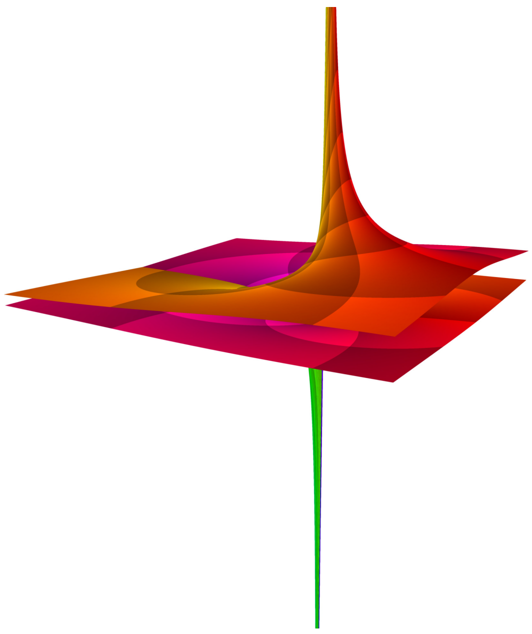


Figure 7.4.3: Domain-coloured Riemann surface of a Pentagon curve in perspective projection

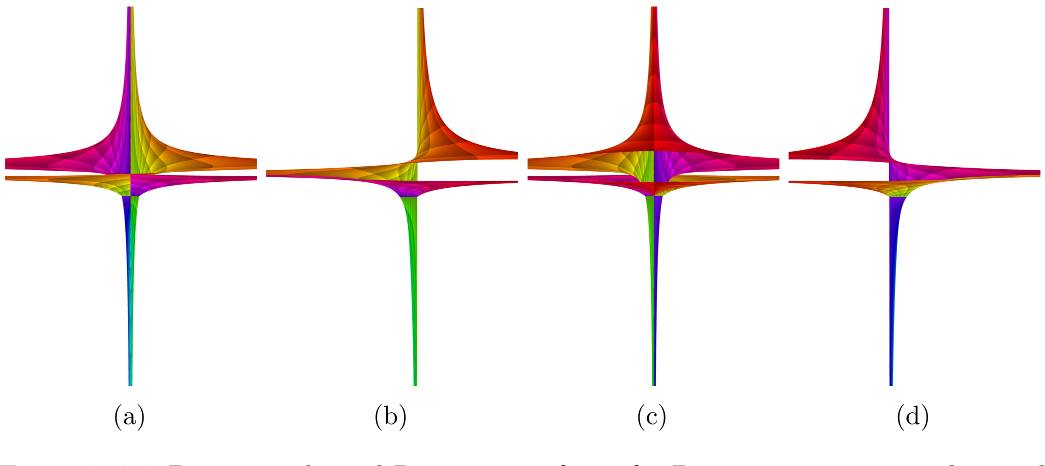


Figure 7.4.4: Domain-coloured Riemann surface of a Pentagon curve in orthogonal projection from left, front, right, and back (from left to right)

7.5 Sextics

7.5.1 Nephroid

A nephroid is a sextic plane algebraic curve

$$\mathcal{C}\colon\ f(x,y) = (x^2+y^2-4a^2)^3 - 108a^4y^2 = 0$$

with parameter $a > 0$.

A nephroid with parameter $a > 0$ is an epicycloid with two cusps. It arises as the locus of a point on a circle of radius a that rolls without slipping on a circle of radius $2a$. This yields the following construction:

Construction 7.5.1. Let A be a point at the origin. Let c_0 be the circle of radius $3a$ centred at A. Let B be a point on c_0. Let c_1 be the circle of radius a centred at B.

1. Let g be the line through A and B.
2. Let C be an intersection of g and c_1.
3. Let h be the line through B parallel to the y-axis.
4. Let D be an intersection of h and c_1.
5. Let c_2 be the circle through D centred at C.
6. Let E be the other intersection of c_1 and c_2.
7. Let c_3 be the circle through C centred at E.
8. Let F be the intersection of c_2 and c_3 other than C.

When point B moves around circle c_0, then point F traces the nephroid with parameter a.

Figure 7.5.3 shows the nephroid with parameter $a = 1$ as a real plane algebraic curve. Figure 7.5.4 shows the domain colouring of six sheets of its Riemann surface over a square of side length 10 centred at the origin. Figure 7.5.5 and Figure 7.5.6 show the corresponding domain-coloured Riemann surface, using the real part of y as a height function.

