JOHANNES KEPLER
UNIVERSITÄT LINZ | JKU

TNF

Technisch-Naturwissenschaftliche
Fakultät

Symbolic-Numeric Algorithms for Plane Algebraic Curves

DISSERTATION

zur Erlangung des akademischen Grades

Doktorin

im Doktoratsstudium der

Technischen Wissenschaften

Eingereicht von:
M.Sc. Mădălina Hodorog

Angefertigt am:
Doctoral Program "Computational Mathematics"
Johann Radon Institute for Computational and Applied Mathematics
Research Institute for Symbolic Computation

Beurteilung:
A. Univ. Prof. Dr. Josef Schicho (Betreuung)
Prof. Dr. Bernard Mourrain

Linz, Oktober, 2011

Johannes Kepler University Linz
Doctoral Program "Computational Mathematics"
Johann Radon Institute for Computational and Applied Mathematics
Austrian Academy of Sciences
Research Institute for Symbolic Computation Linz

Symbolic-Numeric Algorithms for Plane Algebraic Curves

Doctoral Thesis

M.Sc. Mădălina Hodorog

Advised by
A. Univ. Prof. Dr. Josef Schicho
Prof. Dr. Bernard Mourrain

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Dissertation selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt bzw. die wörtlich oder sinngemäß entnommenen Stellen als solche kenntlich gemacht habe.

Linz, Oktober 2011 .....

Mădălina Hodorog

The author was funded by:

- the Upper Austrian Government (1/10/2007-30/09/2008),
- the Austrian Science Fund (FWF): W1214-N15, project DK9 (1/10/2008-present).

Zusammenfassung

In der Computeralgebra ist das Problem der Berechnung der topologischen Invarianten, d.h. Delta-Invariante und Geschlecht, einer ebenen komplexen algebraischen Kurve, gut verstanden, wenn die Kurve durch ein Polynom mit exakten Daten (d.h. ganzen Zahlen und rationalen Zahlen) definiert ist. Eine Herausforderung ist, dieses Problem zu lösen, wenn die Koeffizienten inexakte Daten sind (d.h. numerischen Werten).

In dieser Disseration behandeln wir das algebraische Problem der Berechnung der topologischen Invarianten einer ebenen komplexen algebraischen Kurve definiert durch ein Polynom mit sowohl exakten als auch inexakten Daten. Zu den inexakten Daten assoziieren wir eine positive reelle Zahl, welche Toleranz oder *Rauschen* genannt wird und die Größenordnung des Fehlers in den Koeffizienten angibt. Wir haben es mit einem *schlecht gestellten Problem* zu tun, was bedeutet, dass winzige Änderungen in den Eingangsdaten die Lösung des Problems dramatisch verändern.

Zur Behandlung dieses schlecht gestellten Problems stellen wir eine *Regularisierungsmethode* vor, welche die Invariante einer ebenen komplexen algebraischen Kurve schätzt. Unsere Regularisierungsmethode besteht aus einer Menge von *symbolisch-numerischen Algorithmen*, die strukturelle Informationen über die zu analysierende Kurve extrahiert, und aus einer *Parameterwahlregel*, in Abhängigkeit von der Größenordnung des Rauschens. Wir entwerfen zunächst zur Berechnung der Invarianten einer ebenen komplexen algebraischen Kurve symbolisch-numerische Algorithmen um damit

- die Verschlingung für jede Singularität der Kurve durch numerisches Lösen von Gleichungen zu berechnen;
- das Alexander-Polynom jeder Verschlingung durch das Benützen von Algorithmen aus der algorithmischen Geometrie (z.B. eine adaptierte Version des Bentley-Ottmann Algorithmus) und durch Verwenden von kombinatorischen Objekten aus der Knotentheorie zu berechnen;
- eine Formel für die Delta-Invariante und für das Geschlecht der Kurve abzuleiten.

Im weiteren beweisen wir, dass die symbolisch-numerischen Algorithmen zusammen mit der Parameterwahlregel Näherungslösungen berechnen, die die *Konvergenzeigenschaft für gestört Daten* erfüllen.

Die entworfenen symbolisch-numerischen Algorithmen sind in einem neuen Software-Paket, welches wir *Genom3ck* nennen, implementiert. Diese Software wurde mit dem freien algebraischen geometrischen System *Axel* und mit dem freien Computeralgebrasystem *Mathemagix* entwickelt. Für unsere Zwecke bieten diese beide Systeme sowohl moderne graphische Möglichkeiten als auch algebraische und geometrische Methoden für die Manipulation

algebraischer Kurven und Flächen, welche durch Polynome mit genauen und ungenauen Daten definiert sind. Das Software-Paket *Genom3ck* berechnet nicht nur das Geschlecht einer ebenen komplexen algebraischen Kurve, sondern berechnet auch andere nützliche Informationen, wie zum Beispiel die Singularitäten der Kurve in der projektiven Ebene und den topologischen Typ jeder Singularität.

Schlüsselwörter: Ebene Kurvensingularität, Näherungslösungen, schlecht gestelltes Problem, Regularisierungsmethode, symbolisch-numerische Algorithmen, Verschlingung einer Singularität, Alexander-Polynom, Delta-Invariante, Geschlecht.

Abstract

In computer algebra, the problem of computing topological invariants (i.e. delta-invariant, genus) of a plane complex algebraic curve is well-understood if the coefficients of the defining polynomial of the curve are exact data (i.e. integer numbers or rational numbers). The challenge is to handle this problem if the coefficients are inexact (i.e. numerical values).

In this thesis, we approach the algebraic problem of computing invariants of a plane complex algebraic curve defined by a polynomial with both exact and inexact data. For the inexact data, we associate a positive real number called tolerance or *noise*, which measures the error level in the coefficients. We deal with an *ill-posed* problem in the sense that, tiny changes in the input data lead to dramatic modifications in the output solution.

For handling the ill-posedness of the problem we present a *regularization* method, which estimates the invariants of a plane complex algebraic curve. Our regularization method consists of a set of *symbolic-numeric algorithms* that extract structural information on the input curve, and of *a parameter choice rule*, i.e. a function in the noise level. We first design the following symbolic-numeric algorithms for computing the invariants of a plane complex algebraic curve:

- we compute the link of each singularity of the curve by numerical equation solving;
- we compute the Alexander polynomial of each link by using algorithms from computational geometry (i.e. an adapted version of the Bentley-Ottmann algorithm) and combinatorial objects from knot theory;
- we derive a formula for the delta-invariant and for the genus.

We then prove that the symbolic-numeric algorithms together with the parameter choice rule compute approximate solutions, which satisfy the *convergence for noisy data property*. Moreover, we perform several numerical experiments, which support the validity for the convergence statement.

We implement the designed symbolic-numeric algorithms in a new software package called *Genom3ck*, developed using the *Axel* free algebraic geometric modeler and the *Mathemagix* free computer algebra system. For our purpose, both of these systems provide modern graphical capabilities, and algebraic and geometric tools for manipulating algebraic curves and surfaces defined by polynomials with both exact and inexact data. Together with its main functionality to compute the genus, the package *Genom3ck* computes also other type of information on a plane complex algebraic curve, such as the singularities of the curve in the projective plane and the topological type of each singularity.

Keywords: Plane curve singularity, approximate solutions, ill-posed problem, regular-

ization, symbolic-numeric algorithms, link of a singularity, Alexander polynomial, delta-invariant, genus.

To the memory of my grandparents,
Elena and Ionel Hodorog

Acknowledgements

A lot of people influenced and contributed to the realization of this thesis, which is the result of many years of study and research, hard work and ambition.

I would like to especially thank my advisor Josef Schicho for giving me the chance to be part of the Doctoral Program "Computational Mathematics" in the frame of DK9 and to be a member of the Symbolic Computation Group at RICAM. His valuable guidance contributed to the development of this thesis. I thank Josef for always finding the time to answer my questions and for paving my way in the field of computational algebraic geometry.

I would like to thank Bernard Mourrain for hosting my research visits to his research group at INRIA Sophia-Antipolis and for agreeing to be the second examiner of this thesis. His significant advice led to the advance of this thesis. I thank Bernard for teaching me various implementation issues concerning computer algebra systems and for being an example in combining advanced programming techniques with pure mathematics.

I would like to thank Peter Paule for setting up a well-organized and friendly environment in the Doctoral Program "Computational Mathematics". I thank Peter for his useful lectures on "Analytical combinatorics" and on "Algorithmic combinatorics".

I would like to thank Bruno Buchberger for his support to found the Research Institute e-Austria Timişoara. This led to a valuable connection between RISC and West University of Timişoara, which allowed me to continue my research in Linz. I thank Bruno for his creative comments on this work during the DK Seminar Meetings at Strobl.

I thank Tudor Jebelean for his help during my first year of PhD studies at RISC and for encouraging my application to the "Doctoral Program: Computational Mathematics".

I especially thank Adrian Crăciun. I learnt the basics of research from him. I thank Adi for spending the extra time in teaching me the art of writing a good research paper, for the long training hours in mathematical logic, and for supporting my research activity.

I would like to thank Viorel Negru and Dana Petcu from the West University of Timişoara, who supported my study and my research activity.

