ÖAW Austrian Academy of Sciences

Doctoral Program
Computational Mathematics
Numerical Analysis and Symbolic Computation

JKU JOHANNES KEPLER UNIVERSITY LINZ

A symbolic-numeric algorithm for genus computation

Mădalina Hodorog Josef Schicho

DK-Report No. 2010-06 09 2010

A–4040 LINZ, ALTENBERGERSTRASSE 69, AUSTRIA

Supported by

Austrian Science Fund (FWF) Upper Austria

FWF Der Wissenschaftsfonds.

LAND OBERÖSTERREICH

DK sponsors:

- **Johannes Kepler University Linz** (JKU)
- **Austrian Science Fund** (FWF)
- **Upper Austria**

A symbolic-numeric algorithm for genus computation *

Mădălina Hodorog, Josef Schicho
Johann Radon Institute for Computational and Applied Mathematics
Austrian Academy of Sciences
Johannes Kepler University Linz Austria
{madalina.hodorog, josef.schicho}@oeaw.ac.at

Abstract

We report on a method for computing the genus of a plane complex algebraic curve based on the topology of singular points and on knot theory. We propose a symbolic-numeric algorithm to be used for plane complex algebraic curves whose defining polynomials have numeric coefficients. Together with its main functionality to compute the genus, the algorithm provides also tools for computational operations in knot theory. We split the main algorithm into several interdependent subalgorithms. We base some of our subalgorithms on general algorithms from computational geometry (e.g. Bentley-Ottman). Whenever required, we design our own subalgorithms for solving the specific problems (e.g. computation of the Alexander polynomial). We use for the implementation the Axel algebraic geometric modeler, developed at INRIA, Sophia-Antipolis.

Keywords: Plane complex algebraic curve, singularity, stereographic projection, algebraic link, Alexander polynomial, delta invariant, genus.

1 Introduction

To raise new questions, new possibilities, to regard old problems from a new angle, requires creative imagination and marks real advance in science.

Albert Einstein

The genus computation problem is a classical subject in computer algebra. Presently, several symbolic algorithms are available for computing the genus of plane algebraic curves over an algebraically closed field, see [16, 18, 31]. There exist also good implementations for these algorithms in several packages of some well-known computer algebra systems such as: Maple, Magma, Singular, Axiom. We shortly recall these packages, see [15, 17, 19, 21, 27] for further details: (i) algcurves package developed at the Florida University, by Mark van Hoeij,

*The work is supported by the Austrian Science Funds (FWF) under the grant W1214/DK9

written in Maple; (ii) CASA (Computer algebra system for algebraic geometry) package developed at the Research Institute for Symbolic Computation, Hagenberg Austria, written in Maple; (iii) GHS (Gaundry, Hess, Smart) attack package developed at Berlin University, written in Magma; (iv) normal.lib package developed at Kaiserslautern University, written in Singular; (v) PAFF (Package for algebraic function fields in one variable) package developed at INRIA-Roquencort, by Gaétan Haché, written in Axiom. On the other hand, there are situations when computing with numerical coefficients is preferable, for instance when the coefficients are obtained by measurements. At present no such numerical algorithms are available in the literature. In this paper, we propose such an algorithm for the computation of the genus of plane complex algebraic curves which makes use of the advantages of both symbolic and numeric algorithms. The method is based on combinatorial techniques from knot theory (see [9, 24]), that allow us to successfully analyse the singularities of the plane complex algebraic curve by computing their topology. Previous research and results have successfully shown that the topology of the singularities of a plane complex algebraic curve is mainly determined by their algebraic link, see [26]. The algebraic link can be uniquely identified by its corresponding Alexander polynomial, see [40]. From the Alexander polynomial we derive a general formula for the delta-invariant of each singularity of the plane complex algebraic curve, which allow us to compute the value for the genus of the plane complex algebraic curve.

We have to pay further attention while formulating the genus computation problem due to the existence of numeric errors. As expected, we deduce that the value of the computed genus is highly sensitive to tiny perturbations of the coefficients of the defining polynomial of the input complex plane algebraic curve. Therefore we have to take a decision regarding the interpretation of the results or the reformulation of the input problem. We intend to use the same approach proposed by Z. Zeng for solving other numerical problems from algebraic geometry such as: the computation of the greatest common divisors of polynomials, the computation of the rank of matrices or the computation of the solutions of systems of polynomial equations, see [11, 41]. Other approaches are also taken into considerations, see [10, 33]. Further tests in these different directions of research will guide us in making the best decision.

In this paper we present a symbolic-numeric algorithm for computing the genus of plane complex algebraic curves. In Section 2 we introduce the genus computation problem and we propose a strategy for solving it. In Section 3 we describe the steps that solve the genus computation problem. In Section 4 we present the numerical difficulties for the genus computation problem, that arise when numerical data are used; we tests several approaches before proposing the final remedy for these difficulties. We include several tests and experiments in Section 5, while in Section 6 we outline the conclusions and the future directions of research.

2 The Genus Computation Problem

There are no big problems, there are just a lot of little problems.

Henry Ford

2.1 Genus of plane algebraic curves

The objects of our study are the plane algebraic curves over the field of complex numbers. We define the plane algebraic curves over an algebraically closed field in the following way:

Definition 1. *Let K be an algebraically closed field, and $f(x, y) \in K[x, y]$ a nonconstant squarefree polynomial in x and y with coefficients in K. A plane algebraic curve over K is defined as the set of all solutions in K^2 of the equation $f(x, y) = 0$, i.e. the set $C = \{(x, y) \in K^2 | f(x, y) = 0\}$. For the curve C, f is called the defining polynomial of C. The degree of the polynomial $f(x, y)$ is called the degree of the curve C.*

For a plane complex algebraic curve we are interested in a special type of points and that is its singularities (or singular points or multiple points). Informally, the singularities of an algebraic curve are the points where the curve has nasty behaviour such as a cusp or a point of self-intersection. A cusp is a point at which two branches of the curve meet such that the tangents at each branch are equal, while a point of self-intersection (or double point) is a point at which two branches of the curve meet such that the tangents at each branch are distinct, see Figure 1.

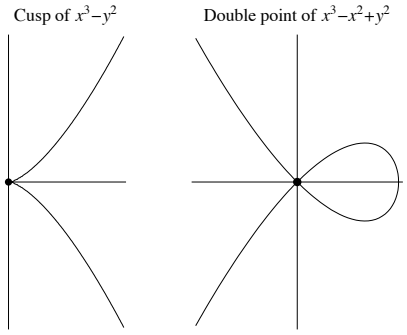


Figure 1: Cusp and double point in the origin

Formally, we can give the following definition for the singularities of a plane algebraic curve over an algebraically closed field:

Definition 2. *Let K be a field (i.e. the complex numbers) and $f(x, y) \in K[x, y]$ be a polynomial in x and y with coefficients over the field K. Let $C = \{(x, y) \in K^2 | f(x, y) = 0\}$ be a plane algebraic curve defined over K and $(a, b) \in C$ be a point on the curve, i.e. $f(a, b) = 0$. The point (a, b) is a singularity of C if and*

only if the x and y partial derivatives of f are both zero at the point (a, b), i.e. $\left(\frac{\partial f}{\partial x}(a,b), \frac{\partial f}{\partial y}(a,b)\right) = (0,0)$.

In the theory of plane algebraic curves, one is interested in computing their genus, which is a birational invariant that plays an important role in the rational parametrization property of plane algebraic curves. From the theory we know that an irreducible plane algebraic curve is rational parametrizable if and only if its genus is 0. The main purpose of this paper is to compute the genus of plane complex algebraic curves.
For algebraic curves with only ordinary singularities we have the following formula for computing the genus:

Theorem 1. *Let C be a plane algebraic curve given by its defining polynomial $f(x, y)$ of degree d. We denote by $OrdSing(C)$ the set of ordinary singular points of the curve C. For a point $P \in OrdSing(C)$ we denoted by m_P the multiplicity of C at the point P. Then the genus of the plane algebraic curve, denoted with $genus(C)$, is computed using the following formula:*

$$genus(C) := \frac{1}{2}\left[(d-1)(d-2) - \sum_{P \in OrdSing(C)} m_P(m_P - 1)\right],$$

where $genus(C) \in \mathbb{Z}$.

We will not focus on the details of this method for computing the genus, as this is not the purpose of our paper. We advise the reader to consult [6, 14, 32, 38] for more information on this method.
We can compute the genus of a plane algebraic curve over an algebraically closed field using the following definition:

Definition 3. *Let C be a plane algebraic curve defined over an algebraically closed field K, $Sing(C)$ the set of singularities of C, and d the degree of C. Then the genus of the plane algebraic curve, denoted with $genus(C)$, is computed using the following formula:*

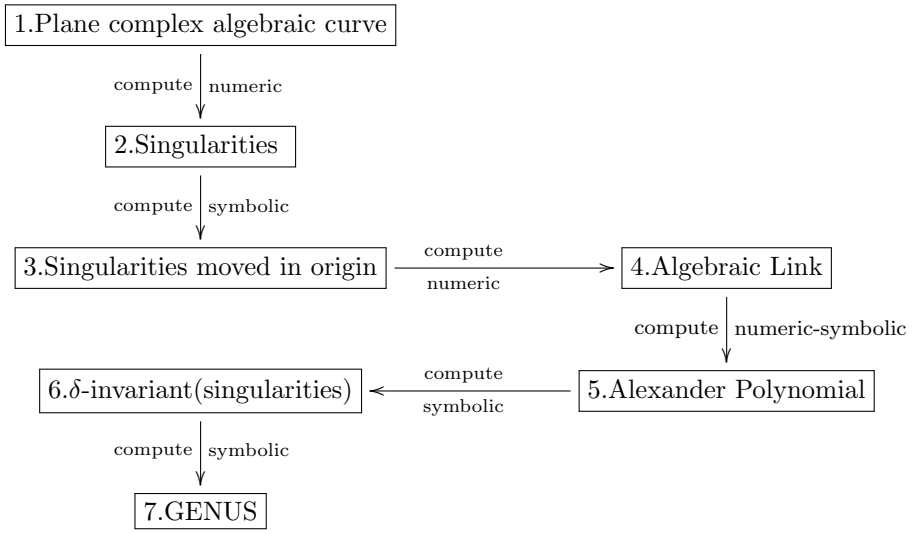
$$genus(C) = \frac{1}{2}(d-1)(d-2) - \sum_{P \in Sing(C)} \delta\text{-}invariant(P),$$

where $genus(C) \in \mathbb{Z}$.

We notice that the computation of the genus reduces to the computation of the *δ-invariant* of each singularity of the curve. We present the method for computing the δ-invariant of each singularity in detail in Section 3.
Thus the genus computation problem that we want to solve is the following: given a plane complex algebraic curve whose defining polynomial contains numeric coefficients, given the degree of the curve and the set of its singularities, we want to compute the value for the genus of the given plane complex algebraic curve.

2.2 Strategy for solving the problem

In order to solve the genus computation problem, we first divide it into several subproblems (some of which are interdependent), we solve each of these subproblems and than we combine the solutions to these subproblems to get the solution to our original problem. We divide the genus computation problem into the following subproblems:



and that is: (i) we compute the singularities of the plane complex algebraic curve; (ii) we translate each computed singularity in the origin; (iii) we compute the algebraic link for each translated singularity; (iv) we compute the Alexander polynomial for each singularity from the algebraic link; (v) we derive a formula for the δ-invariant for each singularity from the Alexander polynomial; (vi) we compute the genus from the δ-invariants of all the singularities.
We use for our implementation the Axel (see [39]) algebraic modeler developed in the Galaad research team at INRIA, Sophia-Antipolis which provides algebraic tools for the manipulation and the computation with implicit curves and surfaces.