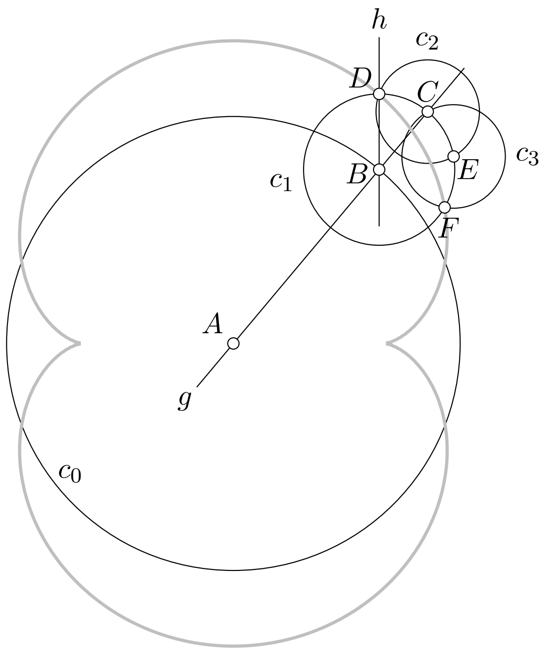


Figure 7.5.2: An instance of Construction 7.5.1

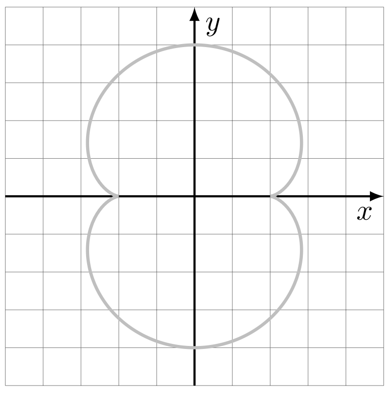
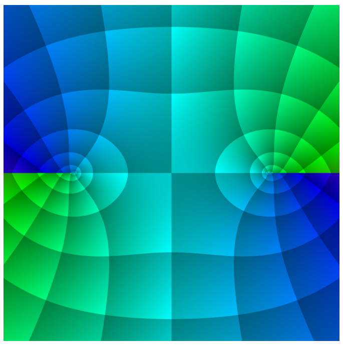
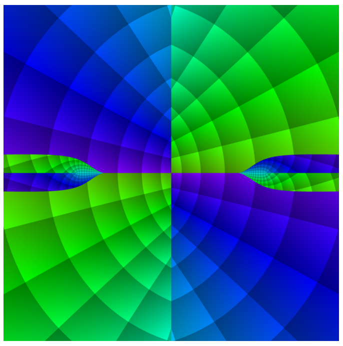
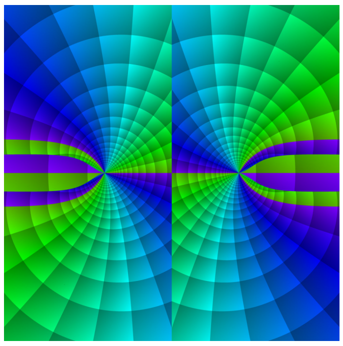
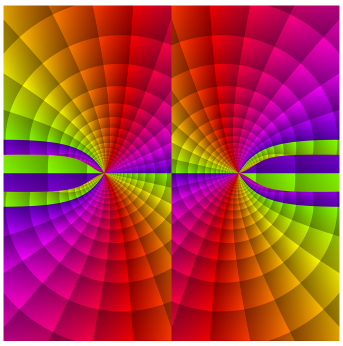


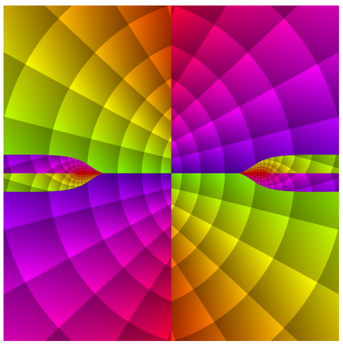
Figure 7.5.3: The nephroid with parameter $a = 1$ as a real plane algebraic curve


(a)


(b)


(c)


(d)


(e)

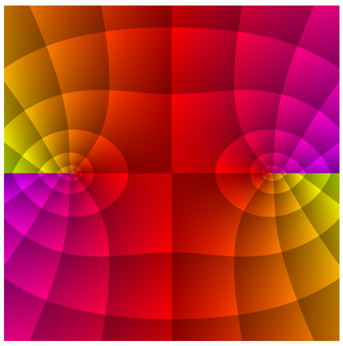

(f)

Figure 7.5.4: Domain colouring of six sheets of the nephroid with parameter $a = 1$

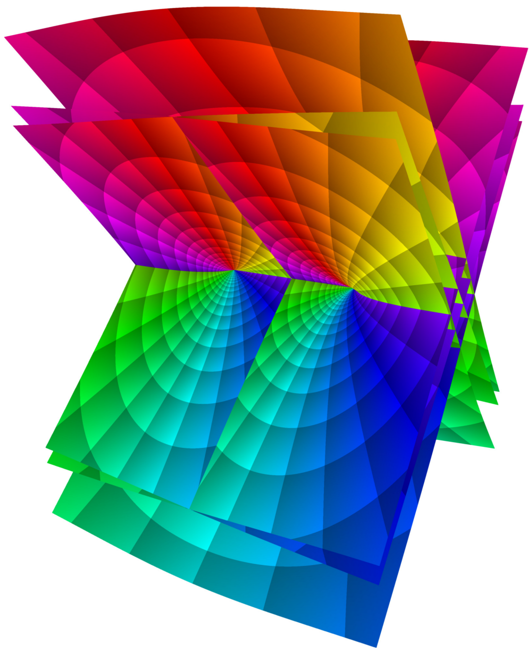


Figure 7.5.5: Domain-coloured Riemann surface of the nephroid with parameter $a = 1$ in perspective projection

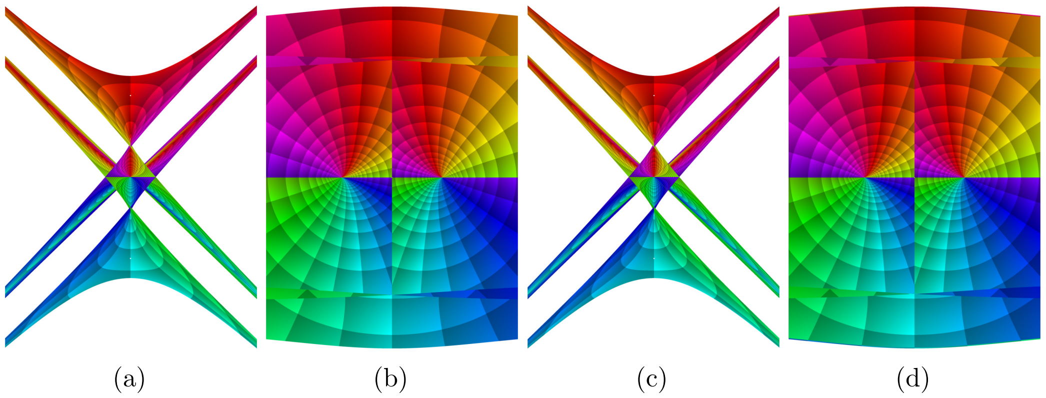


Figure 7.5.6: Domain-coloured Riemann surface of the nephroid with parameter $a = 1$ in orthogonal projection from left, front, right, and back (from left to right)

7.5.2 Watt curves

Watt curves are a family of sextic plane algebraic curves

$$\mathcal{C}\colon f(x,y) = (x^2+y^2)(x^2+y^2-a^2-b^2+c^2)^2 + 4a^2y^2(x^2+y^2-b^2) = 0$$

with parameters $a, b, c > 0$. They can be generated as loci using a four-bar linkage according to the following construction.