I want to thank Dan Bates and Chris Peterson for hosting my research visit to Colorado State University, Fort Collins. I thank Dan for initiating me in the field of numerical algebraic computation and for making my visit faraway from home a pleasant experience. I thank Chris for his interesting lecture on homology. His great teaching ability raised my interest in this intricate field.

I want to thank my colleagues from the "Doctoral Program: Computational Mathematics", especially to Szilvia Béla, Ismael Bleyer, Clemens Hofreither, Lam Xuan Chau Ngo,

Veronika Pillwein, Clemens Raab and Stefan Takacs, for their helpful discussions and for the friendly atmosphere they created. Many thanks to Gabriela Hahn, for the exemplary organization in handling the administrative formalities.

I thank my colleagues from the Symbolic Computation Group, at RICAM, especially to Gábor Hegedüs, Anja Korporal, Niels Lubbes, David Sevilla González, Georg Regensburger for their constant help. Special thanks to the secretaries and the system administrators.

I thank the colleagues from RISC. I thank Johannes Middeke for his practical suggestions concerning the creation of knot pictures in Latex. Most of the knot pictures from this thesis were produced using his suggestions! I thank Ralf Hemmecke for the lecture concerning the revision version control, which I extensively used for my work. Many thanks to Mădălina Eraşcu for being a valuable friend. Special thanks to Laura Kovács and Harald Wutzel for helping me in my first year in Linz.

I thank the members of the Galaad team from INRIA Sophia-Antipolis, especially to Angelos Mantzaflaris and Elias Tsigaridas. Many thanks to Julien Wintz for always replying my long emails concerning the Axel system and for initiating me in the Axel system.

I want to especially thank my friends, who encouraged me during the years. I thank Ioana Cismaş and Ioana Ban (former Goţia), for spending so many hours on solving mathematical problems and programming tasks during the undergraduate studies. Many thanks to Oana Medveşan for hosting my staying in Nice during my visits to INRIA Sophia-Antipolis and for never being tired to have another philosophical discussion. Other friends who constantly supported me are Alina Luputz and Aurelia Luputz. Many thanks to Carmen Puzovic (former Marin), who encouraged my mathematical abilities since high-school.

I am particularly grateful to my family for their constant support and for their visits to Linz, which I was always looking forward to. I thank my parents Gabriela and Doru Hodorog for their advice during the years. I also thank my aunt Loredana and my cousin Andrada for their warmness. Many thanks to Ioana Voina, Dorina, Petre and Cătălin Arapu for their hospitality. I want to specially thank my brother Bogdan Hodorog and his wife Luana for always looking out for me, for guiding my steps throughout the years and for hosting my many visits to Timişoara. My brother also influenced part of my analytical thinking and he introduced me to the basics of programming. I owe him my first compiled program in Pascal!

I would like to express my gratitude to Maria and Thomas Langer for showing me part of the Austrian way and culture and for making my staying in Linz a pleasant home-like experience.

Last but not least I want to especially thank my boyfriend Andreas Langer for his constant understanding and support, for his optimism, for always encouraging me and for his useful suggestions and hints on this thesis.

Contents

Chapter 1

Introduction

1.1 Motivation and Description of the Problem

In several situations from polynomial algebra, the input data (i.e. the coefficients) of the algebraic problems to be solved are only imperfectly known. More precisely, the problems are defined in terms of polynomials with coefficients of limited accuracy, this means that the coefficients are either exact data (i.e. integer or rational numbers) or inexact data (i.e. numerical values). In the latter case, the coefficients are associated with a small error (also called noise or tolerance), caused for instance either by measurements, rounding off or perturbations. These problems defined in terms of polynomials with coefficients of limited accuracy (or rather their problem specifications) are ill-posed algebraic problems in the sense of Hadamard. An ill-posed problem is a problem that does not fulfill Hadamard's definition of well-posedness: (i) for all data, a solution exists; (ii) for all data the solution is unique; (iii) the solution depends continuously on the data.

In the literature, an ill-posed problem usually refers to a problem that does not verify the third condition (iii) of well-posedness in Hadamard's definition. In the case of an ill-posed algebraic problem, small perturbations in the coefficients of the polynomials produce dramatic changes in the solution. This is the case in classical algebraic problems from computer algebra such as: the computation of the greatest common divisor of polynomials, the computation of the singularities of a plane algebraic curve, the factorization of polynomials, root computation of polynomials, Gröbner bases computation, genus computation, etc.

For our purpose, we concentrate on two specific algebraic problems from computer algebra: the analysis of singularities and the computation of topological invariants (i.e. delta-invariant, genus) of a plane complex algebraic curve. We mention that these problems are well-understood if the coefficients of the defining polynomial of the curve are exact data. The challenge is to handle these problems if the coefficients are inexact. In this thesis, we approach the algebraic problem of computing topological invariants of a plane complex algebraic curve defined by a squarefree bivariate complex polynomial with both exact and inexact data. We recall that for the inexact data we associate a positive real number called tolerance or noise, which measures the error level in the coefficients of the defining polynomial of the input curve. By computing the topological invariants of a plane complex algebraic curve we also analyse the local topology of its singularities. We mention that the algebraic problem of computing topological invariants of a plane complex algebraic curve defined by a squarefree polynomial is an ill-posed problem in the sense that tiny changes in the coefficients of the defining polynomial of the input curve produce dramatic changes

in the output solution. To deal with the ill-posedness of our problem, in general we basically construct symbolic-numeric methods that approximate solutions to the considered ill-posed problem, solutions that are stable under tiny changes of the input. In particular, in the case of our problem, we apply the following strategy: instead of analysing directly the singularities of the input plane complex algebraic curve (either by Puisseaux expansion or by resolution as described in [Sendra and Winkler, 1991]), we intersect the input curve having a singularity in the origin with a small sphere centered in the origin and we analyse this intersection.

At this point, we give some reasons, which support the need for studying topological invariants of a plane complex algebraic curve. For the purpose of this thesis, we mention that the topological invariants of a plane complex algebraic curve refer to those properties that are unchanged under continuous deformation (i.e. homeomorphism). Roughly speaking, we mention that a plane complex algebraic curve is a subset of the 2-dimensional complex plane. Since the 2-dimensional complex plane is isomorphic to the 4-dimensional real space, we identify a plane complex algebraic curve with a 2-dimensional object (i.e. a surface) in the 4-dimensional real space. Thus we can associate to a plane complex algebraic curve, its corresponding surface. An important topological invariant of a plane complex algebraic curve is its genus, i.e. the number of holes in the corresponding associated surface. For instance, a coffee mug and a doughnut are topologically equivalent objects up to homeomorphism since they both have one hole (i.e. their genus equals one), as seen in Figure 1.1.

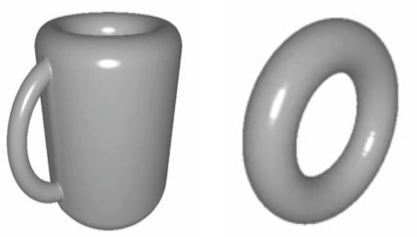


Figure 1.1: In topology, a doughnut and a coffee mug are equivalent objects, they both have one hole (i.e. their genus is 1). Pictures generated from Youtube.

The genus computation problem is a classical subject in computer algebra. Presently, several symbolic algorithms are available for computing the genus computation, see the books of [Gutierrez et al., 2002] [Hess, 2004], [Sendra and Winkler, 1997] for more information. There exist also good implementations for these algorithms in several packages (or libraries) of some well-known computer algebra systems such as: Maple [Geddes et al., 2008], Magma [Bosma et al., 1997], Singular [Greuel and Pfister, 2002], and the system Axiom [Jenks and Sutor, 1992]. We shortly list these packages, for more information see the papers of [Greuel and Pfister, 2002], [Haché, 1994], [Hess, 2004], [Hoeij, 2000], and the paper of [Mnuk and Winkler, 1996]:

- *algcurves* package developed at Florida State University, by Mark van Hoeij, written in Maple;
- *CASA* (Computer algebra system for algebraic geometry) package developed at the Research Institute for Symbolic Computation, Hagenberg Austria and written in Maple;
- *GHS* (Gaundry, Hess, Smart) attack package developed at Berlin Institute of Technology and written in Magma;

- *normal.lib* package developed at Kaiserslautern University and written in Singular;
- *PAFF* (Package for algebraic function fields in one variable) package developed at INRIA-Roquencort, by Gaétan Haché, written in Axiom.

However, these symbolic algorithms for genus computation have some disadvantages, and these are:

- for exact input data (i.e. integer numbers or rational numbers) of moderate size, they become too expensive (in terms of time and memory required for the computation) since they use exact arithmetic. In this case, we would obtain a crash during the running time of the algorithm.
- for inexact input data (i.e. numerical values, that is real values with noise in them) they are practically unusable. In this case, we would obtain an error message, since these symbolic algorithms do not allow the insertion of input data with noise in them.