3 Why Knot? Alternative Solution to the Genus Computation Problem

Mathematician 1: Okay, so there are three steps to your algorithm. Step one is the input and step three is the output. What is step two?
Mathematician 2: Step two is when a miracle occurs.
Mathematician 1: Oh, I see. Uh, perhaps you could explain that second step a bit more?

K.O.Geddes, S.R.Czapor, G.Labahn

3.1 Computing the Singularities of the Algebraic Curve

The first subproblem that we solve is to compute the singularities of the plane complex algebraic curve C. Given the defining polynomial $F(z,w) \in \mathbb{C}[z,w]$ of C with numerical coefficients, we compute the set of all its singularities, that is the set $Sing(C) = \{(z_0,w_0) \in \mathbb{C}^2 | F(z_0,w_0) = 0, \frac{\partial F}{\partial z}(z_0,w_0) = 0, \frac{\partial F}{\partial w}(z_0,w_0) = 0\}$. In order to compute the set $Sing(C)$ we need to solve an overdeterminate system of polynomial equations in $\mathbb{C}^2$:

$$\begin{cases} F(z_0,w_0) = 0 \\ \\ \dfrac{\partial F}{\partial z}(z_0,w_0) = 0 \\ \\ \dfrac{\partial F}{\partial w}(z_0,w_0) = 0 \end{cases} \tag{1}$$

or equivalently, by replacing $F(z,w) = F(x+iy, u+iv) = s(x,y,u,v) + it(x,y,u,v)$, an overdeterminate system of polynomial equations in $\mathbb{R}^4$:

$$\begin{cases} s(x_0, y_0, u_0, v_0) = 0 \\ t(x_0, y_0, u_0, v_0) = 0 \\ \\ \dfrac{\partial s}{\partial x}(x_0, y_0, u_0, v_0) = 0 \\ \\ \dfrac{\partial t}{\partial x}(x_0, y_0, u_0, v_0) = 0 \\ \\ \dfrac{\partial s}{\partial u}(x_0, y_0, u_0, v_0) = 0 \\ \\ \dfrac{\partial t}{\partial u}(x_0, y_0, u_0, v_0) = 0 \end{cases} . \tag{2}$$

The systems (1), (2) are numerical systems [34, 35]. For solving them, we presently use subdivision methods introduced by [1, 2, 23, 28] and implemented in Mathemagix [22] and in Axel [39]. The subdivision algorithms compute the real solutions for $Sing(C)$. In the future, we intend to use algebraic methods to compute the complex solutions for $Sing(C)$ as proposed in [7]. We also mention the work in progress in the same direction of research done in parallel by [4], and by [25]. Further work in this direction is still required so that we can use these methods for our systems.
For $Sing(C)$ we compute all distinct singularities both in the affine and in the projective space. To do this we homogenize the equation of C and dehomogenize it with respect to different variables. We get three affine open subsets of the projective curve, and we have to be careful not to return singularities in the overlaps twice. We give a schematic summary of the algorithm used to compute the singularities of a plane complex algebraic curve.

Algorithm 1 Singularities of the algebraic curve C SING(F, d)

Input: $C = \{(z, w \in \mathbb{C}^2 | F(z, w) = 0)\}, F \in \mathbb{C}[z, w]$ has numeric coefficients
d the degree of C
Output: $Sing(C)$
where $Sing(C)$ is the set of all singularities of C.

1. Homogenize $F(z, w)$ w.r.t. u obtaining $F_1(z, w, u)$;
   (a) Dehomogenize $F_2(z, w) := F_1(z, w, 1)$
   (b) Get S_1 by solving the system of equations $F_2 = \partial_z F_2 = \partial_w F_2 = 0$
   (c) Dehomogenize $F_3(w, u) := F_1(1, w, u)$
   (d) Get S_2 by solving the system $F_3 = \partial_w F_3 = \partial_u F_3 = u = 0$
   (e) Dehomogenize $F_4(z, u) := F_1(z, 1, u)$
   (f) Get S_3 by solving the system $F_4 = \partial_z F_4 = \partial_u F_4 = z = u = 0$
2. $Sing(C) = S_1 \cup S_2 \cup S_3$

3.2 Computing the Algebraic Link of an Isolated Singularity

The second subproblem that we need to solve is to compute the algebraic link for each isolated singularity of the plane complex algebraic curve. In this subsection we first present the main reasons for studying the algebraic link of an isolated singularity of a plane complex algebraic curve, we then define the algebraic link of an isolated singularity, and we conclude with giving a method for computing the algebraic link of an isolated singularity.
We consider a plane complex algebraic curve as a real two-dimensional subset in $\mathbb{C}^2 \cong \mathbb{R}^4$. We need to study and to understand the topology of these subsets near their singularities, which can be determined by the corresponding algebraic link associated to each singularity.
Milnor proved the following important result concerning the topology of complex hypersurfaces:

Theorem 2. *(Milnor[26]) Let $V \subset \mathbb{C}^{n+1}$ be a hypersurface in $\mathbb{C}^{n+1}$, i.e. an algebraic variety defined by a single polynomial. Assume $\vec{0} \in V$ and $\vec{0}$ is an isolated singularity, i.e. there is no other singularity on a sufficiently small neighborhood of $\vec{0}$; S_ϵ is the sphere centered in $\vec{0}$ and of radius ϵ; and D_ϵ is the disk centered in $\vec{0}$ of radius ϵ. Then, for sufficiently small ϵ, $K = S_\epsilon \cap V$ is a $(2n-1)$-dimensional nonsingular set and $D_\epsilon \cap V$ is homeomorphic to the cone over K.*

In the curve case, $n = 1$ and all singularities are isolated. Next we describe how one can compute the algebraic link associated to an isolated singularity of a plane complex algebraic curve. What we also want is to move the computed algebraic link from $\mathbb{R}^4$ to $\mathbb{R}^3$, and the stereographic projection allows us to accomplish this goal.
We compute the algebraic link of an isolated singularity of the plane complex algebraic curve C in the following way: we consider the curve C which has an isolated singularity in the origin; we take the sphere centered in the origin and of a small radius ϵ; we intersect the curve C with this sphere obtaining a set in the 4-dimensional space, which based on Theorem 2 is an algebraic link for sufficiently small radius. Next, we follow Brauner and Heergaard technique [5] to move the algebraic link from the 4-dimensional space to the 3-dimensional space using the stereographic projection. The stereographic projection allows us not only to project the 4-dimensional link into the 3-dimensional space, but actually preserves all the topological properties of the link from the the 4-dimensional space into the 3-dimensional space.
In 3-dimensions, the stereographic projection is a certain mapping that projects a sphere onto a plane. It is constructed in the following way: we take a sphere; we draw a line from the north pole of the sphere to a point P in the equator plane to intersect the sphere at a point Q. The stereographic projection of P is Q. The map is defined at the sphere minus the north pole. In fact, the stereographic projection gives an explicit homeomorphism from the unit sphere minus the north pole to the Euclidean plane.
More generally, the stereographic projection may be applied to a n-sphere S^n in the $(n+1)$-dimensional Euclidean space R^{n+1} in the following way:

Definition 4. *Consider an n-sphere*
$S^n = \{(x_1, x_2, ..., x_{n+1}) \subset \mathbb{R}^{n+1} | x_1^2 + x_2^2 + ... + x_{n+1}^2 = 1\} \in \mathbb{R}^{n+1}$ *in the* $(n+$

1)-*dimensional Euclidean space* $\mathbb{R}^{n+1}$*, and* $Q(0,0,0,...,1) \in S^n$ *the north point of the n-sphere. If* H *is a hyperplane in* $\mathbb{R}^{n+1}$ *not containing* Q*, then the stereographic projection of the point* $P \in S^n \setminus Q$ *is the point* $P^{'}$ *of the intersection of the line* QP *with* H*. The stereographic projection is a homeomorphism from* $S^n \setminus Q \to \mathbb{R}^n$*.*

We now describe the method used for computing the algebraic link of an isolated singularity. For the given algebraic curve C considered as a real two-dimensional subset in $\mathbb{C}^2 \cong \mathbb{R}^4$:

$$C = \{(x,y,u,v) \in \mathbb{R}^4 | F(x,y,u,v) = 0\} \subset \mathbb{C}^2 \cong \mathbb{R}^4$$

for which the origin $(0,0,0,0)$ is a singularity, that is:

$$\left(F(0,0,0,0), \frac{\delta F}{\delta x}(0,0,0,0), \frac{\delta F}{\delta y}(0,0,0,0), \frac{\delta F}{\delta u}(0,0,0,0), \frac{\delta F}{\delta v}(0,0,0,0)\right) = \\ = (0,0,0,0,0) \quad ,$$

we consider the 3-sphere centrated in the origin and of small radius ϵ:

$$S^3 = \{(z,w) \in \mathbb{C}^2 | |z|^2 + |w|^2 = \epsilon^2\} = \\ = \{(x,y,u,v) \in \mathbb{R}^4 | x^2 + y^2 + u^2 + w^2 = \epsilon^2\} \subset \mathbb{R}^4 \quad .$$

We intersect the given curve with this sphere:

$$X = C \bigcap S^3 = \{(x,y,u,v) \in \mathbb{R}^4 | F(x,y,u,v) = 0, x^2 + y^2 + u^2 + v^2 = \epsilon^2\} \subset \mathbb{R}^4 \quad ,$$

obtaining X, a real 1-dimensional set in the 4-dimensional space.
We take a point on the sphere which is not on the curve, that is:

$$P(0,0,0,\epsilon) \in S^3 \setminus C (i.e. F(0,0,0,\epsilon) \neq 0) \quad ,$$

and we apply the generalized stereographic projection for projecting the set X from the 4-dimensional space into the 3-dimensional space. We define the generalized stereographic projection using the following homeomorphism:

$$f : S^3 \setminus \{P\} \subset \mathbb{R}^4 \to \mathbb{R}^3$$

$$(x,y,u,v) \to (a,b,c)^T = \left(\frac{x}{\epsilon - v}, \frac{y}{\epsilon - v}, \frac{u}{\epsilon - v}\right) \quad .$$

Based on Milnor's results we know that for sufficiently small ϵ the image of X under the stereographic projection f is a link, i.e. $f(X)$ is a link. Next we compute this set $f(X)$:

$$f(X) = \{(a,b,c) \in \mathbb{R}^3 | \exists (x,y,u,v) \in C \cap S^3 : (a,b,c) = f(x,y,u,v)\} \quad .$$

We notice that we can rewrite $f(X)$ in the following way:

$$f(X) = \{(a,b,c) \in \mathbb{R}^3 | \exists (x,y,u,v) = f^{-1}(a,b,c) \in C \cap S^3\} \quad ,$$

since f is an homeomorphism, and so it is a bijection, and therefore f is invertible and it admits an inverse.
We now need to compute the inverse f^{-1} of f:

$$f^{-1} : \mathbb{R}^3 \to S^3 \setminus \{P\}$$

$$(a,b,c) \to (x,y,u,v) = \ ?$$

A straight-forward computation shows that the formula for the inverse f^{-1} is:

$$f^{-1} : \mathbb{R}^3 \to S^3 \setminus \{P\}$$

$$(a,b,c) \to (x,y,u,v) = (\tfrac{2a\epsilon}{1+a^2+b^2+c^2}, \tfrac{2b\epsilon}{1+a^2+b^2+c^2}, \tfrac{2c\epsilon}{1+a^2+b^2+c^2}, \tfrac{-\epsilon+a^2\epsilon+b^2\epsilon+c^2\epsilon}{1+a^2+b^2+c^2}) \ .$$