Construction 7.5.7. Let A and B be two points in the plane at distance $2a$ from each other. Let c_0 be a circle with centre A and radius b. Let c_1 be a circle with centre B and radius b.

1. Let C be a point on c_0.
2. Let c_2 be a circle with centre C and radius $2c$.
3. Let D be an intersection of c_1 and c_2.
4. Let E be the midpoint of the segment between C and D.

When point C moves on circle c_0, point E traces a Watt curve with parameters a, b, c.

Depending on parameters a, b, c, Watt curves possess a wide variety of different forms. In what follows, we consider two different sets of parameter values.

Watt curve with parameters $a = 2$, $b = 2.5$, $c = 1.5$

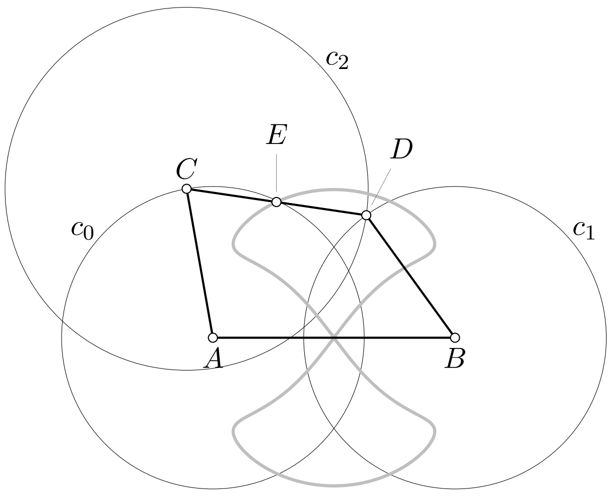
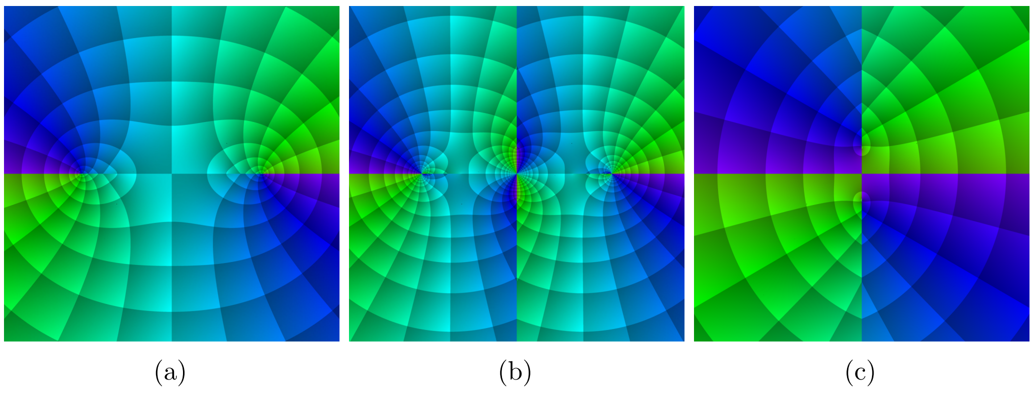


Figure 7.5.8: An instance of Construction 7.5.7. When point C moves on circle c_0, then point E traces a Watt curve with parameters $a = |AB|/2 = 2$, $b = |AC| = |BD| = 2.5$, and $c = |CD|/2 = 1.5$.



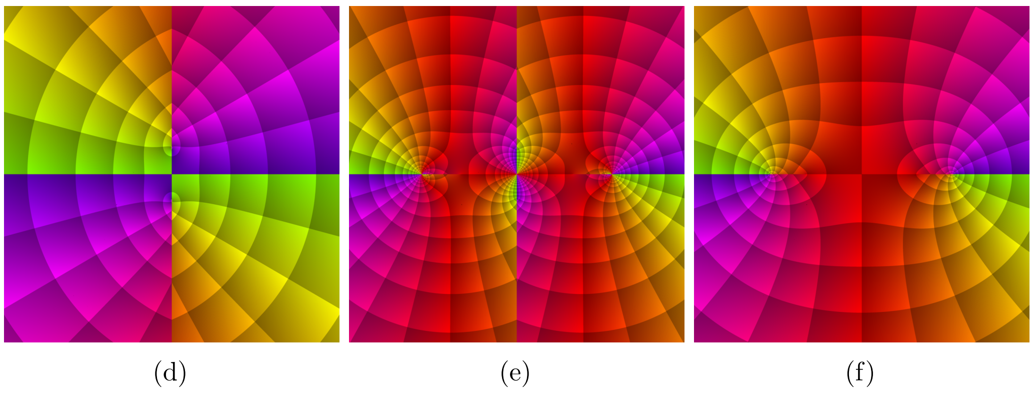


Figure 7.5.9: Domain colouring of six sheets of a Watt curve with parameters $a = 2$, $b = 2.5$, $c = 1.5$

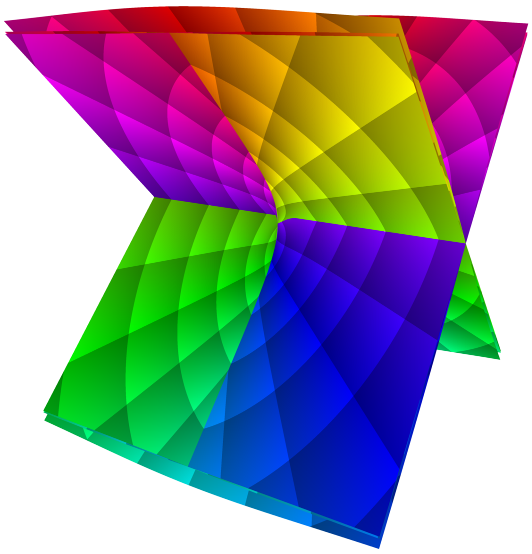


Figure 7.5.10: Domain-coloured Riemann surface of a Watt curve with parameters $a = 2$, $b = 2.5$, $c = 1.5$ in perspective projection