In addition, there are situations when computing with numerical coefficients is preferable, for instance when the coefficients are obtained by measurements. Thus, there arises the need to develop numeric algorithms for genus computation, numeric algorithms that are using floating point arithmetic to speed up the computation process and to handle inexact input data. In this thesis, we propose a symbolic-numeric algorithm for computing the genus (and other topological invariants such as the delta-invariant, the Alexander polynomial) of a plane complex algebraic curve defined by a squarefree polynomial with both exact and inexact coefficients. This algorithm makes use of the advantages of both symbolic and numeric methods and it is mainly based on combinatorial techniques from knot theory see [Adams, 2004], [Livingston, 1993], techniques that allow us to successfully analyse the singularities of the plane complex algebraic curve. Previous research and results have successfully shown that the local topology of the singularities of a plane complex algebraic curve is mainly determined by their links, see [Milnor, 1968]. Moreover, the algebraic link can be uniquely identified by its corresponding Alexander polynomial, as shown in [Yamamoto, 1984]. From the Alexander polynomial, we can derive a formula for the delta-invariant of each singularity of the plane complex algebraic curve and a formula for the genus of the curve. It follows that for solving the problem of computing topological invariants of a plane complex algebraic curve, we need to solve problems in different mathematical areas such as: the intersection of algebraic surfaces (by using notions from real algebraic geometry), the topology (shape) recognition of algebraic curves (by employing techniques from computational geometry), the computation of knot invariants (by using methods from knot theory).

A natural question arises: why are we interested in computing the genus of a plane complex algebraic curve? We first add that given a plane complex algebraic curve defined by a squarefree complex polynomial, the implicit representation of the curve consists of the defining polynomial equation attached to the curve. Basically, the implicit representation of a curve requires to solve a polynomial equation to find the points of the curve. Such a representation has the advantage that it offers an easy and a practical method to decide whether a point lies on the curve or not. From the literature, we know that if the genus of a plane algebraic curve is zero, then the curve has a rational parametric representation. The rational parametric representation of a curve consists of a pair of univariate rational functions, which except for finitely many exceptions, represent all the points on the curve. We mention that for practical applications, the rational representation of a plane algebraic curve is desirable instead of the implicit representation of the curve, since plane algebraic curves defined by their rational parametrization are easier to display on a computer. Moreover, without going into details we add that the rational parametrization of curves is highly used

in application domains such as geometric modeling or computer aided geometric design. It is thus essential to compute the genus of a plane algebraic curve to decide whether the curve admits a rational parametric representation, and in the affirmative case to compute such a rational parametric representation.

1.2 Setting the Framework

As mentioned in Section 1.1, solving the algebraic problem of computing topological invariants of a plane complex algebraic curve requires solving several problems from different domains of mathematics and computer science such as: algebraic geometry, computational geometry, knot theory, approximate algebraic computation. These domains have encountered major achievements and changes over the years, which contributed to the advance of research and technology and which contributed to the development of this thesis. In this section, we give an overview containing the basic facts about these domains, which are important for the purpose of this thesis. We mention that this overview is by no means exhaustive.

1.2.1 Contemporary Algebraic Geometry

It is important to specify that as objects of study, we place the algebraic curves in the broad and the intricate field of algebraic geometry. We make some brief notes concerning this field of research to give the reader a "rough" idea on the "general purpose" of algebraic geometry. The fundamental objects of study in algebraic geometry are the algebraic varieties. We give an informal definition of these objects by mentioning that an algebraic variety is defined to be the set of points in the n-dimensional complex space (or in the n-dimensional real space) satisfying a system of polynomial equations. Thus, an algebraic variety is the set of solutions of a system of polynomial equations. Basically, algebraic geometry deals with the problem of analysing the solution sets of systems of polynomial equations. The main directions of research in algebraic geometry are concerned with two main topics:

- the identification of general properties of algebraic varieties.
- and the classification of algebraic varieties.

An important characteristic of an algebraic variety is its dimension. Informally, the dimension of an algebraic variety represents the number of coordinates needed to specify a point, i.e. the number of degrees of freedom of movement in the space. Formally, the dimension of an algebraic variety X is said to be d if d is the largest integer such that there is a strictly increasing chain $X_0 \subset X_1 \subset ... \subset X_d$ of closed irreducible subsets of X. It follows that the dimension of an algebraic variety is defined to be the length of the longest chain of irreducible closed subsets minus one. Moreover, we distinguish between affine and projective algebraic varieties. In the literature, other types of algebraic varieties are additionally defined. In this setting, it is useful to note that a plane algebraic curve is an algebraic variety of dimension one. For more information on the study of algebraic varieties in algebraic geometry, the reader can consult [Fulton, 1989]. For our purpose, algebraic geometry provides efficient tools for studying plane algebraic curves.

1.2.2 Knot Theory and its Evolution

Knot theory, which dates back to the late 19th century, gained increased attention in the last two decades of the 20th century together with its potential applications in physics,

chemistry, biology or mathematics. Still, most of the problems in knot theory are open problems. Roughly speaking, knot theory is a branch of topology that is involved in the study of three-dimensional manifolds, i.e. in the study of the ways in which knotted copies of a circle can be embedded in the 3-dimensional Euclidean space. We base our overview concerning knot theory on the book of [Adams, 2004].

The study of knots dates back to the $1800s$, when Gauss basically started to study these topological objects from a pure mathematical point of view. In 1877, J. C. Maxwell, William Thompson (also known as Lord Kelvin) and P. Tait produced the first knot tables. In addition, Lord Kelvin's theory "that all matter is made of ether and atoms are knots in the ether" made it interesting for the mathematical society to study knots in more details. Later on, in 1900, Poincarè formalised the modern theory of topology, which led to a significant progress in the field of knot theory. We mention some significant achievements in this direction:

- In 1928, Alexander described a method for associating to each knot a polynomial, i.e. an invariant for knots. He based his method on purely combinatorial aspects (without using any algebraic techniques). Basically, the method depends only on the study of the diagram of the knot. We mention that the diagram of a knot is a special type of projection of the knot in the 2-dimensional Euclidean plane. However, Alexander polynomial is not a complete invariant for knots, as it distinguishes only the knots of 8 crossings or fewer.

- In 1932, K. Reidemeister published the first book "Knotentheorie" about knots, which was considered a big achievement for the field of knot theory.

- In 1984, Jones described a method for associating a new polynomial to each knot, i.e. another invariant for knots. Jones polynomial basically distinguishes all knots of 10 or fewer crossings, and it also distinguishes a knot from its mirror image. Still, it is not a complete invariant for knots, as it does not distinguish all knots (i.e. the mutant knots). Moreover, in 1984, Yamamoto showed that the Alexander polynomial is a complete invariant for a special type of knots called algebraic knots, see [Yamamoto, 1984]. This means that different algebraic knots have different Alexander polynomials. As we will see in the next chapters, this result is of outermost importance for our study.

- In 1985, Hoste, Ocneanu, Millett, Freyd, Lickorish, and Yetter introduced the HOMFLY polynomial as an invariant for knots. We add that the HOMFLY polynomial was independently introduced also by Prztycki and Traczyk in 1987. Still, the HOMFLY polynomial is not a complete invariant for knots, since it also does not distinguish the mutant knots. We mention that at present there exist no complete invariant for knots, this subject representing one of the open problems in the challenging field of knot theory.

We end this section by noting that at present, knot theory is still a dynamic branch of topology. It has concrete applications in the study of enzymes acting on DNA (deoxyribonucleic acid) strands, which basically represent the substance of the living matter. The connection between the study of DNA strands and knot theory is that DNA is tangled up in knots. Note that DNA strands are long and thin molecules found inside the nucleus of a cell. DNA must be topologically manipulated in order for vital process to take place, which means that DNA must first unpack itself so that it can interact with enzymes. By thinking of DNA as a knot, we can use knot theory to estimate how hard DNA is to unknot. For more information on this deep connection, the reader should consult [Crick and Watson, 1953] and [Cozzarelli and Wasserman, 1986].

1.2.3 Computational Geometry

Computational geometry is a branch of computer science, that deals with algorithms expressed in terms of geometric objects. The intricate and the fascinating domain of computational geometry experienced excessive interest starting with the middle of the $1970s$. Recently, computational geometry has known an increasing success due to its different applications in domains such as computer graphics, robotics, computer aided geometric design, geographic information systems, scientific visualization. The domain of computational geometry basically studies flat or straight geometric objects (such as line segments, planes, polygons, polyhedra) or simple straight geometric objects such as circles. We add that computational geometry handles mainly geometric objects in the 2-dimensional Euclidean plane, and a more restrictive class of such objects in the 3-dimensional Euclidean space. In addition, this area of study is focused mainly on the discrete nature than on the continuous nature of geometric objects.