Denoting the denominator of x, y, u, v with $n = 1 + a^2 + b^2 + c^2$, the formula for the inverse f^{-1} becomes:

$$f^{-1} : \mathbb{R}^3 \to S^3 \setminus \{P\}$$

$$(a,b,c) \to (x,y,u,v) = (\tfrac{2a\epsilon}{n}, \tfrac{2b\epsilon}{n}, \tfrac{2c\epsilon}{n}, \tfrac{-\epsilon+a^2\epsilon+b^2\epsilon+c^2\epsilon}{n}) \ .$$

Now we can finally compute the set $f(X)$:

$$= \left\{(a,b,c) \in \mathbb{R}^3 | f^{-1}(a,b,c) \in V(F)\right\} = \left\{(a,b,c) \in \mathbb{R}^3 | (x,y,u,v) \in V(F)\right\} =$$

$$= \left\{(a,b,c) \in \mathbb{R}^3 | (\tfrac{2a\epsilon}{n}, \tfrac{2b\epsilon}{n}, \tfrac{2c\epsilon}{n}, \tfrac{-\epsilon+a^2\epsilon+b^2\epsilon+c^2\epsilon}{n}) \in V(F)\right\} =$$

$$= \left\{(a,b,c) \in \mathbb{R}^3 | F(\tfrac{2a\epsilon}{n}, \tfrac{2b\epsilon}{n}, \tfrac{2c\epsilon}{n}, \tfrac{-\epsilon+a^2\epsilon+b^2\epsilon+c^2\epsilon}{n}) = 0\right\} =$$

$$= \left\{(a,b,c) \in \mathbb{R}^3 | \begin{cases} G := Re\big(F(\tfrac{2a\epsilon}{n}, \tfrac{2b\epsilon}{n}, \tfrac{2c\epsilon}{n}, \tfrac{-\epsilon+a^2\epsilon+b^2\epsilon+c^2\epsilon}{n})\big) = 0 \\ H := Im\big(F(\tfrac{2a\epsilon}{n}, \tfrac{2b\epsilon}{n}, \tfrac{2c\epsilon}{n}, \tfrac{-\epsilon+a^2\epsilon+b^2\epsilon+c^2\epsilon}{n})\big) = 0 \end{cases} \right\}$$

We notice that G and H are two polynomials in (a,b,c) with real coefficients. Their common zero set in $\mathbb{R}^3$ is equal to the algebraic link.
We give a schematic summary of the algorithm used to compute the algebraic link of an isolated singularity of a plane complex algebraic curve.

Algorithm 2 Algebraic link of the isolated singularity $(0,0)$ ALGLINK(F,ϵ)

Input: $F \in \mathbb{C}[z,w]$ with (0,0) an isolated singularity , $\epsilon \in \mathbb{R}^*$ with $\epsilon > 0$
Output: $G,\ H \in \mathbb{R}[a,b,c]$
where the common zero set of G, H is the algebraic link of F in $(0,0)$.

1. Intersect the plane complex algebraic curve:

$$C : F(z,w) = 0 \Leftrightarrow F(x+iy, u+iv) = 0 \ ,$$

that has an isolated singularity in the origin $(0,0)$, with the sphere centrated in the origin and of small radius ϵ:

$$S^3 : x^2 + y^2 + u^2 + v^2 - \epsilon^2 = 0 \ .$$

2. Obtain $X = C \cap S^3 \subset \mathbb{R}^4$;

3. Consider P a point on the sphere but not on the curve;

4. Project X into the 3-dimensional space using the generalized stereographic projection:

$$f : S^3 \setminus \{P\} \subset \mathbb{R}^4 \to \mathbb{R}^3 \ ,$$
$$(x,y,u,v) \to (a,b,c) = (\tfrac{x}{\epsilon - v}, \tfrac{y}{\epsilon - v}, \tfrac{u}{\epsilon - v}) \ .$$

5. Compute the inverse:

$$f^{-1} : \mathbb{R}^3 \to S^3 \setminus \{P\}$$

$$(a,b,c) \to (x,y,u,v) = (\tfrac{2a\epsilon}{n}, \tfrac{2b\epsilon}{n}, \tfrac{2c\epsilon}{n}, \tfrac{-\epsilon + a^2\epsilon + b^2\epsilon + c^2\epsilon}{n}) \ ,$$

where $n = 1 + a^2 + b^2 + c^2$.

6. Compute $f(X)$ using the inverse f^{-1} finding G, H:

$$f(X) = \left\{ (a,b,c) | \left\{ \begin{array}{l} G := Re(F(\tfrac{2a\epsilon}{n}, \tfrac{2b\epsilon}{n}, \tfrac{2c\epsilon}{n}, \tfrac{-\epsilon + a^2\epsilon + b^2\epsilon + c^2\epsilon}{n})) = 0 \\ H := Im(F(\tfrac{2a\epsilon}{n}, \tfrac{2b\epsilon}{n}, \tfrac{2c\epsilon}{n}, \tfrac{-\epsilon + a^2\epsilon + b^2\epsilon + c^2\epsilon}{n})) = 0 \end{array} \right. \right\}.$$

3.3 Computing the Alexander Polynomial of an Algebraic Link

Knot theory and the Alexander polynomial

For our purpose, we distinguish between the following types of knots:

Definition 5. *1. A knot is a piecewise linear or a differentiable simple closed curve in the 3-dimensional space* $\mathbb{R}^3$.
2. A link is a finite union of disjoint knots. The individual knots which make up a link are called the components of the link. A knot will be considered a link with one component (Figure 2).
3. A link is called algebraic if it arises as the intersection of an algebraic curve with a sufficiently small sphere, as described in Subsection3.2.

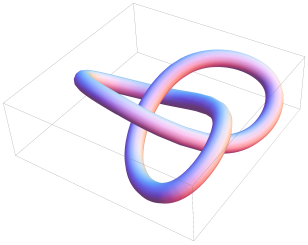


Figure 2: Trefoil knot (figure produced with Mathematica)

In our approach, we approximate a differentiable algebraic link, namely the intersection of G and H computed in Subsection 3.2, by a piecewise linear algebraic link. From now on, we only consider piecewise linear links. When we work with knots, we work with their projection in the 2-dimensional space.

Definition 6. *A regular projection is a linear projection for which no three points on the knot project to the same point, and no vertex projects to the same point as any other point on the knot. A crossing point is an image of two knot points of such a regular projection from* $\mathbb{R}^3$ *to* $\mathbb{R}^2$. *Then:*
1. A link diagram (or simply diagram) is the image under regular projection, together with the information on each crossing point telling which branch goes over and which goes under. Thus we speak about overcrossings and undercrossings.
2. A diagram together with an arbitrary orientation of each knot in the link is called an oriented diagram.

We are interested in the following elements of a diagram:

Definition 7. *1. A crossing is lefthanded if the underpass traffic goes from left to right or it is righthanded if the underpass traffic goes from right to left. We denote a lefthanded crossing with* -1 *and a righthanded crossing with* $+1$.
2. An arc is the part of a diagram between two undercrossings (Figure 3). Whether lefthanded or righthanded, each crossing is determined by three arcs and we denote the overgoing arc with i, *and the undergoing arcs with* j *and* k *(Figure 4). We notice that the number of arcs in a link diagram is equal to the number of crossings in the same link diagram.*

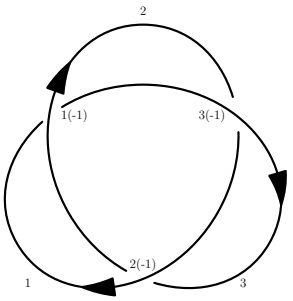


Figure 3: Oriented diagram of the trefoil with 3 arcs and 3 lefthanded crossings

Figure 4: Types of crossings: lefthanded (-1) and righthanded(+1).

The main problem in knot theory is to distinguish between different links and to establish whether two links are equivalent or not. We define the equivalence of links by the following definition called (ambient) isotopy:

Definition 8. *We define a homeomorphism as a continuous bijective function with a continuous inverse. Then we say that two links are equivalent if there exists an orientation-preserving homeomorphism on* $\mathbb{R}^3$ *that maps one link onto the other.*

To prove that two links are not equivalent we use the notion of link invariants:

Definition 9. *A link invariant is a function from link diagrams to some discrete set (*$\mathbb{Z}$ *or* $\mathbb{Z}[t]$*) which is unchanged under the Reidemeister moves of type I, II or III (see Figure 5).*

Some link invariants are: the tricolorability, the unknotting number, the Jones polynomial, the Alexander polynomial. For further details the reader can consult [9, 24]. At present, there exists no complete invariant for links.
Still for our purpose we are intersted only in the invariants of the algebraic links. An important result in this direction of research was proved by Yamamoto in 1984 (see [40]), who showed that the Alexander polynomial is a complete invariant for the algebraic links, that is the Alexander polynomial uniquely defines all the algebraic links up to an (ambient) isotopy.
We now focus our attention on the definition and on the computation of the Alexander polynomial of a link. The Alexander polynomial was introduced

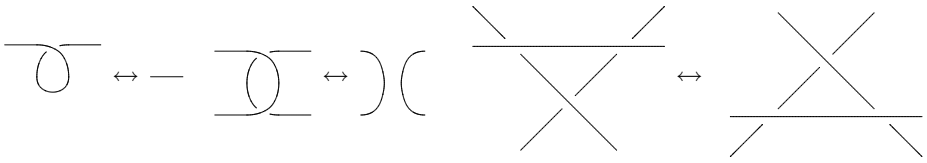


Figure 5: Reidemeister moves of type I, II, III

by Alexander in 1928 (see [3]). It depends on the fundamental group of the complement of the link in $\mathbb{R}^3$ and we define it as follows:

Definition 10. *Let L be a link with m components. The multivariate Alexander polynomial is a Laurent polynomial $\Delta_L \in \mathbb{Z}[t_1^{\pm 1}, ..., t_m^{\pm 1}]$, which is uniquely defined for each link up to a factor of $\pm t_1^{k_1} ... t_m^{k_m}$, with $k_i \in \mathbb{Z}$ and up to a substitution $t_i := \frac{1}{t_i}$, for all $i \in \{1, ..., m\}$ ([8]).*

We follow [24] in our approach to compute the Alexander polynomial. We distinguish three steps when computing the Alexander polynomial Δ_L of an oriented link diagram $D(L)$:

$$\boxed{D(L)} \longrightarrow \boxed{\text{Labelling matrix}(L)} \longrightarrow \boxed{\text{Prealexander matrix}(L)} \longrightarrow \boxed{\Delta_L}$$

First of all we compute the labelling matrix of $D(L)$ defined as follows:

Definition 11. *Let $D(L)$ be an oriented link diagram with m components and n crossings $x_q : q \in \{1, ..., n\}$. We denote the arcs of $D(L)$ with the labels $\{1, ..., n\}$ and separately the crossings of $D(L)$ with the labels $\{1, ..., n\}$. We denote the labelling matrix of $D(L)$ with $LM(L) \in \mathcal{M}(n, 4, \mathbb{Z})$. We define $LM(L) = (b_{ql})_{q,l}$ with $q \in \{1, ..., n\}, l \in \{1, ..., 4\}$ row by row for each crossing x_q as follows:*
-on position b_{q1} we store the type of the crossing x_q ($+1$ or -1);
-on position b_{q2} we store the label of the arc i of the crossing x_q in $D(L)$;
-on position b_{q3} we store the label of the arc j of the crossing x_q in $D(L)$;
-on position b_{q4} we store the label of the arc k of the crossing x_q in $D(L)$.