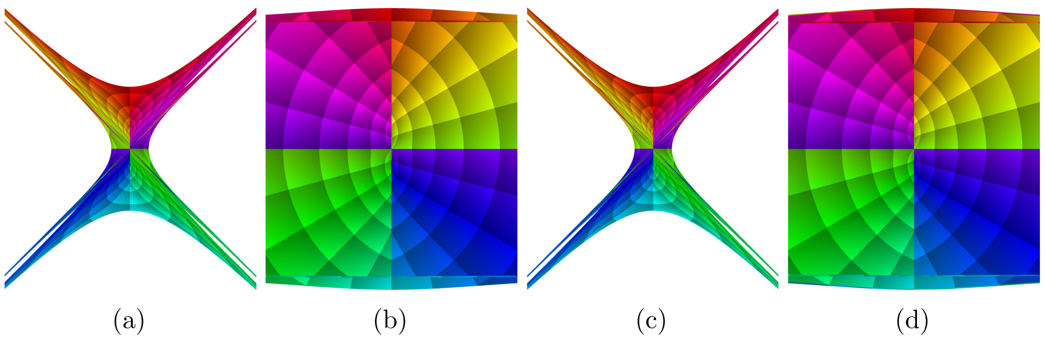


Figure 7.5.11: Domain-coloured Riemann surface of a Watt curve with parameters $a = 2$, $b = 2.5$, $c = 1.5$ in orthogonal projection from left, front, right, and back (from left to right)

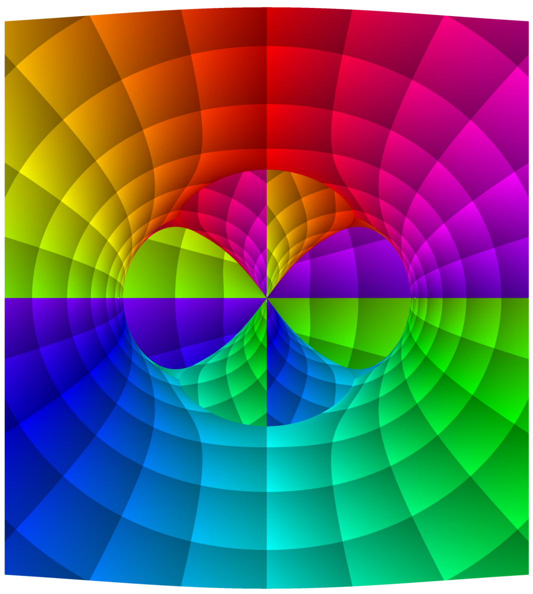


Figure 7.5.12: Domain-coloured Riemann surface of the Watt curve with parameters $a = 2$, $b = 2.5$, $c = 1.5$ in orthogonal projection from front, cut off at the real plane. The Watt curve as a real plane algebraic curve is visible as part of the cross section of the Riemann surface at the real plane.

Watt curve with parameters $a = 2$, $b = 3$, $c = 2$

The Watt curve with parameters $a = 2$, $b = 3$, $c = 2$,

$$\begin{aligned}\mathcal{C}\colon f(x,y) &= (x^2+y^2)(x^2+y^2-2^2-3^2+2^2)^2 + 4\cdot 2^2y^2(x^2+y^2-3^2)\\ &= (x^2+y^2-9)(x^4+2x^2y^2+y^4-9x^2+7y^2) = 0,\end{aligned}$$

decomposes into two irreducible components, a circle and a lemniscate.

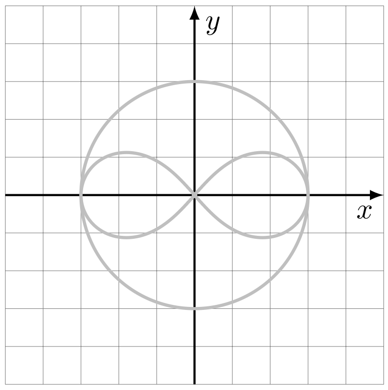


Figure 7.5.13: The Watt curve with parameters $a = 2$, $b = 3$, $c = 2$ as a real plane algebraic curve

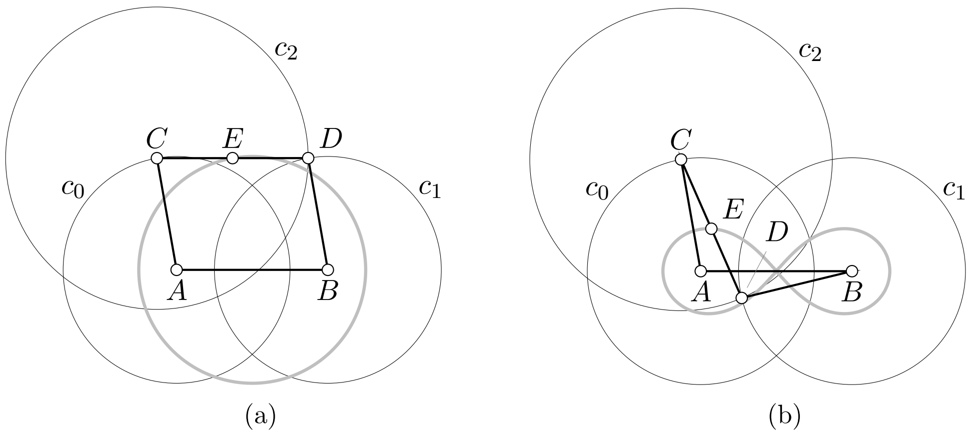


Figure 7.5.14: Two instances of Construction 7.5.7. When point C moves on circle c_0, then point E traces an irreducible component of the Watt curve with parameters $a = |AB|/2 = 2$, $b = |AC| = |BD| = 3$, and $c = |CD|/2 = 2$, which decomposes into a circle and a lemniscate.