For our purpose, we handle a rather classical problem from computational geometry and that is the problem of computing all the intersection points among a set of given edges (i.e. line segments) in the 2-dimensional Euclidean plane. In our study, this set of given edges represents the projection of a set of edges from the 3-dimensional space, which are part of a 3-dimensional graph data structure. We add that a 3-dimensional graph data structure is given as a set of vertices (points) in the 3-dimensional Euclidean space together with their Euclidean coordinates, and a set of edges connecting them. For solving the problem of computing all the intersection points among the edges of the projection of a 3-dimensional graph data structure, we design an adapted version of the Bentley-Ottmann algorithm from [Berg et al., 2008]. We mention that the Bentley-Ottmann algorithm is an essential tool in computational geometry, which solves the fundamental problem of computing the intersection points of a set of edges in the 2-dimensional Euclidean plane. Furthermore, the Bentley-Ottmann algorithm uses the sweep line technique for solving this problem. Based on [Berg et al., 2008], we include a list containing other fundamental problems from computational geometry:

- The problem of computing the planar convex hull of a set of points: given a set of points in the 2-dimensional Euclidean plane, find the smallest convex polygon containing all the points, which represents the planar convex hull of the given set of points.
- The polygon triangulation problem: given a polygon in the 2-dimensional Euclidean plane, partition its interior into triangles.
- The problem of computing the Voronoi diagram of a set of points: given a set of points including the point p, decompose the space into regions (cells) around each point such that all the points in the region around the point p are closer to p than any other point in the given set.
- The problem of computing the Delaunay triangulation of a set of points: given a set of points, the Delaunay triangulation is the triangulation of the convex hull determined by the given set of points such that the collection of edges in this triangulation satisfy the property that for each edge e in the triangulation, we can find a circumcircle containing the endpoints of e but not containing any other points. The Delaunay triangulation and the Voronoi diagram in the 2-dimensional Euclidean plane are dual to each other.
- 1-dimensional searching problem: given a set of points on the real line, find all the points in a given interval, which is called a 1-dimensional query rectangle.

For details on the main geometric algorithms developed for solving these fundamental computational geometry problems, the reader is advised to consult the detailed book of [Berg et al., 2008] on computational geometry. We point out that most of the computational geometry algorithms rely on the following algorithmic techniques: (i) the incremental technique, (ii) the sweep line technique, (iii) and the divide and conquer technique.

1.2.4 Recent Progress in Approximate Algebraic Computation

The field of approximate commutative algebra (also called numerical polynomial algebra) handles ill-posed algebraic problems by employing different methods to design robust and efficient algorithms to solve these problems [Corless et al., 2003], [Stetter, 2004]. In this subsection, we give a survey on these methods. We mention that this survey is by no means exhaustive. We mention that this chapter is part of the paper [Hodorog and Schicho,].

A first technique in handling ill-posedness of algebraic problems is to develop numerical algorithms that compute solutions to ill-posed problems. These solutions are computed in such a way that they are stable under small changes of the input data. In the literature, this type of technique is called a *regularization* method. In [Zeng, 2003], [Zeng, 2009b], [Zeng, 2009a] [Zeng, 2005], [Zeng and Dayton, 2004] the authors introduce methods based on regularizing principles for designing algorithms in numerical polynomial algebra for solving ill-posed algebraic problems. These methods are basically using matrix computations (i.e. singular value decomposition method) and the Newton iteration method. The authors employ these methods to compute: the approximate irreducible factorization of univariate and multivariate polynomials, the approximate greatest common divisor of univariate and multivariate polynomials, the approximate rank of a matrix, the approximate kernel of a matrix and more, see [Zeng, 2009b] for details.

A second way of solving ill-posed algebraic problems is to look for minimal perturbations in the coefficients representing the input data of the problem such that the obtained perturbed data satisfy the desired property. By using this strategy, in [Shuhong et al., 2008] and in [Kaltofen et al., 2007], the authors design a method for computing the approximate greatest common divisors of univariate polynomials. The main idea is to reduce the problem of computing the greatest common divisor of polynomials to the problem of approximating the low rank of a Sylvester matrix. The method basically uses structured least norm algorithms applied on matrices with Sylvester structure.

In different situations from geometric computing, one has to handle with geometric algorithms. The geometric algorithms usually imply numerical computations and combinatorial algorithms, but also algebraic computations. Hence, the geometric algorithms are ill-posed, since small numerical errors in input lead to huge modifications in the output. For instance, the zero test is an ill-posed problem, intensively used in geometric algorithms. A useful tool in handling ill-posedness in geometric problems is the *exact geometric paradigm* developed by [Li et al., 2004], which is based on three main ingredients: constructive zero bounds, approximate expression evaluation and numerical filters. Consequently, the exact geometric computation paradigm handles the non-robustness in geometric computations.

Another method to solve ill-posed algebraic problems is the *homotopy continuation method*. This numerical method is used in the field of numerical algebraic geometry to compute the solution set to a system of polynomial equations. Other applications of the homotopy continuation method in the field of numerical algebraic geometry are: the computation of all isolated solutions of a polynomial system, the numerical irreducible decomposition of an algebraic set, the numerical computation of the geometric genus of any one-dimensional irreducible component of an algebraic set, see [Bates et al., 2011] and [Sommese and Wampler, 2005] for details. We notice that the method described in

[Bates et al., 2011] is computing the numerical genus of a plane complex algebraic curve. We mention that the genus is a global topological invariant of a plane complex algebraic curve.

At present, in the literature, another method for computing the genus of a plane complex algebraic curve is reported in [Pérez-Díaz et al., 2010]. The authors approach the problem of computing an approximate parametrization of an affine plane complex algebraic curve with ordinary singularities. The idea of the algorithm for solving this problem is to introduce the notion of cluster for the approximate singularities and to work with linear systems of curves. Hence, the algorithm computes the genus of an affine plane complex algebraic curve with ordinary singularities. Even though this method is rather restrictive (i.e. it applies only to plane algebraic curves with ordinary singularities that are computed in the affine plane), it still offers a robust and stable solution to the ill-posed problem of computing the genus of a plane complex algebraic curve.

1.2.5 Development of Mathematical Software Packages and Libraries

The invention and the evolution of computers throughout the years have contributed tremendously to the development of science and of technology and to important advances in different fields of mathematics. In this thesis, by a mathematical software package (or library) we mean a software (or computer program), which is mainly developed and used for computing with geometric, symbolic and numeric objects and data. In the last years, a wide variety of mathematical software packages and libraries were developed in the mathematical research community. These packages and libraries have different purposes and they handle different areas of mathematics such as algebraic geometry, algebra, computational geometry, knot theory, combinatorics. For our purpose, we present the main mathematical software packages and libraries essential for the topic of this thesis, i.e. for developing symbolic-numeric algorithms for plane complex algebraic curves:

- *Maple* [Geddes et al., 2008] computer algebra system: this system was originally developed at the University of Waterloo, Canada in the 1980*s*. The system includes a wide varieties of tools for symbolic manipulation, numeric computation, visualization and programming and it presents a friendly user interface. The system is oriented for offering both extensive support for mathematical and engineering education but also for performing applied research such as financial modeling, high-performance computing, operations research.

- *Mathematica* [Wolfram, 2000] computer algebra system: the system was released in 1988 and it offers tools for symbolic computation, numerical computation, statistics, visualization. The system is used in science, engineering and various fields of mathematics.

- *Singular* [Greuel and Pfister, 2002] computer algebra system: this system was developed starting in 1984. Singular is an open source computer algebra system for polynomial computations, which concentrates mostly on computations in commutative and noncommutative algebra, algebraic geometry and singularity theory. The system is written in the C programming language and it is mainly developed at the University of Kaiserslautern, Germany.

- *CoCoA* (Computations in Commutative Algebra) [CoCoATeam, 1996] computer algebra system: as its name states it, this system is mainly handling computations in commutative algebra. The system performs operations on multivariate polynomial

rings and on different data related to them such as modules, ideals, rational functions and matrices. The system is written in the C programming language, it is open source and it was originally developed starting in 1987.

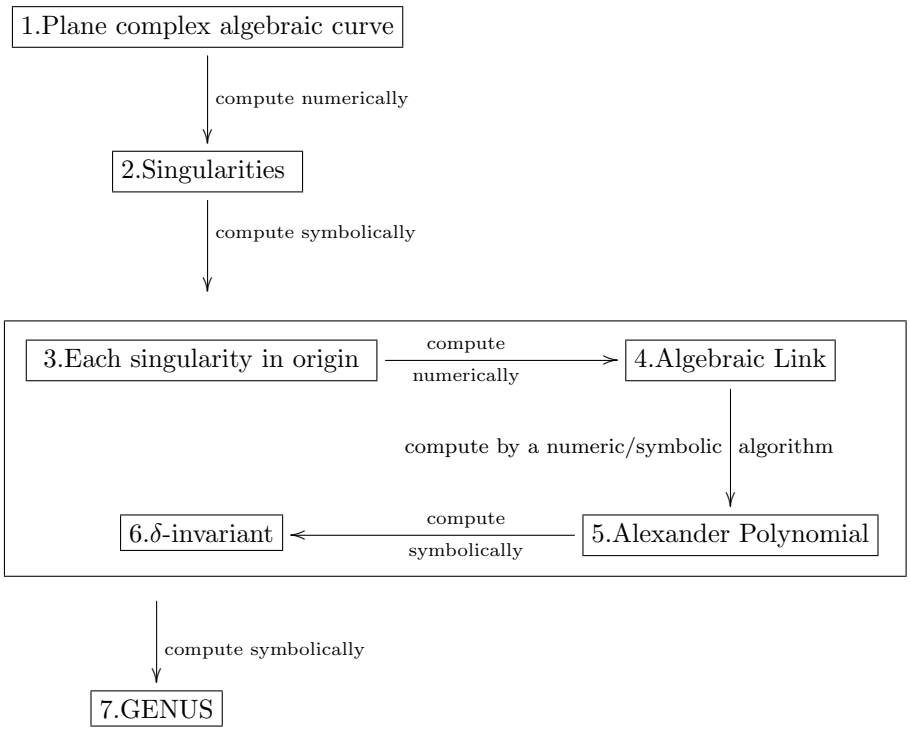
- *CGAL* [CGAL Project Members, 1997]. The name of this package stands for Computational Geometry Algorithms Library. This is an open source library and it was developed starting in 1995. The library offers efficient data structures and algorithms in the vast domain of geometry. We mention some of the fundamental data structures and algorithms included in CGAL: triangulations, Voronoi diagrams, polygons, arrangements of curves and their applications, mesh generation, geometry processing, convex hulls algorithms, shape analysis, fitting, search structures, interpolation. The library is written in the C++ programming language and it is used both in academia and in industry in different fields such as medical modeling, geographic information systems, computational topology and computer vision.
- *GMP* (GNU Multiprecision Library) [Foundation, 2000] library: this is an open source library for arbitrary precision arithmetic, which was originally released in 1991. The library basically operates on signed integers, rational numbers, and floating points numbers and it is written in the C programming language.
- *MPFR* [Hanrot et al., 2005] library: this is a free library for multiple-precision floating-point computations with correct rounding. The library is written in the C programming language and it was released in 1991. We add that the MPFR library is based on the GMP library. We mention that CGAL and Mathemagix computer algebra system are both using the MPFR library.
- *Mathemagix* [van der Hoeven et al., 2002] computer algebra system: this is a free computer algebra system, written in the C++ programming language and released in 2000. The system offers various tools for algebraic computations, symbolic computations, and numeric computations.
- *Bertini* [Bates et al., 2006] system: this is a free software for numerical algebraic geometry, designed mainly for solving systems of polynomial equations by using the homotopy continuation method. The system is written in the C programming language and it was originally developed around 2006.