Secondly we compute the prealexander matrix of $D(L)$ defined using the labelling matrix $LM(L)$ as follows:

Definition 12. *Let $D(L)$ be an oriented link diagram with m components and n crossings $x_q : q \in \{1, ..., n\}$. We denote the arcs and the crossings of $D(L)$ as in Definition 11. We consider $LM(L)$ the labelling matrix of $D(L)$ as in Definition 11. We denote the prealexander matrix of L with $PM(L) \in \mathcal{M}(n, n, \mathbb{Z}[t_1, t_2, ..., t_m])$. We define $PM(L)$ row by row for each crossing x_q depending on $LM(L)$. For x_q we consider the variable t_s, where $s \in \{1, ..., m\}$ is the s-th knot component of $D(L)$, which contains the overgoing arc that determines the crossing x_q. Then:*

-if the crossing x_q is righthanded, i.e. $b_{q1} = +1$ in $LM(L)$ then at position b_{q2} of $PM(L)$ we store the label $1 - t_s$, at position b_{q3} we store the label -1 and at position b_{q4} we store the label t_s;
-if the crossing x_q is lefthanded, i.e. $b_{q1} = -1$ in $LM(L)$ then at position b_{q2} of $PM(L)$ we store the label $1 - t_s$, at position b_{q3} we store the label t_s and at position b_{q4} we store the label -1;
-if two or all of the positions b_{q2}, b_{q3}, b_{q4} have the same value, then we store the sum of the corresponding labels at the corresponding position;
-all other entries of the matrix are 0.

Finally, we define the Alexander polynomial of $D(L)$ depending on the number of components in L:

Definition 13. *Let $D(L)$ be an oriented link diagram with m components and n crossings, $LM(L)$ be its labelling matrix as in Definition 11 and $PM(L)$ be its prealexander matrix as in Definition 12.*
1. Univariate case, (L has one component, $m = 1$, see [24]).
The univariate Alexander polynomial $\Delta_L(t_1) \in \mathbb{Z}[\pm t_1]$ is the normalized polynomial computed as the determinant of any $(n-1) \times (n-1)$ minor of the prealexander matrix of $D(L)$.
2. Multivariate case, (L has more than one component, $m \geq 2$, see [8]).
The multivariate Alexander polynomial $\Delta_L(t_1, ..., t_m) \in \mathbb{Z}[t_1^{\pm 1}, ..., t_m^{\pm 1}]$ is the normalized polynomial computed as the greatest common divisor of all the $(n-1) \times (n-1)$ minor determinants of the prealexander matrix of $D(L)$.

A normalized polynomial is a polynomial in which the term of the lowest degree is a positive constant.

Example. We compute the Alexander polynomial of the oriented diagram of the trefoil knot L from Figure 3. We denote the arcs and separately the crossings of the diagram with the labels $\{1, 2, 3\}$. We compute the labelling matrix of L with Definition 11:

$$LM(L) = \left(\begin{array}{c|cccc} & type & label_i & label_j & label_k \\ \hline c_1 & -1 & 2 & 1 & 3 \\ c_2 & -1 & 1 & 3 & 2 \\ c_3 & -1 & 3 & 2 & 1 \end{array}\right)$$

From $LM(L)$ we compute the prealexander matrix of $D(L)$ with Definition 7 and Definition 12. We notice that L has only one knot compoent so $s = 1$ in Definition 12:

$$PM(L) = \left(\begin{array}{c|ccc} & label_i & label_j & label_k \\ \hline c_1 & 2 & 1 & 3 \\ -1 & 1-t_1 & t_1 & -1 \\ \hline c_2 & 1 & 3 & 2 \\ -1 & 1-t_1 & t_1 & -1 \\ \hline c_3 & 3 & 2 & 1 \\ -1 & 1-t_1 & t_1 & -1 \end{array}\right) = \left(\begin{array}{c|ccc} & 1 & 2 & 3 \\ \hline c_1 & t_1 & 1-t_1 & -1 \\ c_2 & 1-t_1 & -1 & t_1 \\ c_3 & -1 & t_1 & 1-t_1 \end{array}\right)$$

From $PM(L)$ we compute the Alexander polynomial with Definition 13:

$$det\Big(Minor_{33}\big(PM(L)\big)\Big) = det\begin{pmatrix} t_1 & 1-t_1 \\ 1-t_1 & -1 \end{pmatrix} = -t_1^2 + t_1 - 1$$

$$\Delta_L(t_1) = Normalize(-t_1^2 + t_1 - 1) = t_1^2 - t_1 + 1$$

We note that we apply the same strategy described in this subsection to compute the Alexander polynomial of an algebraic link L, this being a special case of link. We give a schematic summary of the algorithm used to compute the Alexander polynomial of an algebraic link diagram $D(L)$.

Algorithm 3 Alexander polynomial for $D(L)$ ALEXPOLY$(D(L), m, n)$

Input: $D(L)$ oriented algebraic link diagram with m components, n crossings
Output: $\Delta_L(t_1, ..., t_m) \in \mathbb{Z}[t_1^{\pm 1}, ..., t_m^{\pm 1}]$
where $\Delta_L(t_1, ..., t_m)$ is the Alexander polynomial of $D(L)$.

1. Denote the arcs and separately the crossings of $D(L)$ with $\{1, ..., n\}$;
2. Compute $LM(L)$ the labelling matrix of $D(L)$;
3. Compute $PM(L)$ the prealexander matrix of $D(L)$;
4. If $m = 1$ then:
   (a) Compute M any $(n-1) \times (n-1)$ minor of $PM(L)$;
   (b) Compute D the determinant of the minor M;
   (c) $\Delta_L(t_1) = Normalize(D)$;
5. If $m \geq 2$ then:
   (a) Compute all the $(n-1) \times (n-1)$ minors of $PM(L)$;
   (b) Compute G the greatest common divisor of all the computed minors in 5.(a);
   (c) $\Delta_L(t_1, ...t_m) = Normalize(G)$.

Alexander polynomial and computational geometry

In Subsection 3.3 we noticed that in order to compute the Alexander polynomial of an algebraic link L we need to compute the diagram of L denoted with $D(L)$. We compute L using the stereographic projection method described in Subsection 3.2 and Axel system [39] for the actual implementation. We remember that for the plane complex algebraic curve C with $F(z, w) \in \mathbb{C}[z, w], \epsilon \in \mathbb{R}^*, \epsilon > 0$ and $(0, 0)$ an isolated singularity, we compute two polynomials $G, H \in \mathbb{R}[a, b, c]$. We have shown that the algebraic link L of $(0, 0)$ is the zero common set of G, H, that is L is a smooth implicit curve in $\mathbb{R}^3$ given as the intersection of two implicit surfaces in $\mathbb{R}^3$ whose defining polynomials are G, H. Axel uses certified algorithms to compute a piecewise linear approximation $L^{'}$ for L, which is isotopic to L[23]. $L^{'}$ is computed as a graph $G = \langle \mathcal{P}, \mathcal{E} \rangle$, where $\mathcal{P}$ is a set of points together with their euclidean coordinates and $\mathcal{E}$ is a set of edges connecting them. From now on we denote $L^{'} := Graph(L)$.

Example 1 (Compute algebraic link with Axel).
For $C^4 = \{(z, w) \in \mathbb{C}^2 | z^3 - w^2 = 0\} \subset \mathbb{R}^4$ with $(0, 0)$ isolated singularity and

$\epsilon = 1$ we get L, the algebraic link of $(0,0)$ by the stereographic projection method proposed in Subsection 3.2:

$$f(C^4 \cap S) := L = \{(a,b,c) \in \mathbb{R}^3 | G := ReF(...) = 0, H := ImF(...) = 0\}$$

and with Axel we get $Graph(L)$ as the intersection of the two surfaces G, H:

$$Graph(L) = \langle \mathcal{V}, \mathcal{E} \rangle,\ \mathcal{V} = \{p = (m,n,q) \in \mathbb{R}^3\},\ \mathcal{E} = \{(i,j) | i,j \in \mathcal{V}\}$$

such that $Graph(L) \cong_{isotopic} L$, see Figure 6.

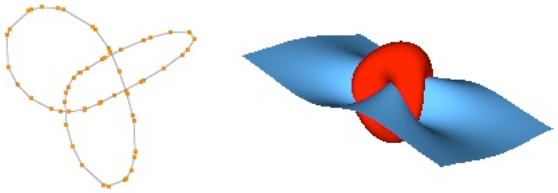


Figure 6: Algebraic link as intersection of surfaces in Axel

Now from the ouput computed by Axel, $Graph(L)$, we need to compute $D(L)$, the diagram of the algebraic link L. We then use $D(L)$ to compute the Alexander polynomial of L with the algorithm proposed in Subsection 3.3.
We compute the elements of $D(L)$, i.e. the arcs of the diagram and the number of knot components in the diagram, plus for each knot component its crossings with their types. We develop new computational algorithms for computing $D(L)$ given $Graph(L)$. The main idea is that all these algorithms are operating on the data structure of $Graph(L)$ returned by Axel. Each point in the graph is given as a 4-tuple $p(index, x, y, z)$, where $index$ is an integer that uniquely identifies each point, and $(x,y,z) \in \mathbb{R}^3$ are the euclidean coordinates of p. We use $xyzcoord(index)$ for denoting the x, y, z coordinates of $index$, $xycoord(index)$ for denoting the x, y coordinates of $index$, $xcoord$ for denoting the x coordinate of $index$, and $ycoord$ for denoting the y coordinate of $index$. Each edge in the graph is given by a pair $e(source, destination)$, where $source$ is the index of the source point of e, and $destination$ is the index of the destination point of e. For simplicity reasons, we denote the pair $e(source, destination) := e(s,d)$. We consider the edges of $Graph(L)$ to be "small" edges, i.e. the projection of any edge of $Graph(L)$ has at most one crossing point. Here we shortly describe these computational algorithms, for more information the reader can consult [20].
The first algorithm is an adapted version of the Bentley-Ottman algorithm [12]. For $Graph(L) \in \mathbb{R}^3$ with the set of points $p_i = (x_i, y_i, z_i) \in \mathbb{R}^3$ we consider its projection in $\mathbb{R}^2$ with the set of points $p_i = (x_i, y_i) \in \mathbb{R}^2$. We also consider no vertical edges in the projection. This algorithm computes the intersection points of all the edges of the projection of $Graph(L)$ and some extra information: (i) for each intersection point p the pair of edges (e_i, e_j) that contains p; (ii) and each pair of edges (e_i, e_j) is ordered, i.e. e_i is under e_j in $\mathbb{R}^3$. These intersection points together with the extra information coincide with the crossings of $D(L)$. Our adapted Bentley-Ottman algorithm operates as follows:

- the edges of the projection of $Graph(L)$ are oriented from left to right and they are ordered in a list $E = \{e_0, ..., e_N\}$ as follows: (1) by the x-coordinates of their source points; (2) if the x-coordinates of the source

points of two edges coincide, then the two edges are ordered by the two slopes of their supporting lines; (3) if the x-coordinates of the source points and the slopes of two edges coincide, then the two edges are ordered by the y-coordinates of their destination points. The ordering criteria is necessary for the correctness of the algorithm.