Whether the locus generation algorithm produces the circle or the lemniscate as a locus depends on the initial position of point E, respectively on which intersection of circles c_1 and c_2 we select for point D.

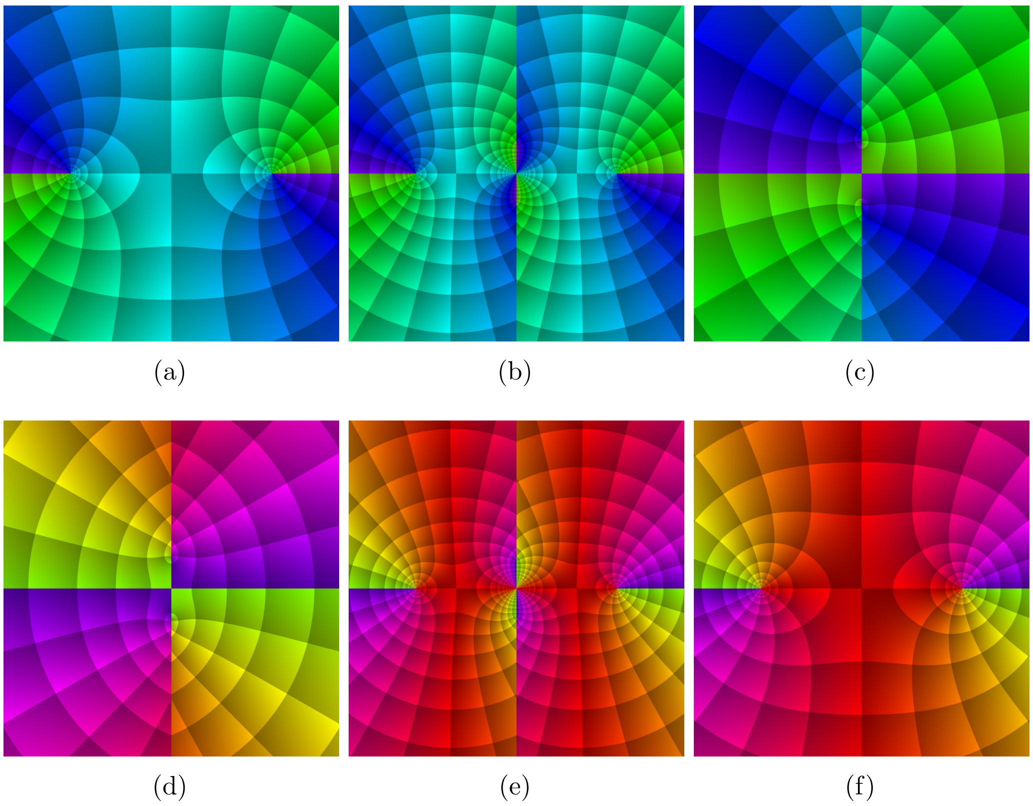


Figure 7.5.15: Domain colouring of six sheets of the Watt curve with parameters $a = 2$, $b = 3$, $c = 2$.

Note that the sheets of the circle, shown in Figure 7.5.15a and Figure 7.5.15f, and the sheets of the lemniscate, shown in Figure 7.5.15b–Figure 7.5.15e, have only the ramification points at -3 and $+3$ in common. Circle and lemniscate are irreducible components of the Watt curve; they are not analytically path-connected.

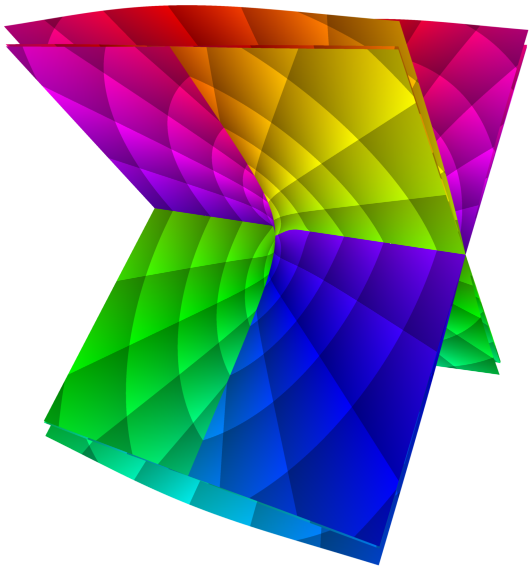


Figure 7.5.16: Domain-coloured Riemann surface of the Watt curve with parameters $a = 2$, $b = 3$, $c = 2$ in perspective projection

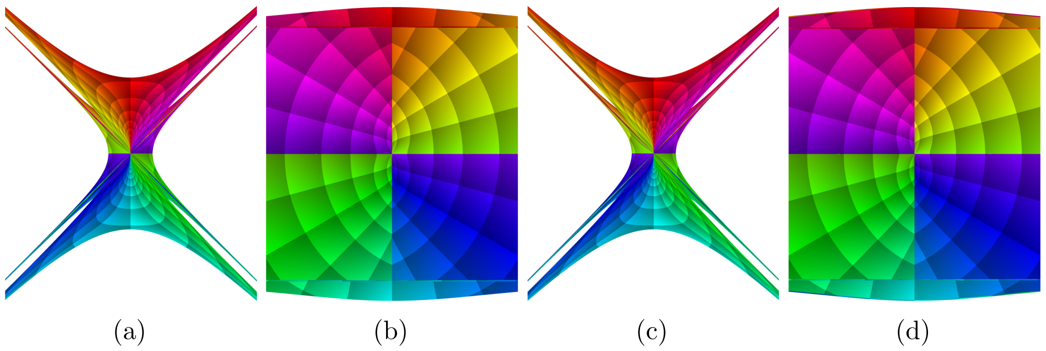


Figure 7.5.17: Domain-coloured Riemann surface of the Watt curve with parameters $a = 2$, $b = 3$, $c = 2$ in orthogonal projection from left, front, right, and back (from left to right)