1.3 Strategy for Solving the Problem

We recall that in this thesis we deal with the algebraic problem of computing topological invariants of a plane complex algebraic curve defined by a squarefree polynomial with both exact (i.e. integers or rationals) and inexact data (i.e. numerical values given together with some noise). We handle an ill-posed algebraic problem, i.e. small changes in the input data produce huge changes in the output solution. We solve the ill-posed algebraic problem of computing topological invariants of a plane complex algebraic curve by designing numerical methods that compute approximate solutions to ill-posed algebraic problems, approximate solutions that are stable under small changes in the input data. In particular, we intersect the input plane complex algebraic curve with a singularity in the origin with a small sphere centered in the origin and we analyse this intersection. This intersection basically allows us to analyse the local topology of the input plane complex algebraic curve around its singularity.

In order to solve the algebraic problem of computing topological invariants (i.e. delta-invariant, genus) of a plane complex algebraic curve, we first need to design the symbolic-numeric algorithms for computing the topological invariants of a plane complex algebraic

curve and its singularities. In order to design these symbolic-numeric algorithms for computing the topological invariants of a plane complex algebraic curve, we need to solve the following interdependent subproblems:



and that is: (i) we compute the singularities of the input plane complex algebraic curve; (ii) we translate each computed singularity in the origin; (iii) we compute the link for each translated singularity depending on an input parameter ϵ; (iv) we compute the Alexander polynomial for each singularity from the link of the singularity; (v) we derive a formula for the delta-invariant of each singularity from the Alexander polynomial; (vi) we compute the genus from the delta-invariants of all the singularities. (vii) in addition, we can derive formulas for the Milnor number of each singularity and for the Euler characteristic of the Riemann surface attached to the resolution of singularities of the input plane complex algebraic curve.

We make the important observation that the development of the symbolic-numeric algorithms for computing topological invariants of a plane complex algebraic curve is only one step in solving the algebraic problem of computing topological invariants of a plane complex algebraic curve defined by a squarefree polynomial with both exact and inexact coefficients. We will give a detailed description concerning the complete solution of the considered algebraic problem in the next section.

1.4 Contributions of the Thesis

We mention that the results from this thesis are based on the papers [Hodorog et al., 2010a], [Hodorog and Schicho, 2011b], [Hodorog et al., 2010b], [Hodorog and Schicho, 2011a] and [Hodorog et al., 2011], [Hodorog and Schicho,]. Moreover, we base this study also on the technical reports [Hodorog and Schicho, 2010a] and [Hodorog and Schicho, 2010b].

In computer algebra, the problem of computing topological invariants of a plane complex

algebraic curve is well-understood if the coefficients of the defining polynomial of the input curve are exact data (i.e. integer numbers or rational numbers). The challenge is to handle this problem if the coefficients of the input polynomial are inexact data, i.e. real numbers with given noise in them, noise that measures the error level in the coefficients.

In this thesis, we treat the algebraic problem of computing topological invariants (i.e. delta-invariant, genus) of a plane complex algebraic curve defined by a squarefree polynomial with coefficients of limited accuracy, i.e. the coefficients are both exact (i.e. integer numbers or rational numbers) and inexact data (i.e. numerical values). For the inexact values, we associate a positive real number called tolerance (or noise), which measures the error level in the coefficients of the defining polynomial of the curve. This problem is ill-posed in the sense that tiny changes in the coefficients of the defining polynomial of the input curve cause huge changes in the output solution of the problem. We employ a *regularization method* based on [Engl et al., 1996, Tikhonov and Arsenin, 1977] to handle the ill-posedness of the problem. This regularization method allows us to construct approximate solutions to the considered ill-posed problem, solutions that are stable under small changes in the initial data.

Our regularization method consists of the following two components:

(i) a set of symbolic-numeric algorithms (called also approximate algorithms). These algorithms take as input the defining squarefree polynomial of the input plane complex algebraic curve, a positive real number called the input parameter (or the *regularization parameter*), and a subset of the 3-dimensional Euclidean space. We mention that the coefficients of the defining polynomial of the input curve are both exact and inexact data. Moreover, the inexact data is associated with a positive real number called *noise level*. These symbolic-numeric algorithms return as output the approximate topological invariants of the input plane complex algebraic curve and its singularities. We add that the computation of the approximate topological invariants of the curve depends on the regularization parameter.

(ii) a *parameter choice rule* for the regularization parameter. This parameter choice rule is basically a function in the noise level associated to the coefficients of the defining polynomial of the input curve. It is essential for the parameter choice rule to satisfy the following property called *convergence for noisy data*: as the noise level in the coefficients of the defining polynomial of the input curve decreases to zero, the approximate solutions computed by the symbolic-numeric algorithms together with the parameter choice rule converge to the exact solution of the considered problem. The convergence for noisy data property basically states that if the regularization parameter is chosen according to a specific rule (i.e. the parameter choice rule), then as the noise level decreases to zero, the approximate solutions computed by the symbolic-numeric algorithms together with the regularization parameter tend to the exact solution of the original ill-posed algebraic problem.

Remark 1. We make an important observation concerning the terminology used in this thesis. The name of "symbolic-numeric algorithms" is derived from the fact that the developed algorithms use both symbolic methods (e.g. Sturm-Habicht sequences to compute the greatest common divisors of polynomials, Bareiss algorithm to compute the determinant of polynomials matrices from [van der Hoeven et al., 2002]) and numeric methods (e.g. subdivision from [Mourrain and Pavone, 2009]). Furthermore, since the input information processed by the symbolic-numeric algorithms is represented by both exact and inexact data, we sometimes refer to these algorithms as the approximate algorithms. Finally, by computing the approximate topological invariants of a plane complex algebraic curve and by proving the existence of a parameter choice rule satisfying the convergence for noisy

data property, we basically estimate the topological invariants of a plane complex algebraic curve defined by a squarefree polynomial with both exact and inexact data.

For designing the regularization method for computing the approximate topological invariants of a plane complex algebraic curve, we proceed in the following way:

(1) We develop the symbolic-numeric algorithms for computing the approximate topological invariants of a plane complex algebraic curve defined by a squarefree complex bivariate polynomial with exact and inexact data. For computing the approximate topological invariants of a plane complex algebraic curve, we proceed as follows:

- Firstly, we compute the set of numerical singularities of the input curve in the projective real plane by subdivision methods from [Mourrain and Pavone, 2009], see Chapter 3, Section 3.1. Alternatively, we extend the subdivision methods to compute the numerical singularities of the input curve in the projective complex plane. We can also use the homotopy method from [Sommese and Wampler, 2005] to compute the set of numerical singularities of the input curve in the projective complex plane.
- Secondly, based on [Milnor, 1968], we compute the approximate link of each numerical singularity of the input curve by intersecting the curve with a small sphere centered in the singularity, see Chapter 3, Section 3.2. We consider the radius of this sphere to be the *regularization parameter* for the problem. We compute the approximate link of each numerical singularity as a 3-dimensional graph data structure by using subdivision methods from [Liang et al., 2008]. We mention that a 3-dimensional graph data structure is a set of vertices in $\mathbb{R}^3$ together with their Euclidean coordinates and a set of edges connecting them. Alternatively, we can use the homotopy continuation method from [Sommese and Wampler, 2005] to compute the 3-dimensional graph data structure representing the approximate link of each numerical singularity of the input curve. Basically, the computation of the approximate link of each numerical singularity allows us to analyse the local topology of each singularity of the input curve.
- Thirdly, we compute the approximate Alexander polynomial of the approximate link of each singularity, see Chapter 3, Section 3.3. For computing the approximate Alexander polynomial we use adapted algorithms from computational geometry [Berg et al., 2008] (i.e. an adapted version of the Bentley-Ottmann algorithm) and combinatorial objects from knot theory [Cimasoni, 2004, Livingston, 1993]. We mention that as shown in [Yamamoto, 1984] the Alexander polynomial is a complete invariant for links of singularities, i.e. different links of singularities have different Alexander polynomials. We mention that a symbolic algorithm for computing the Alexander polynomial of the singularity of a plane algebraic curve defined by a squarefree polynomial with exact coefficients (i.e. there is no noise in the input data) is reported and implemented in the computer algebra system Singular [Greuel and Pfister, 2002].
- Finally, from the approximate Alexander polynomial we derive formulas for the approximate delta-invariant of each numerical singularity, see Chapter 3, Section 3.4 for details. From the approximate delta-invariants of all the singularities of the input curve we derive a formula for the approximate genus of the input curve, see Chapter 3, Section 3.6. We make the observation that we compute the approximate genus of a plane complex algebraic curve independently of the type of singularity of the input plane complex algebraic curve (i.e. ordinary or non-ordinary singularity). In the case of a plane algebraic curve defined by a squarefree

polynomial with exact coefficients (i.e. there is no noise in the input data), the formula for the genus is derived depending on the type of the singularity of the curve (i.e. ordinary or nonordinary), see [Sendra et al., 2008] for more details.