- we consider a vertical sweep line l that sweeps the plane from left to right. While l moves, it intersects several edges from E. The list of edges that intersect l at one point during the sweeping procees, denoted SW, is called the sweep list. SW changes while l sweeps the plane. The algorithm is based on the key observation that SW is updated only at certain points of the edges from E called event points. The sweep list SW is ordered in this algorithm by the y-coordinates of the intersections of the edges of E with the sweep line L.

- we notice that in E each *index* appears two times in E. Due to this property, we can manage SW in a simpler way in our adapted Benttley-Ottman algorithm than in the original version.

- while we traverse E, we insert the current edge $e_m(s_m, d_m)$ from E in SW in the right position and that is: (1) we search for an edge $e_n(s_n, d_n)$ in SW such that its destination coincide with the source of $e_m \in E$, i.e. $d_n = s_m$; if we find such an $e_n \in SW$ we replace it with $e_m \in E$; (2) if such an edge $e_n \in SW$ does not exist, we insert e_m in SW depending on its position against the current edges from SW. We assume $SW = \{e_{i_1}, e_{i_2}, e_{i_3}, ..., e_{i_k}\}$, with $e_{i_q} \in E$ for all $q \in \{1, ..., k\}$. There exists a unique index j with $0 \leq j \leq k$ such that $ycoord(s_m)$ is larger than the y-coordinates of all the intersections of $e_{i_1}, ..., e_{i_j}$ with l and smaller than the y-coordinates of all the intersections of $e_{i_{j+1}}, ..., e_{i_k}$ with l. This index j can be found by checking all the signs of the determinants $det\big[\big(xycoord(s_m), 1\big), \big(xycoord(s_{i_j}), 1\big), \big(xycoord(d_{i_j}), 1\big)\big]$. Then we insert e_m in SW between the two edges e_{i_j} and $e_{i_{j+1}}$ and we obtain $SW = \{e_{i_1}, e_{i_2}, ..., e_{i_j}, e_m, e_{i_{j+1}}, ..., e_{i_k}\}$. When we insert an edge from E into SW on the right position we have to additionally update SW depending on the event points encountered:

  - we test each inserted edge in SW against its two neighbors for intersection. If an intersection point p is found we report it together with the ordered pair of edges that contains it. In addition we swap the edges that intersect in SW. As opposed to the original Bentley-Ottman algorithm after swaping the edges in SW, we do not test the edges against their new neighbors for intersections because we consider only "small" edges.
  - we test each inserted edge in SW against its two neighbors for common destination. In addition, when two edges are swapped in SW after reporting their intersection point, we test them against their new neighbors for common destination. Whenever we find two consecutive edges with common destinations we erase them from SW. As opposed to the original Bentley-Ottman algorithm after deleting edges from SW, we do not test the new neighbors for intersection because we consider only "small" edges.

The second algorithm constructs the knot components of the diagram from the projection of $Graph(L)$. It also returns the total number of knot components. We consider E as in the previous algorithm. We denote a positive edge in $\mathbb{R}^2$ with $e(s,d)$, and its corresponding negative edge with $-e(d,s)$. The positive edges are oriented from left to right, while the negative ones are oriented from right to left. We denote the knots with $K_i, i \in \mathbb{N}$. All K_i have the properties: (1) for each edge $e_k(s_k,d_k) \in K_i$ there exists $e_{k+1}(s_{k+1},d_{k+1}) \in K_i$ with $d_k = s_{k+1}$; (2) for $K_i = \{e_0(s_0,d_0),...,e_n(s_n,d_n)\}$: $d_n = s_0$. As opposed to the list E which contains only positive edges oriented from left to right, each list K_i contains both positive and negative edges. We initialize the first knot K_0 with the first edge $e_0(s_0,d_0)$ from E. Next we look for the edge e_n in E which has a common index, either source or destination, with d_0. If we find $e_n(d_0,d_n) \in E$ then we insert $e_n(d_0,d_n)$ in K_0 as a positive edge. If we find $e_n(s_n,d_0) \in E$ then we insert $-e_n(d_0,s_n)$ in K_0 as a negative edge. After we insert e_n in K_0 we erase it from E. We will always find such an edge e_n in E, because each *index* such as d_0 appears two times in E. We continue with inserting edges in K_0 from E until the *destination* of an inserted edge coincide with s_0 the source of the first edge from K_0. We apply the same strategy to constructs all the knots K_i of $D(L)$ until E is empty, increasing i each time a new knot starts being constructed. At the end of the algorithm, the index i returns the total number of knot components of $D(L)$.

The third algorithm constructs the arcs for each knot component of the link. It also decides the type of crossings (righthanded or lefthanded) for each knot component. For constructing the arcs, we consider E as in the previous algorithms. This algorithm operates on the outputs of the previous two algorithms, i.e. the list of intersection points I together with the list of ordered pairs of edges E_I, and the lists of edges for all the knot components $K_i, i \in \mathbb{N}$. The key point of the algorithm is to search in K_i all the undergoing edges from E_I and to splitt them in two parts. For instance, we assume that for $E = \{e_0,...,e_n,e_m,...,e_l,e_k...,e_t,e_s,...,e_{last}\}$, we compute the following outputs with the previous two algorithms:

$$I = \{(x_1,y_1),(x_2,y_2),(x_3,y_3)\}, E_I = \{(-e_n,e_m),(e_l,e_k),(e_s,-e_t)\}$$

$$K_0 = \{e_0,...,e_k,...,e_s,...,e_m,...,e_l,...,-e_t,...,-e_n,...,-e_1\}$$

We search the three undergoing edges $-e_n, e_l, e_s$ one by one in K_0 and we replace them with $-e_n \to (-e_n^d, -e_n^u), e_l \to (e_l^d, e_l^u), e_s \to (e_s^d, e_s^u)$ obtaining:

$$K_0^{'} = \{e_0,..,e_k,..,e_s^d,e_s^u,..,e_m,..,e_l^d,e_l^u,..,-e_t,..,-e_n^d,-e_n^u,..,-e_1\}.$$

From Definition 7, we conjecture that an arc contains the list of edges from a modified knot component $K_i^{'}, i \in \mathbb{N}$ starting with an edge of type $e_j^u, j \in \mathbb{N}$ from $K_i^{'}$ and ending with the next consecutive edge of type $e_k^d, k \in \mathbb{N}$ from $K_i^{'}$. While we insert the edges from $K_i^{'}$ into the list of edges representing the arcs we erase them from $K_i^{'}$. Thus from the modified loop $K_0^{'}$ we compute the following three arcs until $K_0^{'}$ is empty:

$$K_0^{'} = \{e_0,..,e_k,..,e_s^d,\cancel{[e_s^u,..,e_m,..,e_l^d]},e_l^u,..,-e_t,..,-e_n^d,-e_n^u,..,-e_1\}$$

$$arc_0 = \{e_s^u,...,e_m,...,e_l^d\}$$

$$K_0^{'} = \{e_0, .., e_k, .., e_s^d, \cancel{[e_l^u, .., -e_t, .., -e_n^d]}, -e_n^u, .., -e_1\}$$
$$arc_1 = \{e_l^u, .., -e_t, .., -e_n^d\}$$
$$K_0^{'} = \{\cancel{[e_0, .., e_k, .., e_s^d]}, \cancel{[-e_n^u, .., -e_1]}\}$$
$$arc_2 = \{e_n^u, .., -e_1, e_0, .., e_k, .., e_s^d\}$$

For deciding the type of crossings, we observe that in each knot component for a positive edge $e_i(s_i, d_i) : xcoord(s_i) < xcoord(d_i)$ and for a negative edge $-e_j(s_j, d_j) : xcoord(s_j) > xcoord(d_j)$. Each type of crossing depends on the pair of edges (e_{under}, e_{over}) that contains the corresponding intersection point, and that is: (i) on the orientation of e_{under}, and e_{over}, i.e. whether they are oriented from left to right (positive) or from right to left (negative); (ii) on the comparison relation between the slope of e_{under} and the slope of e_{over}. Depending on these three parameters, we have 2^3 possible cases for deciding the type of crossings. For instance, we consider a crossing c determined by the pair of ordered edges $\big(-e_l(s_l, d_l), e_k(s_k, d_k)\big)$, for which $-e_l$ is the undergoing edge and e_k is the overgoing edge in $\mathbb{R}^3$. We have $xcoord(s_l) > xcoord(d_l)$ for the negative undergoing edge e_l, and $xcoord(s_k) < xcoord(d_k)$ for the positive overgoing edge e_k. If additionally we suppose that $slope(e_l) < slope(e_k)$, then c is a lefthanded crossing.
We give the schematic algorithm for the computation of the diagram $D(L)$ of a differentiable algebraic link L computed as in Subsection 3.2 and approximated by a piecewise linear algebraic link $Graph(L)$.

Algorithm 4 Diagram of piecewise linear links DIAGRAM($Graph(L)$)

Input: $Graph(L) = \langle \mathcal{P}, \mathcal{E} \rangle$ piecewise linear algebraic link which approximates
L a differentiable algebraic link as computed in Subsection3.2
$\mathcal{P}$ set of points with their euclidean coordinates
$\mathcal{E}$ set of edges connecting them
Output: $D(L)$
where $D(L)$ is the diagram of $Graph(L) \cong_{isotopic} L$.

1. Compute the crossings of $D(L)$;
   (a) Compute I the intersections of the edges of E;
   (b) Compute E_I the pairs of ordered edges containing each intersection;
2. Compute $K_i, i \in \mathbb{N}$ the lists of edges from E for all the knots of $D(L)$;
3. Compute the arcs of $D(L)$ and the type of crossings in $D(L)$.

3.4 Computing the delta-Invariant of an Isolated Singularity

We use Milnor results for computing the δ-invariant of the isolated singularity $(0, 0)$. Following [26]:

- we consider μ a positive integer that measures the amount of degeneracy at the critical point $(0, 0)$ of the complex polynomial $F(z, w)$. In fact, μ

is the Milnor number. It is shown that μ is the degree of the characteristic polynomial Δ of the link $L = V \cap S_\epsilon$ determined by $V = F^{-1}(0)$. The characteristic polynomial Δ coincides with the Alexander polynomial $\Delta_L(t)$ if L has one knot component, and $\Delta = \frac{(t-1)}{\pm t_i}\Delta_L(t, ..., t)$ if L has more than one knot components. We observe that μ is the degree of the characteristic polynomial Δ. Based on this observation we deduce that μ is the degree of the Alexander polynomial if L has one knot component, and μ is the degree of the Alexander polynomial $+1$ if L has more than one knot components.

- we consider r the number of local analytic branches of $V = F^{-1}(0)$ with $L = V \cap S_\epsilon$ passing through origin. That is r is the number of knot components in the link L determined by V, i.e. r is the number of variables in the Alexander polynomial of the link L.