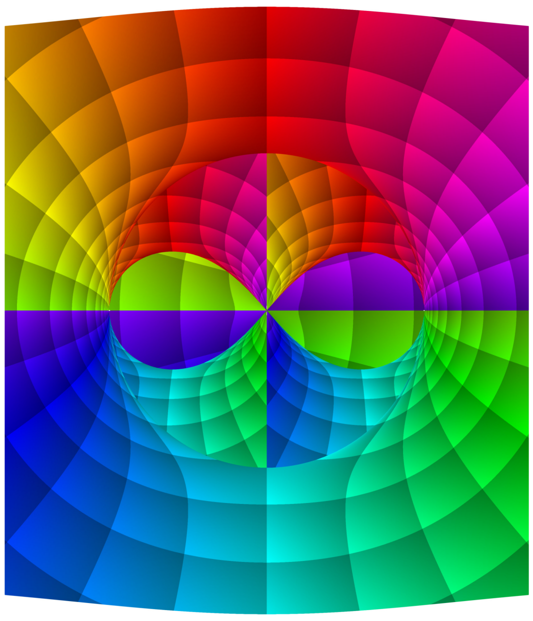


Figure 7.5.18: Domain-coloured Riemann surface of the Watt curve with parameters $a = 2$, $b = 3$, $c = 2$ in orthogonal projection from front, cut off at the real plane. The irreducible components of the Watt curve as a real plane algebraic curve, a circle and a lemniscate, are visible as part of the cross section of the Riemann surface at the real plane.

8 Outlook

Theory

The epsilon-delta bound of Theorem 2.2.1 is applicable to plane algebraic curves defined by an affine algebraic equation $f(x, y) = 0$. It allows us to perform certified homotopy continuation of plane algebraic curves. Projective plane algebraic curves, defined by a homogeneous algebraic equation $f(x, y, z) = 0$, are tractable via suitable projective transformations $z = \beta_1 x + \beta_2 y + \beta_3$ with $\beta_3 \neq 0$. Thus we only need to consider Euclidean coordinate patches. This approach is commonly used for homotopy continuation of systems of polynomials (Morgan and Sommese, 1987, Theorem 2, Theorem 3, and Section 3; Sommese and Wampler, 2005, Chapter 3). Can we generalize the epsilon-delta bound so that it applies more directly to projective plane algebraic curves? Can we find similar bounds of simpler form?

Moreover, a generalization of the algorithm for certified homotopy continuation of systems of plane algebraic curves (Algorithm 2.6.4) to arbitrary (triangular) systems of polynomials might be an interesting challenge for further research.

In Chapter 3, we have discussed an approach, based on (Denner-Broser, 2008; 2013), for tracing a sequence of elementary operations (arithmetic and n-th roots). In principle, we can use it to automatically trace certain computations in programming languages. For example, we may want to trace certain computations written in CindyScript, the scripting language of Cinderella (Kortenkamp and Richter-Gebert, 2006). Can we extend the approach to trigonometric functions, inverse trigonometric functions, exponential function, and logarithms? To that end, we would need to define operations in circular arithmetic for these functions; for inverse trigonometric functions and logarithms, we would also need to find suitable bounds that allow us to select the right branch by proximity.

The locus generation algorithm of Chapter 4 is based on several assumptions. We assume that its complex detours always wind around at most one ramification point of a coordinate of the tracer, and only around ramification points at which the position of the tracer is real-valued (Assumption 4.2.3). If the assumption is violated, the algorithm may miss part of a real algebraic locus. When we trace a geometric construction, can we ensure that a complex detour of time parameter t winds around at most one ramification point?

Besides, the locus generation algorithm is based on the assumption that it does not miss a real position of the tracer (Assumption 4.2.4). For Variant B of the algorithm, this means that we must determine complex points in time along a complex detour when the position of the tracer is real-valued. Can we somehow reliably approximate such points in time?

Can we prove Conjecture 4.4.1 that the locus generation algorithm terminates if it takes small enough complex detours and small enough steps on every complex detour? Can we prove Conjecture 4.5.1 that Variant B of the locus generation algorithm generates an entire real connected component of a real algebraic locus if it takes small enough complex detours and small enough steps on every complex detour?

In Chapter 5, we have discussed randomized algorithms for identification of real algebraic loci from a point cloud generated by the locus generation algorithm. Algorithm 5.2.2 allows us to determine an approximate implicit equation of a real plane algebraic curve containing the real algebraic locus; for real algebraic loci contained in rational plane algebraic curves, Algorithm 5.2.10 allows us to determine an approximate rational parameterization. For both algorithms, we need to determine whether all points of the input point cloud lie on a candidate plane algebraic curve up to a tolerance TOL. How do we best measure whether a point lies on a plane algebraic curve? How should we choose TOL depending on the degree of the candidate equation and depending on the precision of the point cloud? What is the maximum degree of a plane algebraic curve $\mathcal{C}\colon f(x, y) = 0$ that we can likely identify for a given input precision? Can we generalize the approach to determine polar equations of (rational) plane algebraic curves?

We have used Algorithm 6.2.1 to generate Riemann surface meshes for a plane algebraic curves and Algorithm 6.2.6 to visualize them as domain-coloured surfaces. The quality of the visualization depends to a large extent on the quality of the root-finding algorithm used in the generation of the Riemann surface mesh. Are there better choices than Laguerre's method or Weierstraß–Durand–Kerner method that are more well-suited for implementation in GPU shaders?

Practice

It would be nice if the algorithms of this dissertation could be used by anybody to interactively experiment with geometric constructions and gain insight about their dynamic behaviour, beyond the examples discussed in this dissertation.

To that end, the development of CindyJS (The CindyJS Project, 2015), a reimplementation of Cinderella as a JavaScript framework for interactive (mathematical) content, provides a great opportunity. The project plans to implement and compare various approaches for complex tracing of geometric constructions. The approach of Chapter 3, based on (Denner-Broser, 2008; 2013), could be among them. For drawing rational plane algebraic curves containing a real algebraic locus, the algorithm for approximate rational parameterization (Algorithm 5.2.10) could be used.