- As applications, we add that by computing the approximate Alexander polynomial of each numerical link of a plane curve singularity we estimate the approximate local topological type of the plane curve singularity, see Chapter 3, Section 3.5. In addition, from the approximate Alexander polynomial we derive a formula for the approximate Milnor number of each singularity of the input plane complex algebraic curve. Moreover, from the approximate genus of the input curve we compute the approximate Euler characteristic of the Riemann surface attached to the resolution of singularities of the input plane complex algebraic curve, see Chapter 3, Subsection 3.6.2. Furthermore, in Chapter 3, Section 3.7, we compute the following knot theory properties attached to the approximate link of each singularity of the input curve: (i) the genus of the link; (ii) the unknotting number of the link; (iii) the determinant of the link; (iv) the number of knot components in the link; (v) the linking numbers of all the knot components in the link. In addition, we decide whether the approximate link is colorable or not.

(2) We implement the symbolic-numeric algorithms for computing the approximate topological invariants of a plane complex algebraic curve defined by a squarefree polynomial with exact and inexact data in a new software package called GENOM3CK (GENus cOMputation of plane Complex algebraiC Curves using Knot theory), see Chapter 5. For the implementation of this package we use the Mathemagix free computer algebra system [van der Hoeven et al., 2002] and the Axel free algebraic geometric modeler [Wintz et al., 2006]. For our purpose, both of these systems provide tools for algebraic computation, tools for exact and approximate computation, tools for geometric modeling and tools for modern 3-dimensional visualization of geometric objects. As the Mathemagix and the Axel systems, GENOM3CK is a free software released under the GNU General Public License.

(3) We proof the existence of a parameter choice rule for the regularization parameter of the problem, parameter choice rule that satisfies the convergence for noisy data property, see Chapter 4. For constructing the proof we use notions and principles from topology and from real algebraic geometry. We add that by proving the convergence for noisy data of the designed symbolic-numeric algorithms that computes the approximate topological invariants of a plane complex algebraic curve, we basically estimate the topological invariants of this curve.

(4) We perform several test experiments with the package GENOM3CK, test experiments that support the convergence for noisy data property.

1.5 Structure of the Thesis

We organize this thesis as follows:

(1) In **Chapter 2** we include the fundamental mathematical notions required for our study concerning the symbolic-numeric algorithms for plane complex algebraic curves. We divide this chapter into five main parts:

- In *Section 2.1* we give a short historical background concerning the plane algebraic curves, emphasizing on the applications of these algebraic objects to the real world problems. In addition, we define the affine and the projective plane algebraic

curves defined over an algebraically closed field (e.g. the complex numbers) and we include several examples for a better understanding of the presented notions.

- In *Section 2.2*, we define the singularities of affine and of projective plane algebraic curves defined over an algebraically closed field (e.g. the complex numbers) and we include several examples. Moreover, we present some applications of singularities in various areas such as catastrophe theory.
- In *Section 2.3*, we discuss the topology of plane complex algebraic curves. We mention that the notions introduced in this part refer only to plane complex algebraic curves. First of all, we introduce some preliminaries notions from the broad field of topology. We then discuss the global topology of plane complex algebraic curves (including the notion of the genus of a plane complex algebraic curve) and the local topology of a plane complex algebraic curve around its singular points. We emphasize that the study of the local topology of a plane complex algebraic curve around its singularities can be identify by studying the link of the singularity. We define in detail the notion of a link of a plane curve singularity and we include several examples.
- In *Section 2.4*, we introduce the topological invariants of a plane complex algebraic curve: the link of each singularity of the plane complex algebraic curve, the Alexander polynomial of the link of each singularity of the curve, the delta-invariant of each singularity of the curve, the genus of the curve, the Milnor number of each singularity, the Euler characteristic of the Riemann surface attached to the resolution of singularities of the plane complex algebraic curve. We discuss in details each of these invariants of a plane complex algebraic curve.
- In *Section 2.5*, we define the key notions studied in this thesis, i.e. the approximate topological invariants of a plane complex algebraic curve. We explain what exactly do we mean by an approximate topological invariant of a plane complex algebraic curve. We study the following approximate topological invariants of a plane complex algebraic curve: the approximate link of each singularity of the plane complex algebraic curve, the approximate Alexander polynomial of each approximate link, the approximate delta-invariant of each singularity, the approximate genus of the curve, the approximate Milnor number of each singularity, the approximate Euler characteristic of the Riemann surface attached to the resolution of singularities of the input curve.

(2) **Chapter 3** contains the main symbolic-numeric algorithms that we developed in order to compute the approximate topological invariants of a plane complex algebraic curve, approximate invariants that we defined in Chapter 2. We basically design the following symbolic-numeric algorithms for a plane complex algebraic curve:

- In *Section 3.1*, an algorithm for computing the real singularities of the input plane complex algebraic curve in the projective real plane. This algorithm basically uses subdivision methods from [Mourrain and Pavone, 2009]. Furthermore, based on the same subdivision methods we present an extended algorithm for computing the complex singularities of the input plane complex algebraic curve in the projective complex plane.
- In *Section 3.2*, an algorithm for computing the approximate link of each singularity of the input plane complex algebraic curve. This algorithm depends on a positive real number (called regularization parameter), which basically represent the radius of a small sphere centered around the singularity. We compute the approximate link of each singularity as the stereographic projection of the intersection of this

sphere with the input curve. We mention that the approximate link of a singularity is a smooth and closed space algebraic curve, defined as the intersection of two algebraic surfaces in the 3-dimensional real space. We basically compute the approximate link as a 3-dimensional graph data structure, by using subdivision methods from [Liang et al., 2008]. We mention that a 3-dimensional graph data structure is given as a set of points (vertices) in the 3-dimensional space together with their Euclidean coordinates, and a set of edges connecting them.

- In *Section 3.3*, an algorithm for computing the approximate Alexander polynomial of each approximate link. This algorithm depends also on the input regularization parameter. For computing the approximate Alexander polynomial we use algorithms from computational geometry (i.e. an adapted version of the Bentley-Ottmann algorithm) and combinatorial objects from knot theory (i.e the diagram of the approximate link, which is a special kind of projection of the approximate link in the 2-dimensional real plane).
- In *Section 3.4*, an algorithm for computing the approximate delta-invariant of each singularity of the input curve. This algorithm depends also on the input regularization parameter. We derive a formula for the approximate delta-invariant of each singularity based on the degree and on the number of variables of the approximate Alexander polynomial of the approximate link of the singularity.
- In *Section 3.5*, an algorithm for computing the approximate local topological type of each singularity of the input curve. This algorithm depends as well on the input regularization parameter. We define the approximate topological type of the singularity Q of the input curve as the pair containing the value for the approximate link of the singularity Q, the value for the approximate Alexander polynomial of the approximate link of Q, and the value of the approximate delta-invariant of the singularity Q.
- In *Section 3.6*, an algorithm for computing the approximate genus of the input plane complex algebraic curve. This algorithm depends on the input regularization parameter. We derive a formula for the approximate genus based on the degree of the input curve and of the values of the approximate delta-invariants of all the singularities of the input plane complex algebraic curve.
- In *Subsection 3.3.4*, an algorithm for computing the approximate Milnor number of each singularity of the input curve. Similar to the other algorithms, this algorithm depends also on the input regularization parameter. We derive a formula for computing the approximate Milnor number based on the degree and on the number of variables of the approximate Alexander polynomial of the approximate link of the singularity.
- In *Subsection 3.6.2*, an algorithm for computing the approximate Euler characteristic of the Riemann surface attached to the resolution of singularities of the input plane complex algebraic curve. This algorithm depends on the input regularization parameter. We compute the value for the approximate Euler characteristic based on the value of the approximate genus of the input curve.
- In *Section 3.7*, an algorithm for computing the following various properties from knot theory for the approximate link of each singularity of the input plane complex algebraic curve: the genus of the approximate link, the unknotting number of the approximate link, the determinant of the approximate link, the number of knot components of the approximate link, the sequence of linking numbers for all the knot components of the approximate link. We also decide whether the approximate link is colorable or not.