We base our algorithm for the computation of the δ-invariant on the following theorem proved by Milnor:

Theorem 3. *Suppose that r branches of the curve $V = F^{-1}(0)$ pass through the origin $s = (0, 0)$, which is an isolated singularity for V. Then the delta-invariant of the isolated singularity $s = (0, 0)$ denoted with δ_s is related to the Milnor number μ by the equation $2\delta_s = \mu + r - 1$ ([26]). It is always an integer.*

We give the schematic algorithm for the computation of the δ-invariant of the isolated singularity $(0, 0)$.

Algorithm 5 Delta-invariant of the isolated singularity $(0, 0)$ DELTA(Δ_L, μ, r)

Input: $\Delta_L(t_1, ..., t_m)$ the Alexander polynomial of L
L the algebraic link of the isolated singularity $s = (0, 0)$,
d the degree of Δ_L, m the number of variables in Δ_L

Output: $\delta_s \in \mathbb{Z}^*_+$

where δ_s is the delta-invariant of $s = (0, 0)$.

1. If $m = 1$ then $\delta_s = \frac{d}{2}$
2. If $m \geq 2$ then $\delta_s = \frac{d+m}{2}$

3.5 Computing the Genus of the Algebraic Curve

We now give the schematic algorithm for computing the genus of a plane complex algebraic curve whose defining polynomial has numeric coefficients. The computed genus is the approximate genus, which is defined as the lowest possible genus of a curve defined by a nearby polynomial. We discuss the notion of approximate genus in detail in Section 4.

Algorithm 6 Genus of a plane complex algebraic curve GENUS(F, d, ϵ)

Input: $C = \{(z, w) \in \mathbb{C}^2 | F(z, w) = 0\}$,
$F(z, w) \in \mathbb{C}[z, w]$ with numeric coefficients,
d the degree of C, $\epsilon \in \mathbb{R}^*, \epsilon > 0$
Output: $genus(C) \in \mathbb{Z}$
where $genus(C)$ is the approximate genus of C.

1. $sumDeltaInv = 0$;
2. Compute $Sing(C) = \text{SING}(F, d)$;
3. For each $s_i = (z_i, w_i) \in Sing(C)$ do:
   (a) Move s_i in $(0, 0) : C = \{(z + z_i, w + w_i) \in \mathbb{C}^2 | F(z + z_i, w + w_i) = 0\}$
   (b) Compute $L = \text{ALGLINK}(F, \epsilon)$ (L is approximated by $Graph(L)$);
   (c) Compute $D(L) = \text{DIAGRAMLINK}(Graph(L))$;
   (d) Compute $\Delta_L(t_1, ..., t_m) = \text{ALEXPOLY}(D(L), m, n)$;
   (e) Compute $\delta_{s_i} = \text{DELTA}(\Delta_L, \mu, r)$;
   (f) $sumDeltaInv = sumDeltaInv + \delta_{s_i}$;
4. $genus(C) = \dfrac{(d-1)(d-2)}{2} - sumDeltaInv.$

4 What precisely means "approximate computation"?

It is the mark of an instructed mind to rest satisfied with the degree of precision to which the nature of the subject admits and not to seek exactness when only an approximation of the truth is possible.

Aristotle

In this section we introduce a theory for describing approximate algorithms for computing discrete information from continuous data. We do not claim originality for this theorem, as it is based on regularization theory (see [13]) and general principles that are commonly used in approximate algebraic computation (see [36, 41]). For a probabilistic foundation of the theory, we refer to [30].
In the second part of the section, we apply the theory to our problem of computing Alexander polynomials for equations with numeric coefficients.

Let D be a metric space. For any $x \in D$ and $\epsilon > 0$, we define the open ball $B_\epsilon(x) := \{y \in D \mid \text{dist}(x, y) < \epsilon\}$. Let V be a partially ordered set. Let $f : D \to V$ be an upper semicontinuous function, with respect to the discrete topology on V. A *regularization* of f is a function $F : (D \times \mathbb{R}_+) \to V$ such

that there exists a pair of continuous and bijective and increasing functions $d_1, d_2 : \mathbb{R}_+ \to \mathbb{R}_+$ such that

$$\forall x \in D\ \forall \epsilon > 0\ \forall y \in B_{d_1(\epsilon)}(x) : f(y) \leq F(x, \epsilon), \tag{3}$$

$$\forall x \in D\ \forall \epsilon > 0\ \exists y \in B_{d_2(\epsilon)}(x) : f(y) = F(x, \epsilon). \tag{4}$$

We say that the function $F_\epsilon : D \to V : x \mapsto F(x, \epsilon)$ "approximates f at level ϵ". The functions d_1 and d_2 are called "upper and lower error bound". The quotient d_2/d_1 is a kind of quality measure for the regularization.

Example 1. The simplest nontrivial upper semicontinuous function is the zero test function $z : \mathbb{R} \to \mathbb{Z}$ defined by $z(0) = 1$ and $z(x) = 0$ for $x \neq 0$.
A possible regularization is $F_\epsilon(x) = 1$ for $|x| < \epsilon$ and $F_\epsilon(x) = 0$ otherwise. The error bounds are $d_1(\epsilon) = d_2(\epsilon) = \epsilon$, and this is the best possible ratio of upper and lower error bound.

For any upper semicontinuous functions, it is possible (and actually not very difficult) to define a regularization with $d_1 = d_2$. However, if the function is more complicated, then it is computationally much cheaper to compute a regularization such that $d_1 < d_2$.
For all regularizations of f, the approximations converge pointwise to f, and they are eventually continuous at any fixed input x. Here is the precise statement:

Theorem 1. *Let F be a regularization of f.*
a) For any $x \in D$, we have $\lim_{\epsilon \to 0} F_\epsilon(x) = f(x)$.
b) For any $x \in D$, there exists $\epsilon > 0$ such that for any $\epsilon' < \epsilon$, $F_{\epsilon'}$ is constant in some neighborhood of x.

Proof. Let $x \in D$ be arbitrary. We define

$$\beta := \sup\{\alpha \mid \forall y \in B_\alpha(x) : f(y) \leq f(x)\}.$$

Then we claim that $F_\epsilon(x) = f(x)$ for all $\epsilon < d_2^{-1}(\beta)$. Indeed, (3) implies $f(x) \leq F_\epsilon(x)$ (for any ϵ), and (4) implies the existence of $y \in B_{d_2(\epsilon)}(x) \subseteq B_\beta(x)$ such that $F_\epsilon(x) = f(y) \leq f(x)$. This shows (a).
For any x, we choose $\epsilon := d_2^{-1}\left(\frac{\beta}{2}\right)$. Then for any $\epsilon' < \epsilon$, we choose δ as the minimum of $\frac{\beta}{2}$ and $d_1(\epsilon')$. Then we claim that $F_{\epsilon'}(y) = f(x)$ for all $y \in B_\delta(x)$, which shows (b).
By (3), we obtain $f(x) \leq F_{\epsilon'}(y)$. We assume, indirectly, that $F_{\epsilon'}(y) > f(x)$. By (4), there exists $z \in B_{d_2(\epsilon')}(y)$ such that $f(z) = F_{\epsilon'}(y) > f(x)$. It follows that

$$\operatorname{dist}(x, z) \leq \operatorname{dist}(x, y) + \operatorname{dist}(y, z) < \delta + d_2(\epsilon') \leq \frac{\beta}{2} + \frac{\beta}{2} = \beta,$$

and this is a contradiction to the definition of β. □

The Alexander polynomial of a given equation can be formulated as an upper semicontinuous function in the following way. First, we fix a finite set $E \subset \mathbb{Z}^2$. Then the set of polynomials with exponents in E form a finite-dimensional linear space. We take D as the subset of all polynomials with norm 1. Second, the set V is the set of classes multivariate Laurent polynomials over the integers,

where the equivalence relation is generated by multiplying with monomials and substitution of variables by their multiplicative inverses. In order to introduce a partial order on V which makes the Alexander polynomial an upper semicontinuous function, we use a well-known theorem from singularity theory.

Theorem 2. *1) The Milnor number is an upper semicontinuous function of the coefficients.*
2) In every connected subset in coefficient space where the Milnor number is constant, the topological type is also constant.

Proof. For the proof of the first assumption the reader can consult [37] and for proof of the second one [29]. □

Remark 3. The δ-invariant is also upper semicontinuous, but it does not satisfy statement (2) in the above theorem. For example, there exist a sequence of nodes degenerating to a cusp, but both the cusp and the node have both δ-invariant equal to 1.

The Alexander polynomial determines μ, the Milnor number, by: μ is the degree of the Alexander polynomial if L has one knot component, and μ is the degree of the Alexander polynomial plus 1 if L has more than one knot components as deduced in Subsection 3.4. We denoted with L the algebraic link of a singularity as described in Subsection 3.2. We define now the partial order on V as: $P_1 < P_2$ if and only the Milnor number of P_1 is less than the Milnor number of P_2 .
We think that the algorithm described in Subsection 3.3 computes a regularization of the Alexander polynomial. The precise proof and the analysis of the error bounds is still under construction.

5 Numerical Experiments

> *There is no such thing as a failed experiment, only experiments with unexpected outcomes.*
>
> R. B. Fuller

In this paper, we will give some experimental evidence for the statement that our algorithm satisfies the conclusion of Theorem 1. All the experiments, numerical and symbolical, are done with the software, *GENOM3CK*-Symbolic numeric techniques for *GEN*us c*OM*putation of *C*omplex algebrai*C* *C*urves using *K*not theory. GENOM3CK is implemented and included as a library in the free system Axel [39], written in C++ with Qt Script for Applications (QSA).
As evidences for the **convergency property** we consider an input polynomial $F(x, y) \in \mathbb{C}[x, y]$ with both exact and inexact coefficients and we compute $A_\epsilon(F(x, y))$ with the approximate algorithm A_ϵ. We compute $A_\epsilon(F(x, y))$ with the approximate algorithm for different values of the parameter ϵ. We obtain several outputs such as: the singularities of the input curve defined by $F(x, y)$, the algebraic link of each singularity (i.e. the topology of the singularity), the Alexander polynomial of each algebraic link, the delta-invariant of each singularity, and the genus of the curve. The computation of the Alexander polynomial, delta-invariant and the genus depends on the computation of the algebraic link

of each singularity. From the experiments, we observe that the approximate solution computed with A_ϵ converges to the exact solution as ϵ tends to 0.

Example 1. We consider $F(x,y) = x^2 - x*y - y^3$. We notice that $x^2 - x*y = x(x-y)$ thus $F(x,y)$ has a vertical tangent $x=0$ in $\mathbb{C}^4$. In order to assure a valid stereographic projection in $\mathbb{R}^3$ we make the substitution $\{x \to y, y \to x\}$ in $F(x,y)$ obtaining the polynomial $F(x,y) = -x^3 - x*y + y^2$, and thus we consider this polynomial as the input of the problem. We use Arnold's results concerning the analysis of curve singularities and we deduce that the algebraic link of the singularity $(0,0)$ of the polynomial $x^2 - x*y - y^3$ is the same as the algebraic link of the singularity $(0,0)$ of the polynomial $x^2 - x*y$ which is the Hopf link, and which represents the exact solution for the algebraic link of the singularity $(0,0)$ of $F(x,y)$. We notice that the approximate solution converges to the exact solution as ϵ tends to 0.