I would like to release the WebGL implementation of the algorithms for the visualization of algebraic Riemann surfaces (Algorithm 6.2.1 and Algorithm 6.2.6) as a CindyJS plugin. I would also like to develop a website where users can visualize algebraic Riemann surfaces by entering the defining equation of a plane algebraic curve.

Bibliography

Ahlfors, Lars Valerian. 1979. *Complex Analysis*, 3rd ed., McGraw-Hill, Singapore.

Alefeld, Götz and Jürgen Herzberger. 1983. *Introduction to Interval Computations*, translated by Rokne, Jon, Academic Press, New York, NY, USA.

Allgower, Eugene L. and Kurt Georg. 1990. *Numerical Continuation Methods*: *An Introduction*, Springer Series in Computational Mathematics, vol. 13, Springer, Berlin.

Beltrán, Carlos and Anton Leykin. 2012. *Certified Numerical Homotopy Tracking*, Experimental Mathematics **21**, no. 1, 69–83, DOI 10.1080/10586458.2011.606184.

———. 2013. *Robust certified numerical homotopy tracking*, Foundations of Computational Mathematics **13**, no. 2, 253–295, DOI 10.1007/s10208-013-9143-2.

Boubekeur, Tamy and Christophe Schlick. 2008. *A Flexible Kernel for Adaptive Mesh Refinement on GPU*, Computer Graphics Forum **27**, no. 1, 102–113, DOI 10.1111/j.1467-8659.2007.01040.x.

Brieskorn, Egbert and Horst Knörrer. 1986. *Plane algebraic curves*, translated by Stillwell, John, Birkhäuser, Basel (English).

The CindyJS Project. 2015. *CindyJS*: *A JavaScript framework for interactive (mathematical) content*. `http://cindyjs.org`.

Crane, Keenan, Ignacio Llamas, and Sarah Tariq. 2007. *Real-Time Simulation and Rendering of 3D Fluids*, GPU gems 3 (Hubert Nguyen, ed.), Addison-Wesley, pp. 633–675.

Denner-Broser, Britta. 2008. *Tracing-Problems in Dynamic Geometry*, dissertation, Freie Universität Berlin.

———. 2013. *About Tracing Problems in Dynamic Geometry*, Discrete & Computational Geometry **49**, no. 2, 221–246, DOI 10.1007/s00454-012-9473-x.

NIST (ed.) 2014. *NIST Digital Library of Mathematical Functions*. Release 1.0.9 of 2014-08-29. Online companion to (Olver et al., 2010). `http://dlmf.nist.gov`.

Durand, Émile. 1960. *Equations du type $F(x)$, racines d'un polynôme*, Solutions numériques des équations algébriques, vol. 1, Masson, Paris.

Ferréol, Robert and Jaques Mandonnet. 2005. *Encyclopédie des formes mathématiques remarquables*, `http://www.mathcurve.com`.

Fujiwara, Matsusaburô. 1916. *Über die obere Schranke des absoluten Betrages der Wurzeln einer algebraischen Gleichung*, Tohoku Mathematical Journal, First Series **10**, 167–171.

Gargantini, Irene and Peter Henrici. 1972. *Circular Arithmetic and the Determination of Polynomial Zeros*, Numerische Mathematik **18**, 305–320.

Hauenstein, Jonathan D., Ian Haywood, and Alan C. Liddell Jr. 2014. *An a posteriori certification algorithm for Newton homotopies*, ISSAC '14 (Kobe, Japan, 2014), Proceedings of the 39th International Symposium on Symbolic and Algebraic Computation, ACM, New York, NY, USA, pp. 248–255, DOI 10.1145/2608628.2608651.

Hauenstein, Jonathan D. and Frank Sottile. 2012. *Algorithm 921: alphaCertified*: *Certifying Solutions to Polynomial Systems*, ACM Transactions on Mathematical Software **38**, no. 4, 28:1–28:20, DOI 10.1145/2331130.2331136.

Hoffmann, Tim. 2009. *Discrete Differential Geometry of Curves and Surfaces*, MI Lecture Notes, vol. 18, Faculty of Mathematics, Kyushu University, Japan.

Kerner, Immo O. 1966. *Ein Gesamtschrittverfahren zur Berechnung der Nullstellen von Polynomen*, Numerische Mathematik **8**, no. 3, 290–294, DOI 10.1007/BF02162564.

Klein, Felix. 1926. *Vorlesungen über die Entwicklung der Mathematik im 19. Jahrhundert* (Richard Courant and Otto Neugebauer, eds.), Vol. 1, Springer, Berlin.

Klein, Felix and Robert Hermann. 1979. *Development of mathematics in the nineteenth century*, translated by Ackerman, M., Mathematical Science Press, Brookline, MA, USA. English translation of (Klein, 1926), with an appendix by Robert Hermann.

Kline, Morris. 1972. *Mathematical Thought from Ancient to Modern Times*, Oxford University Press, New York, NY, USA.

Kortenkamp, Ulrich. 1999. *Foundations of Dynamic Geometry*, dissertation, ETH Zürich, Zurich.

Kortenkamp, Ulrich and Jürgen Richter-Gebert. 2001a. *Decision Complexity in Dynamic Geometry*, Automated Deduction in Geometry (Jürgen Richter-Gebert and D. Wang, eds.), Lecture Notes in Artificial Intelligence, vol. 2061, Springer, Heidelberg, 2001, pp. 216–220.

______. 2001b. *Grundlagen dynamischer Geometrie*, Zeichnung – Figur – Zugfigur: Mathematische und didaktische Aspekte dynamischer Geometrie-Software (H.-J. Elschenbroich, Th. Gawlick, and H.-W. Henn, eds.), Franzbecker, Hildesheim, pp. 123–144.

______. 2002. *Complexity issues in dynamic geometry*, Festschrift in the honor of Stephen Smale's 70th birthday (M. Rojas and Felipe Cucker, eds.), World Scientific, pp. 355–404.

______. 2006. *Cinderella: The interactive geometry software.* `http://cinderella.de`.

Kranich, Stefan. 2012. *Real-time Visualization of Geometric Singularities*, Master's thesis, Technische Universität München.