(3) In **Chapter 4** we introduce regularization principles used in the field of approximate algebraic computation. We explain the way in which we apply these regularization principles to the considered algebraic problem of computing invariants of a plane complex algebraic curves defined by a squarefree polynomial with exact and inexact data. We recall that for the inexact data we associate a positive real number called noise, which measures the error level in the coefficients. We recall that the symbolic-numeric algorithms designed in Chapter 3 depend on an input parameter called regularization parameter, which is a positive real number. We show that the designed symbolic-numeric algorithms designed in Chapter 3 compute approximate solutions to the considered algebraic problem, approximate solutions that satisfy the following property (called convergence property for noisy data): as the noise level in the coefficients of the defining polynomial of the input curve tends to zero and the regularization parameter is chosen according to a specific rule (called a parameter choice rule), the approximate solutions computed by the designed symbolic-numeric algorithms tend to the exact solutions of the considered problem. Thus instead of computing exact solutions to the considered algebraic problem, we compute approximate solutions satisfying the convergence for noisy data property.

(4) In **Chapter 5** we discuss the implementation issues. We present in details the new software package GENOM3CK (Symbolic numeric techniques for GENus cOMputation of plane Complex algebraiC Curves using Knot theory). GENOM3CK is an open source library, which contains the implementation of the symbolic-numeric algorithms designed in Chapter 3. We organize this chapter in four parts:

- In *Section 5.1*, we describe the main functionalities of the library, we include a short history concerning the development of the library and we present the main interface of the library.
- In *Section 5.2*, we present the main systems on which the library was built on and we present the main dependencies of the library
- In *Section 5.3*, we describe in details a set of instructions for guiding both the interested user and the experienced developer in using the library.
- In *Section 5.4*, we include a large set of test experiments performed with the library. A first set of experiments shows the computation of the approximate invariants of a plane complex algebraic curve, performed with the symbolic-numeric algorithms from Chapter 3. A second set of experiments represents the evidence for the convergence for noisy data property of the symbolic-numeric algorithms, property that we proved in Chapter 3.

(5) In **Chapter 6** we include the conclusions concerning the work presented in this thesis and we discuss the future directions of research.

Chapter 2

Plane Complex Algebraic Curves

2.1 Preliminaries on Affine and Projective Plane Complex Algebraic Curves

In this section we introduce and we define the main objects of our research, i.e. the affine and the projective plane complex algebraic curves. We first give a short outline concerning the historical evolution of plane algebraic curves and then we introduce several definitions and examples concerning plane algebraic curves defined over an algebraically closed field (e.g. the complex numbers).

2.1.1 A Brief Historical Background

In this subsection we make a summary on the historical facts concerning plane algebraic curves. For this purpose, we mainly follow the book of [Brieskorn and Knorrer, 1986].

We recall that plane algebraic curves are the fundamental objects of study in the field of algebraic geometry, which increasingly developed in the past and in the present century and which is still encountering recent valuable developments. In this subsection we give a short overview of the origin of plane algebraic curves and we indicate some of the main reasons for which mathematicians began the study of these mathematical objects. As a rough general remark, we mention that the primary reasons for the origin of plane algebraic curves are: the development of mathematical problems, the mathematical constructions, or the applications in other fields such as astronomy, technology, architecture, etc. In the following we present the main classes of plane algebraic curves by shortly discussing their origin and by depicting their main applications into the real world.

The first class of basic plane algebraic curves are the circle and the line. These objects were known by Greeks such as Thales (600 B.C.) or Euclid (300 B.C.) and they were defined as loci of points having certain distance properties to given points. For instance, the circle is defined as the locus of points situated at the same distance from a given point called the center of the circle. Moreover the distance from any point and the center is called the radius of the circle. As applications, the circle and the line were mainly used for land measurements or for building constructions. Even though it was not until 1637 that R. Descartes introduced the use of Cartesian coordinates for describing the plane algebraic curves, for the sake of completion in our overview we will also introduce the implicit equations using Cartesian coordinates of the discussed plane algebraic curves. Thus,

we remember that the implicit equation of the circle centered in the point of Cartesian coordinates (a, b) and of radius r is $(x-a)^2+(y-b)^2 = r^2$, whereas the implicit equation of the line through two points of Cartesian coordinates (x_1, y_1) and (x_2, y_2) is given by $(y_1 - y_2)x + (x_2 - x_1)y + x_1y_2 - x_2y_1 = 0$. In Figure 2.1 we visualize a circle and a line in the 2-dimensional Euclidean plane.

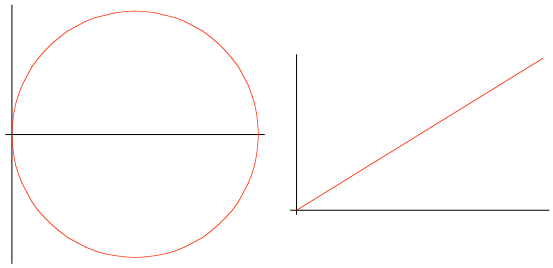


Figure 2.1: Example of a circle and a line. From left to right: (1) the circle given by $(x-1)^2+(y-0)^2+1 = 0$; (2) the line given by $y = x/2$. Pictures produced with Mathematica, see [Wolfram, 2000] for more information.

The second class of fundamental plane algebraic curves are the conic sections, which were generated by Greeks as intersections of cones with planes. These conic sections are: the hyperbola, the parabola and the ellipse. In Figure 2.2 we visualize the intersection of the cone with equation $x^2 + y^2 - z^2 = 0$ and of the plane with equation $3x + 2y - 2 = 0$, where we notice that the intersection is a hyperbola.

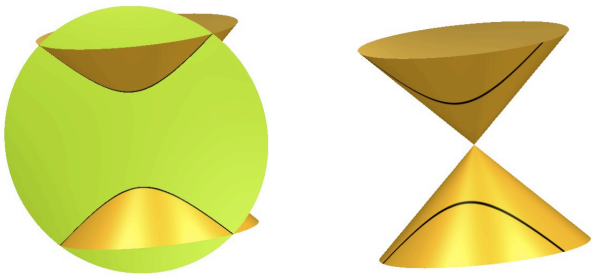


Figure 2.2: Generation of the hyperbola by intersecting the cone C given by $x^2+y^2-z^2 = 0$ with the plane P given by $3x + 2y - 2 = 0$. From left to right: (1) the intersection of the cone (in yellow) with the plane (in green); (2) the hyperbola (in black) defined by the intersection $C \cap P$. Pictures produced with Surfex, see [Holzer and Labs, 2008] for more information.

In Figure 2.3 we visualize the intersection of the cone with equation $x^2 + y^2 - z^2 = 0$ and of the plane with equation $x - z - 2 = 0$, where we notice that the intersection is a parabola.

In Figure 2.4 we visualize the intersection of the cone with equation $x^2 + y^2 - z^2 = 0$ and of the plane with equation $x - y + 4z + 4 = 0$, where we notice that the intersection is an ellipse.

In today's notation, the conic sections, which are quadratic curves, are represented by the following equation:

$$y^2 = px + qx^2. \tag{2.1}$$

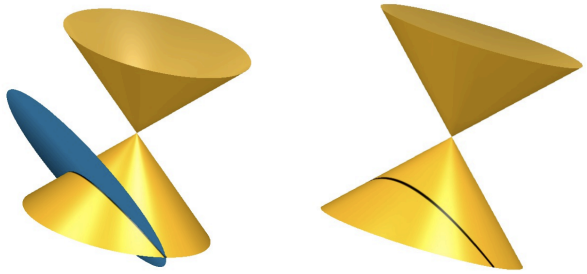


Figure 2.3: Generation of the parabola by intersecting the cone C given by $x^2+y^2-z^2=0$ with the plane P given by $x-z-2=0$. From left to right: (1) the intersection of the cone (in yellow) with the plane (in blue); (2) the parabola (in black) defined by the intersection $C \cap P$. Pictures produced with Surfex, see [Holzer and Labs, 2008] for more information.

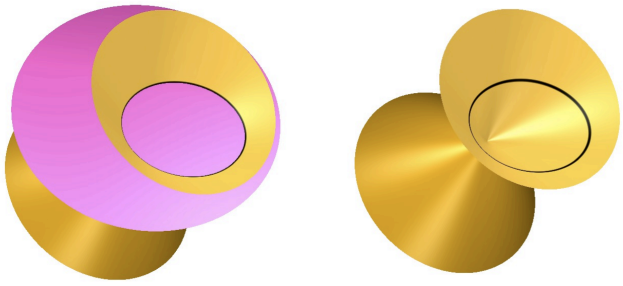


Figure 2.4: Generation of the ellipse by intersecting the cone C given by $x^2+y^2-z^2=0$ with the plane P given by $3x-y+4z+4=0$. From left to right: (1) the intersection of the cone (in yellow) with the plane (in pink); (2) the ellipse (in black) defined by the intersection $C \cap P$. Pictures produced with Surfex, see [Holzer and Labs, 2008] for more information.

Depending on the values of the parameters p and q in Equation (2.1), we obtain the following particular conic sections:

- if $q = 0$, then we get the parabola;
- if $q > 0$, then we obtain the hyperbola;
- and if $q < 0$, then we acquire the ellipse.