Equation and ϵ		**Link**	**Alexander, δ invariants, genus**		
$-x^3 - xy + y^2$	1.00	Trefoil knot	$\Delta(t_1) = t_1^2 - t_1 + 1$	$\delta = 1$	$g = 0$
$-x^3 - xy + y^2$	0.5	Trefoil knot	$\Delta(t_1) = t_1^2 - t_1 + 1$	$\delta = 1$	$g = 0$
$-x^3 - xy + y^2$	0.25	Hopf link	$\Delta(t_1, t_2) = 1$	$\delta = 1$	$g = 0$
$-x^3 - xy + y^2$	0.14	Hopf link	$\Delta(t_1, t_2) = 1$	$\delta = 1$	$g = 0$

We can consider the input polynomial with both exact and inexact coefficients, such as $F(x,y) = -x^3 - x*y + y^2 - 0.01$. We observe again that the approximate solution converges to the exact solution when ϵ tends to 0.

Equation and ϵ		**Link**	**Alexander, δ invariants, genus**		
$-x^3 - xy + y^2 - 0.01$	1.00	Trefoil knot	$\Delta(t_1) = t_1^2 - t_1 + 1$	$\delta = 1$	$g = 0$
$-x^3 - xy + y^2 - 0.01$	0.5	Hopf link	$\Delta(t_1, t_2) = 1$	$\delta = 1$	$g = 0$
$-x^3 - xy + y^2 - 0.01$	0.25	Hopf link	$\Delta(t_1, t_2) = 1$	$\delta = 1$	$g = 0$
$-x^3 - xy + y^2 - 0.01$	0.22	Hopf link	$\Delta(t_1, t_2) = 1$	$\delta = 1$	$g = 0$

Example 2. We consider $F(x,y) = x^2 - y^2 - y^3$. We use Arnold's results concerning the analysis of curve singularities and we deduce that the algebraic link of the singularity $(0,0)$ of $F(x,y)$ is the same as the algebraic link of the singularity $(0,0)$ of the polynomial $x^2 - y^2$ which is the Hopf link, and which represent the exact solution for the algebraic link of the singularity $(0,0)$ of $F(x,y)$. We notice that the approximate solution converges to the exact solution as ϵ tends to 0.

Equation and ϵ		Link	Alexander, δ invariants, genus		
$x^2-y^2-y^3$	1.00	1 singularity curve	–	–	–
$x^2-y^2-y^3$	0.7	Hopf link	$\Delta(t_1,t_2)=1$	$\delta=1$	$g=0$
$x^2-y^2-y^3$	0.5	Hopf link	$\Delta(t_1,t_2)=1$	$\delta=1$	$g=0$
$x^2-y^2-y^3$	0.19	Hopf link	$\Delta(t_1,t_2)=1$	$\delta=1$	$g=0$

As evidences for the **continuity property** we consider an input curve defined by the polynomial $F(x,y) \in \mathbb{C}[x,y]$ with exact and inexact coefficients and we compute $A_\epsilon(F(x,y))$ with the approximate algorithm A_ϵ. The continuity property of A_ϵ states that small changes in the input polynomial $F(x,y)$ produce constant output for the computed solution. To observe this we proceed in the following way:

- we consider a polynomial $p(x,y) \in \mathbb{C}[x,y]$ which contains only exact coefficients;
- for $\sigma \in \mathbb{R}^*$, we slightly perturbed the coefficients of the polynomial $p(x,y)$ obtaining some new polynomials denoted with $p_\sigma(x,y)$ that we call perturbations of the polynomial $p(x,y)$. We call σ the perturbation of the exact polynomial $p(x,y)$.
- we consider several values for the parameter ϵ. For each of these values, we execute the approximate algorithm A_ϵ on the perturbed polynomials $p_\sigma(x,y)$ for different values of $\sigma \in \mathbb{R}^*$. The perturbed polynomials $p_\sigma(x,y)$ represent the input polynomials $F(x,y)$ with exact and inexact coefficients, i.e. $F(x,y)=p_\sigma(x,y)$, for $\sigma \in \mathbb{R}^*$.

We distinguish between two types of perturbations:

1. Perturbations of type I: For these types of perturbations, $p_\sigma(x,y)$ is of the following form: $p_\sigma(x,y)=p(x,y)+\sigma$, where $p(x,y)$ is the exact polynomial and $\sigma \in \mathbb{R}^*$ is a real number different from 0.
2. Perturbations of type II: For these types of perturbations, $p_\sigma(x,y)$ is of the following form: $p_\sigma(x,y)=p(x,y)+\sigma q(x,y)$, where $p(x,y)$ is the exact polynomial, $\sigma \in \mathbb{R}^*$ and $q(x,y) \in \mathbb{C}[x,y]$ is an arbitrary exact polynomial.

From the experiments, we observe that for the perturbed polynomials the approximate computed solution is preserved, that is for small changes of the input data we obtain constant output for the computed solution.

Example 1. For the exact polynomial $p(x,y)=-x^3-x*y+y^2$, we consider perturbations on type I of the form $p_\sigma(x,y)=-x^3-x*y+y^2-\sigma$, with $\sigma \in \{10^{-2},...,10^{-10}\}$.

Perturbations I and ϵ		$\sigma = 10^{-e}, e \in \mathbb{N}^*$	**Link**	**Invariants**
$-x^3-xy+y^2-10^{-e}$	0.5	$\{10^{-2}, ..., 10^{-10}\}$	Trefoil knot	$\Delta(t_1) = t_1^2-t_1+1$ $\delta = 1$ $g = 0$
$-x^3-xy+y^2-10^{-e}$	0.25	$\{10^{-2}, ..., 10^{-10}\}$	Hopf link	$\Delta(t_1, t_2) = 1$ $\delta = 1$ $g = 0$

For the perturbations of type II we consider the exact polynomial $p(x, y) = -x^3 - xy + y^2$, the arbitrary exact polynomial $q(x, y) = -x^3 - 2xy + y^2$ and $\sigma \in \{10^{-1}, ..., 10^{-10}\}$, obtaining the perturbed polynomials $p_\sigma(x, y) = p(x, y) + \sigma q(x, y) = -x^3 - xy + y^2 + \sigma(-x^3 - 2xy + y^2) = -(1 + \sigma)x^3 - (1 + 2\sigma)xy + (1 + \sigma)y^2$. For $\sigma = 0.1$ we obtain the perturbed polynomial $p_{\sigma\leftarrow 0.1} = -1.1x^3 - 1.2xy + 1.1y^2$; for $\sigma = 0.01$ we obtain the perturbed polynomial $p_{\sigma\leftarrow 0.01} = -1.01x^3 - 1.02xy + 1.01y^2$; for $\sigma = 0.001$ we obtain the perturbed polynomial $p_{\sigma\leftarrow 0.001} = -1.001x^3 - 1.002xy + 1.001y^2$, etc.

Perturbations II and ϵ		$\sigma = 10^{-e}, e \in \mathbb{N}^*$	**Link**	**Invariants**
$-(1+10^{-e})x^3-(1+2\cdot 10^{-e})xy+(1+10^{-e})y^2$	0.15	$\{10^{-1}, ...10^{-10}\}$	Hopf link	$\Delta(t_1, t_2) = 1$ $\delta = 1$ $g = 0$
$-(1+10^{-e})x^3-(1+2\cdot 10^{-e})xy+(1+10^{-e})y^2$	0.14	$\{10^{-1}, ...10^{-10}\}$	Hopf link	$\Delta(t_1, t_2) = 1$ $\delta = 1$ $g = 0$

Example 2. For the exact polynomial $p(x, y) = x^2 - y^2 - y^3$, we consider perturbations on type I of the form $p_\sigma(x, y) = x^2 - y^2 - y^3 - \sigma$, with $\sigma \in \{10^{-1}, ..., 10^{-10}\}$.

Perturbations I and ϵ		$\sigma = 10^{-e}, e \in \mathbb{N}^*$	**Link**	**Invariants**
$x^2 - y^2 - y^3 - 10^{-e}$	0.5	$\{10^{-1}, ..., 10^{-10}\}$	Hopf link	$\Delta(t_1, t_2) = 1$ $\delta = 1$ $g = 0$
$x^2 - y^2 - y^3 - 10^{-e}$	0.14	$\{10^{-1}, ..., 10^{-10}\}$	Hopf link	$\Delta(t_1, t_2) = 1$ $\delta = 1$ $g = 0$

For the perturbations of type II we consider the exact polynomial $p(x, y) = x^2 - y^2 - y^3$, the arbitrary exact polynomial $q(x, y) = x^2 - 3y^2 - 4y^3$ and $\sigma \in \{10^{-1}, ..., 10^{-10}\}$, obtaining the perturbed polynomials $p_\sigma(x, y) = p(x, y) + \sigma q(x, y) = x^2 - y^2 - y^3 + \sigma(x^2 - 3y^2 - 4y^3) = (1+\sigma)x^2 - (1+3\sigma)y^2 - (1+4\sigma)y^3$. For $\sigma = 0.1$ we obtain the perturbed polynomial $p_{\sigma\leftarrow 0.1} = 1.1x^2 - 1.3y^2 - 1.4y^3$; for $\sigma = 0.01$ we obtain the perturbed polynomial $p_{\sigma\leftarrow 0.01} = 1.01x^2 - 1.03y^2 - 1.04y^3$; for $\sigma = 0.001$ we obtain the perturbed polynomial $p_{\sigma\leftarrow 0.001} = 1.001x^2 - 1.003y^2 - 1.004y^3$, etc.

Perturbations II and ϵ		$\sigma = 10^{-e}, e \in \mathbb{N}^*$	**Link**	**Invariants**
$(1+10^{-e})x^2-(1+3\cdot 10^{-e})y^2-(1+4\cdot 10^{-e})y^3$	0.25	$\{10^{-1}, ...10^{-10}\}$	Hopf link	$\Delta(t_1, t_2) = 1$ $\delta = 1$ $g = 0$
$(1+10^{-e})x^2-(1+3\cdot 10^{-e})y^2-(1+4\cdot 10^{-e})y^3$	0.14	$\{10^{-1}, ...10^{-10}\}$	Hopf link	$\Delta(t_1, t_2) = 1$ $\delta = 1$ $g = 0$

6 Conclusion and Future Work

If I have seen further than others,
it is by standing upon the
shoulders of giants.

Isaac Newton

For each input plane complex algebraic curve C defined by the polynomial $F(z, w)$ with numeric coefficients, GENOM3CK performs the following computational operations: (i) it computes the set of all distinct real singularities of C; (ii) it computes and visualize the algebraic link L of each singularity of the input curve C in the three-dimensional space; for each algebraic link L, which is a smooth, implicitly defined closed curve in $\mathbb{R}^3$, it computes and visualize the two implicit surfaces that define the algebraic link L. In fact these surfaces represent the Milnor fibration; (iii) it computes the diagram of each algebraic link L; (iv) it computes the Alexander polynomial of each algebraic link L; (v) it computes the δ-invariant of each singularity; (vi) it computes the genus of the curve C; (vii) it also computes the time needed for performing each of these operations.
We have reported on a symbolic-numeric algorithm for genus computation of plane complex algebraic curves whose defining polynomials have coefficients of limited accuracy, i.e the coefficients of the polynomial are both exact and inexact data. We have successfully realized a complete automatization for the steps of the proposed symbolic-numeric algorithm in the GENOM3CK library using Axel, an algebraic geometric modeller. The library allows us to compute several invariants of an input plane complex algebraic curve, such as: the algebraic link, the Alexander polynomial and the delta-invariant of each singularity of the curve. In addition, the library allows us to analyse the performace of the proposed symbolic-numeric algorithm. The test experiments indicate the efficiency of the proposed symbolic-numeric algorithm. Moreover, we use the library to offer practical evidences for the convergency and the continuity properties of the proposed symbolic-numeric algorithm. These tests also indicate that the proposed symbolic numeric algorithm can be described using priciples from regularization theory and approximate algebraic computation. Using these principles, we intend to give a precise meaning to the notion of approximate genus of the input plane complex algebraic curve computed also using the proposed symbolic-numeric algorithm.