______. 2015a. *An epsilon-delta bound for plane algebraic curves and its use for certified homotopy continuation of systems of plane algebraic curves*, arXiv:1505.03432 [math.CV], available at `http://arxiv.org/abs/1505.03432`.

______. 2015b. *GPU-based visualization of domain-coloured algebraic Riemann surfaces*, arXiv:1507.04571 [cs.GR], available at `http://arxiv.org/abs/1507.04571`.

______. 2015c. *Generation of real algebraic loci via complex detours*, arXiv:1510.05464 [math.AG], available at `http://arxiv.org/abs/1510.05464`.

Lebmeir, Peter and Jürgen Richter-Gebert. 2007. *Recognition of Computationally Constructed Loci*, Automated Deduction in Geometry: 6th International Workshop, ADG 2006, Pontevedra, Spain, August 31–September 2, 2006. Revised Papers (Francisco Botana and Tomas Recio, eds.), Lecture Notes in Computer Science, vol. 4869, Springer, Berlin, pp. 52–67, DOI 10.1007/978-3-540-77356-6_4.

Lebmeir, Peter. 2009. *Feature Detection for Real Plane Algebraic Curves*, dissertation, Technische Universität München.

Lindemann, Ferdinand. 1882. *Ueber die Zahl π*, Mathematische Annalen **20**, no. 2, 213–225, DOI 10.1007/BF01446522.

de Longchamps, Gaston Albert Gohierre. 1890. *Essai sur la Géométrie de la Règle et de l'Équerre*, Librairie Ch. Delagrave, Paris.

______. 1897. *Note sur le bicorne*, Journal des mathématiques spéciales **21**, 35–41.

Loria, Gino. 1910. *Spezielle algebraische und transzendente ebene Kurven: Theorie und Geschichte*, 2nd ed., translated by Schütte, Fritz, Vol. 1, Teubner, Leipzig (German).

Morgan, Alexander and Andrew Sommese. 1987. *A Homotopy for Solving General Polynomial Systems That Respects m-Homogeneous Structures*, Applied Mathematics and Computation **24**, 101–113.

Nieser, Matthias, Konstantin Poelke, and Konrad Polthier. 2010. *Automatic Generation of Riemann Surface Meshes*, Advances in Geometric Modeling and Processing (Bernard Mourrain, Scott Schaefer, and Guoliang Xu, eds.), Lecture Notes in Computer Science, vol. 6130, Springer, Berlin, pp. 161–178, DOI 10.1007/978-3-642-13411-1_11.

O'Connor, John J and Edmund F Robertson. 2006. *Famous curves index*, `http://www-history.mcs.st-and.ac.uk/history/Curves/Curves.html`.

Olver, F. W. J., D. W. Lozier, R. F. Boisvert, and C. W. Clark (eds.) 2010. *NIST Handbook of Mathematical Functions*, Cambridge University Press, New York, NY. Print companion to (NIST, 2014).

Petković, M. and Ljiljana Petković. 1980. *On a representation of the k-th root in complex circular interval arithmetic*, Interval Mathematics (Karl L. E. Nickel, ed.), Academic Press, pp. 473–479, DOI 10.1016/B978-0-12-518850-0.50039-1.

Petković, Ljiljana and M. Petković. 1984. *On the k-th Root in Circular Arithmetic*, Computing **33**, 27–35.

Poelke, Konstantin and Konrad Polthier. 2009. *Lifted Domain Coloring*, Computer Graphics Forum **28**, no. 3, 735–742, DOI 10.1111/j.1467-8659.2009.01479.x.

______. 2012. *Domain Coloring of Complex Functions: An Implementation-Oriented Introduction*, IEEE Computer Graphics and Applications **32**, no. 5, 90–97, DOI 10.1109/MCG.2012.100.

Poncelet, Jean-Victor. 1822. *Traité des propriétés projectives des figures*, Mallet-Bachelier, Paris.

______. 1862. *Applications d'analyse et de géométrie*, Vol. 1, Mallet-Bachelier, Paris.

______. 1865. *Traité des propriétés projectives des figures*, 2nd ed., Gauthier-Villars, Paris.

Press, William H., Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. 2007. *Numerical Recipes: The Art of Scientific Computing*, 3rd ed., Cambridge University Press, New York.

Richter-Gebert, Jürgen. 2011. *Perspectives on Projective Geometry: A Guided Tour Through Real and Complex Geometry*, Springer, Berlin.

______. 2014. Personal communication.

Shapiro, Stewart (ed.) 2005. *The Oxford Handbook of Philosophy of Mathematics and Logic*, Oxford University Press, New York, NY, USA.

Smale, Steve. 1986. *Newton's Method Estimates from Data at One Point*, The Merging of Disciplines: New Directions in Pure, Applied, and Computational Mathematics (Richard E. Ewing, Kenneth I. Gross, and Clyde F. Martin, eds.), Springer, New York, pp. 185–196, DOI 10.1007/978-1-4612-4984-9_13.

Sommese, Andrew J. and Charles W. Wampler II. 2005. *The Numerical Solution of Systems of Polynomials Arising in Engineering and Science*, World Scientific, Singapore.

Trott, Michael. 2008. *The Return of the Riemann Surface*, The Mathematica Journal **10**, no. 4, 626–656, DOI 10.3888/tmj.10.4-1.

Wantzel, Pierre Laurent. 1837. *Recherches sur les moyens de reconnaître si un problème de Géométrie peut se résoudre avec la règle et le compas*, Journal de Mathématiques Pures et Appliquées **1**, no. 2, 366–372.

Wegert, Elias. 2012. *Visual Complex Functions: An Introduction with Phase Portraits*, Birkhäuser, Basel.

Weierstraß, Karl. 1891. *Neuer Beweis des Satzes, dass jede ganze rationale Function einer Veränderlichen dargestellt werden kann als ein Product aus linearen Functionen derselben Veränderlichen*, Sitzungsberichte der Königlich Preußischen Akademie der Wissenschaften zu Berlin **2**, 1085–1101.