References

[1] L. Alberti and B. Mourrain. Regularity criteria for the topology of algebraic curves and surfaces. In *Proc. IMA Conference of the Mathematics of Surfaces*, pages 1–28, 2007.

[2] L. Alberti and B. Mourrain. Visualization of implicit algebraic curves. In *Proc. 15th Pacific Conference on Computer Graphics and Applications*, pages 303–312, 2007.

[3] J. W. Alexander. Topological invariant of knots and links. *Transactions of the American Mathematical Society*, 30:275–306, 1928.

[4] S. Béla and B. Jüttler. Fat arcs for implicitly defined curves. In *Proc. MMCS 2009*, page to appear, 2009.

[5] K. Brauner. Zur geometrie der funktionen zweier komplexer veränderlichen. *Abh. Math. Sem. Hamburg*, 6:1–54, 1928.

[6] E. Brieskorn and H. Knorrer. *Plane algebraic curves.* Birkhäuser, Berlin, 1986.

[7] L. Busé, H. Khalil, and B. Mourrain. Resultant-based methods for plane curves intersection problems. In *Proc. CASC 2005*, volume 3718, pages 75–92, 2005.

[8] D. Cimasoni. Studying the multivariable alexander polynomial by means of seifert surfaces. *Bol. Soc. Mat. Mexicana (3)*, 10:107–115, 2004.

[9] C. A. Colin. *The knot book. An elementary introduction to the mathematical theory of knots.* W. H. Freeman and Company, U.S.A., 2004.

[10] R. M. Corless, S. M. Watt, and L. Zhi. Qr factoring to compute the gcd of univariate approximate polynomials. *IEEE Trans. Signal Process.*, 52:3394–3402, 2004.

[11] B. H. Dayton and Z. Zeng. The approximate gcd of inexact polynomials. part ii: A multivariate algorithm. In *Proc. 2004 Internat. Symp. Symbolic Algebraic Comput*, pages 320–327, 2004.

[12] M. de Berg, M. Krefeld, M. Overmars, and O. Schwarzkopf. *Computational geometry: algorithms and applications. Second edition.* Springer, Berlin, 2008.

[13] H. W. Engl, M. Hanke, and A. Neubauer. *Regularization of inverse problems.* Kluwer Academic Publishers Group, 1996.

[14] W. Fulton. *Algebraic curves-An introduction to algebraic geometry.* Addison-Wesley, Redwood City California, 1989.

[15] G. M. Greuel and G. Pfister. *A Singular introduction to commutative algebra.* Springer-Verlag Berlin Heidelberg, 2002.

[16] J. Gutierrez, R. Rubio, and J. Schicho. Polynomial parametrization of curves without affine singularities. *Computer Aided Geometric Design*, 19:223–234, 2002.

[17] G. Haché. Example of axiom package paff. http://axiom-wiki.newsynthesis.org/PAFF.

[18] F. Hess. Computing riemann-roch spaces in algebraic function fields and related topics. *Symbolic Computation*, 33:425–445, 2002.

[19] F. Hess. Generalising the ghs attack on the elliptic curve discrete logarithm. *LMS of Computation and Mathematics*, 7:167–192, 2004.

[20] M. Hodorog and J. Schicho. Computational geometry and combinatorial algorithms for the genus computation problem. Technical report, RICAM, Linz, 2010. to appear.

[21] M. V. Hoeij. The algcurves package in maple. http://www.math.fsu.edu/ hoeij/compalg/algcurves.html.

[22] V. D. J. Hoeven, G. Lecerf, and B. Mourrain. Mathemagix computer algebra system. http://www.mathemagix.org/www/main/index.en.html.

[23] C. Liang, B. Mourrain, and J.P. Pavone. *Subdivision methods for 2d and 3d implicit curves*, chapter 11, pages 199–214. Springer, geometric modeling and algebraic geometry (eds. jüttler b. piene r.) edition, August 2008.

[24] C. Livingston. *Knot theory*. Mathematical Association of America, U.S.A, 1993.

[25] A. Mantzaflaris, B. Mourrain, and E. Tsigaridas. Continued fraction expansion of real roots of polynomial systems. In *Proc. 2009 SNC Conference on Symbolic-Numeric Computation*, pages 85–94, 2009.

[26] J. Milnor. *Singular points of complex hypersurfaces*. Princeton University Press and the University of Tokyo Press, New Jersey, 1968.

[27] M. Mnuk and F. Winkler. Casa - a system for computer aided constructive algebraic geometry. In *Proc. International Symposium on Design and Implementation of Symbolic Computation Systems*, pages 297–307, 1996.

[28] B. Mourrain and J.P. Pavone. Subdivision methods for solving polynomial equations. Technical Report 5658, INRIA, Sophia-Antipolis, 2005. to appear.

[29] Lê Dŭng Tráng and C. P. Ramanujam. The invariance of Milnor's number implies the invariance of the topological type. *Amer. J. Math.*, 98:67–78, 1976.

[30] H. K. Pikkarainen and J. Schicho. A Bayesian model for root computation. *Math. in Comp. Sci.*, 2:567–586, 2009.

[31] J. R. Sendra and F. Winkler. Parametrization of algebraic curves over optimal field extensions. *Symbolic Computation*, 23:191–208, 1997.

[32] J. R. Sendra, F. Winkler, and S.P. Diaz. *Rational algebraic curves. A computer algebra approach*. Springer-Verlag Berlin Heidelberg, Germany, 2008.

[33] G. Shuhong, E. Kaltofen, J. May, Z. Yang, and L. Zhi. Approximate factorization of multivariate polynomials via differential equations. *Symbolic Computation*, 43:359–376, 2008.

[34] A. J. Sommese and C. W. Wampler. Numerical algebraic geometry. *Lectures in Applied Mathematics*, 32:749–763, 1996.

[35] A. J. Sommese and C. W. Wampler. *The numerical solution of systems of polynomials arising in engineering and science*. World Scientific Publishing, Singapore, 2005.

[36] H. J. Stetter. *Numerical polynomial algebra*. SIAM, Philadelphia, 2004.

[37] J. C. Tougeron. *Ideaux de fonctions differentiables*. Springer-Verlag, Berlin, 1972.

[38] R. J. Walker. *Algebraic curves*. Springer-Verlag New York, U.S.A., 1978.

[39] J. Wintz. *Algebraic methods for geometric modelling*. PhD thesis, University of Nice, Sophia-Antipolis, 2008.

[40] M. Yamamoto. Classification of isolated algebraic singularities by their alexander polynomials. *Topology*, 23:277–287, 1984.

[41] Z. Zeng. Computing multiple roots of inexact polynomials. *Math. Comp.*, 74:869–903, 2005.

Technical Reports of the Doctoral Program "Computational Mathematics"

2010

2010-01 S. Radu, J. Sellers: *Parity Results for Broken k-diamond Partitions and (2k+1)-cores* March 2010. Eds.: P. Paule, V. Pillwein

2010-02 P.G. Gruber: *Adaptive Strategies for High Order FEM in Elastoplasticity* March 2010. Eds.: U. Langer, V. Pillwein

2010-03 Y. Huang, L.X.Châu Ngô: *Rational General Solutions of High Order Non-autonomous ODEs* June 2010. Eds.: F. Winkler, P. Paule

2010-04 S. Beuchler, V. Pillwein, S. Zaglmayr: *Sparsity optimized high order finite element functions for H(div) on simplices* September 2010. Eds.: U. Langer, P. Paule

2010-05 C. Hofreither, U. Langer, C. Pechstein: *Analysis of a non-standard finite element method based on boundary integral operators* September 2010. Eds.: B. Jüttler, J. Schicho

2010-06 M. Hodorog, J. Schicho: *A symbolic-numeric algorithm for genus computation* September 2010. Eds.: B. Jüttler, R. Ramlau

2009

2009-01 S. Takacs, W. Zulehner: *Multigrid Methods for Elliptic Optimal Control Problems with Neumann Boundary Control* October 2009. Eds.: U. Langer, J. Schicho

2009-02 P. Paule, S. Radu: *A Proof of Sellers' Conjecture* October 2009. Eds.: V. Pillwein, F. Winkler

2009-03 K. Kohl, F. Stan: *An Algorithmic Approach to the Mellin Transform Method* November 2009. Eds.: P. Paule, V. Pillwein

2009-04 L.X.Chau Ngo: *Rational general solutions of first order non-autonomous parametric ODEs* November 2009. Eds.: F. Winkler, P. Paule

2009-05 L.X.Chau Ngo: *A criterion for existence of rational general solutions of planar systems of ODEs* November 2009. Eds.: F. Winkler, P. Paule

2009-06 M. Bartoň, B. Jüttler, W. Wang: *Construction of Rational Curves with Rational Rotation-Minimizing Frames via MöbiusTransformations* November 2009. Eds.: J. Schicho, W. Zulehner

2009-07 M. Aigner, C. Heinrich, B. Jüttler, E. Pilgerstorfer, B. Simeon, A.V. Vuong: *Swept Volume Parameterization for Isogeometric Analysis* November 2009. Eds.: J. Schicho, W. Zulehner

2009-08 S. Béla, B. Jüttler: *Fat arcs for implicitly defined curves* November 2009. Eds.: J. Schicho, W. Zulehner

2009-09 M. Aigner, B. Jüttler: *Distance Regression by Gauss–Newton–type Methods and Iteratively Re–weighted Least–Squares* December 2009. Eds.: J. Schicho, W. Zulehner

2009-10 P. Paule, S. Radu: *Infinite Families of Strange Partition Congruences for Broken 2-diamonds* December 2009. Eds.: J. Schicho, V. Pillwein

2009-11 C. Pechstein: *Shape-explicit constants for some boundary integral operators* December 2009. Eds.: U. Langer, V. Pillwein

2009-12 P. Gruber, J. Kienesberger, U. Langer, J. Schöberl, J. Valdman: *Fast solvers and a posteriori error estimates in elastoplasticity* December 2009. Eds.: B. Jüttler, P. Paule

2009-13 P.G. Gruber, D. Knees, S. Nesenenko, M. Thomas: *Analytical and Numerical Aspects of Time-Dependent Models with Internal Variables* December 2009. Eds.: U. Langer, V. Pillwein

Doctoral Program
"Computational Mathematics"

Director:

Prof. Dr. Peter Paule
Research Institute for Symbolic Computation

Deputy Director:

Prof. Dr. Bert Jüttler
Institute of Applied Geometry

Address:

Johannes Kepler University Linz
Doctoral Program "Computational Mathematics"
Altenbergerstr. 69
A-4040 Linz
Austria
Tel.: ++43 732-2468-7174

E-Mail:

office@dk-compmath.jku.at

Homepage:

http://www.dk-compmath.jku.at

Submissions to the DK-Report Series are sent to two members of the Editorial Board who communicate their decision to the Managing Editor.