As a remark, we add that the circle is also a quadratic curve, being in this way included in the class of conic sections. In Figure 2.5 we include some examples of conic sections.

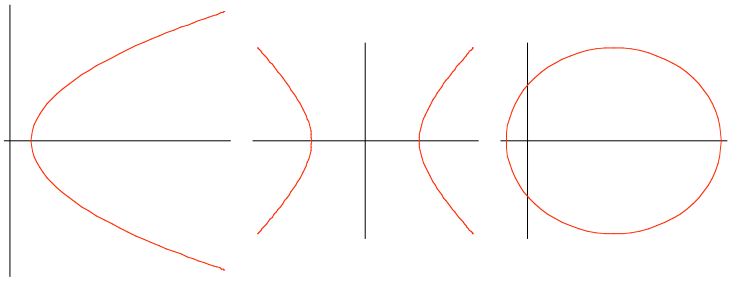


Figure 2.5: Example of conic sections. From left to right: (1) the parabola given by $y^2 = 2x-1$; (2) the hyperbola given by $y^2 = x^2-1$; (3) the ellipse given by $16y^2 = -12x^2+8x-1$. Pictures produced with Mathematica, see [Wolfram, 2000] for more information.

The Greeks used the conic sections to solve some classical problems such as the duplication of the cube problem also known as the Delian problem. We briefly explain this problem. For more information on the classical geometrical problems the reader is advised to consult the fascinating book of [Beman and Smith, 2007] on this topic. Basically, the Delian problem is formulated in the following way: given the length of an edge of a cube denoted with x, one is asked to construct a second cube having double the volume of the initial given cube. Solving the Delian problem reduces to solving the equation $x^3 - 2 = 0$, i.e. finding the root of the equation $x^3 - 2 = 0$. In the past, finding the roots of equations was mainly performed by geometrical constructions, which were basically suited to algebraic treatment. Furthermore, these geometrical constructions were restricted to using only two types of instruments, i.e. a ruler (also called a straight-edge because it did not present any markings on it) and a compass. An important observation is that the equation $x^3 - 2 = 0$ does not have roots that are constructible only by ruler and by compass. This observation follows from Galois theory, see [Gaal, 1998] for more details. It follows that one cannot construct the length $\sqrt[3]{2}$ by ruler and compass alone, which shows in fact that the duplication of the cube problem is unsolvable by this method. Still by using the conic sections, the Greeks successfully solved the Delian problem, for more details see [Brieskorn and Knorrer, 1986, p. 6]. The main idea for solving the Delian problem using the conic sections is to determine the intersection of the two parabolas given by the following equations:

$$\begin{cases} y^2 = 2x \\ x^2 = y. \end{cases} \tag{2.2}$$

We notice that solving the system of equations (2.2) reduces to solving the equation $x^4 = 2x$, from which we obtain the following solutions: $x = 0$ and $x = \sqrt[3]{2}$. Therefore the Delian problem can be solved assuming that a method for constructing parabolas does exist, and that this method generates the whole curve and not only individual points of the curve.

The third class of plane algebraic curves are the cubic curves, such as the cissoid of Diocles (180 B.C.). This curve has a cusp in the origin, and it is symmetric to the Ox axis. In addition, this curve is one of the oldest example of a curve with such a singularity as the cusp, a notion that we will define and explain in the next sections. The implicit equation of the cissoid of Diocles in Cartesian coordinates is given by

$$x^3 + xy^2 - 2ay^2 = 0. \tag{2.3}$$

In addition, this curve has a vertical asymptote at $x = 2a$. In Figure 2.6 we include an example of a cissoid of Diocles defined by the Equation (2.3), where we choose the parameter a to be 1.

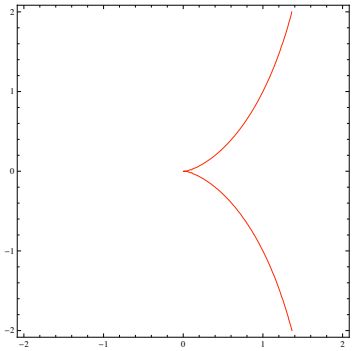


Figure 2.6: Example of a cissoid of Diocles given by $x^3 - 2y^2 + xy^2 = 0$. Picture produced with Mathematica, see [Wolfram, 2000] for more information.

The main reason for which Diocles invented this curve is for solving the Delian problem. We give here the basic idea for solving the Delian problem using the cissoid of Diocles. We consider the following equation of the cissoid of Diocles, where we set the parameter a from Equation (2.3) to be $a = \frac{1}{2}$:

$$x^3 + xy^2 - y^2 = 0. \tag{2.4}$$

Furthermore, we rewrite the Equation (2.4) in the form $x^3 = (1-x)y^2$. Then by multiplying both sides of this equality with y, we get the equivalent equation:

$$\left(\frac{y}{x}\right)^3 = \frac{y}{1-x}. \tag{2.5}$$

Next we construct the line with the equation $y = 2x$, which cuts the asymptote $x = 1$ in the segment of length 2 and which intersects the cissoid in a point with the following property:

$$\frac{y}{1-x} = 2^3. \tag{2.6}$$

The Equality (2.6) represents the equation of the line passing through the point of Cartesian coordinates $(1, 0)$ denoted with P, and therefore of the line joining the point P to the point of the cissoid. It follows that the length $\sqrt[3]{2}$ can be constructed in the following way: (i) we draw the line l joining the two points of Cartesian coordinates $(2, 0)$ and $(1, 0)$. This line l intersects the cissoid in a point Q; (ii) we draw the line m joining the origin and the point Q. This line m intersects the asymptote $x = 1$ in a segment that equals the length $\sqrt[3]{2}$. For more information on this issue see [Beman and Smith, 2007, p.44].

We further report on more essential plane algebraic curves. The conchoids of Nicomedes (180 B.C.) are in fact a one parameter family of curves. These curves also solve the Delian problem. For details, the reader can check [Brieskorn and Knorrer, 1986, p. 13]. The conchoids of Nicomedes are defined as a family of curves with two parameters a and b by the following equation:

$$(x-a)^2(x^2+y^2) = b^2x^2. \tag{2.7}$$

As a note we mention that this family of curves has an asymptote $x = a$, and the area between either branch and the asymptote is infinite. Depending on the values of the parameters a and b in Equation (2.7), Nicomedes distinguished between the following three different classes of curves: (1) the class of curves for $0 < a/b < 1$; (2) the class of curves for $a/b = 1$; (3) and the class of curves for $a/b > 1$. We notice that for $a = 0$, this family of curves degenerates to a circle. In Figure 2.7 we observe some examples of curves belonging to each of these three different classes of curves. In addition, for visualization purposes in Figure 2.8 we render the conchoids of Nicomedes with the Equation (2.7) in the domain $[-1.1, 3.1] \times [-2, 2] \subset \mathbb{R}^2$.

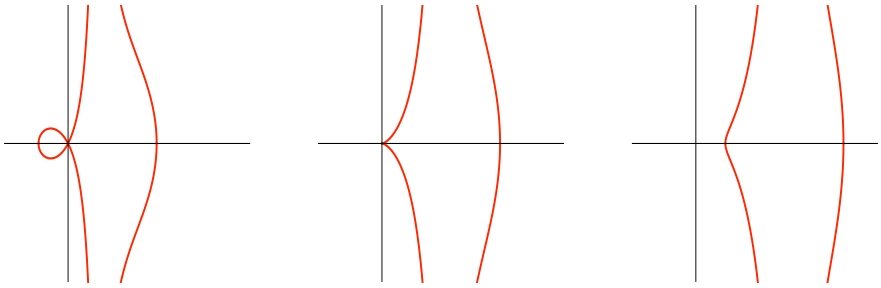


Figure 2.7: Example of representatives from the family of curves with Equation (2.7), defining the conchoids of Nicomedes. From left to right: (1) representative for $a/b = 1/2$; (2) representative for $a/b = 1$; (3) representative for $a/b = 3/2$. Pictures produced with Mathematica, see [Wolfram, 2000] for more information.

Another class of special curves are the epicycle curves, which were used by astronomers as Ptolemy (150 B.C.) to describe the paths of the planets. We notice several examples of epicycle curves in Figure 2.9. In particular, an epicycle is defined as the locus of a point P on the boundary of a circle of radius r_1 rolling without slipping on the outside of a fixed circle of radius r_2, see Figure 2.10 for a rough visualization of this type of generation process.

During the Renaissance period, other types of interesting curves were found such as the wheel curves. In fact, it was later on noticed that the wheel curves are included in the class of epicyclic curves. A particular class of wheel curves are the cycloids, see Figure 2.11 for some examples. More precisely, the cycloid is defined as the locus of a point on the boundary of a circle of radius r rolling along a straight line, see Figure 2.12 for a fundamental visualization of this type of generation process.

During the Renaissance, a huge number of literature was written concerning the cycloids by famous mathematicians. We do not wish to insist on the long list with the names of these famous mathematicians, for a survey see [Brieskorn and Knorrer, 1986, p.26], but we do wish to report on two technical applications of the cycloids: (i) the cycloidal gear, which is used in the construction of mechanical clocks and watches. We give a basic explanation for the notion of a cycloidal gear: by a gear we mean a rotating machine with teeth, which meshes with another rotating machine with teeth to transmit motion or to change speed

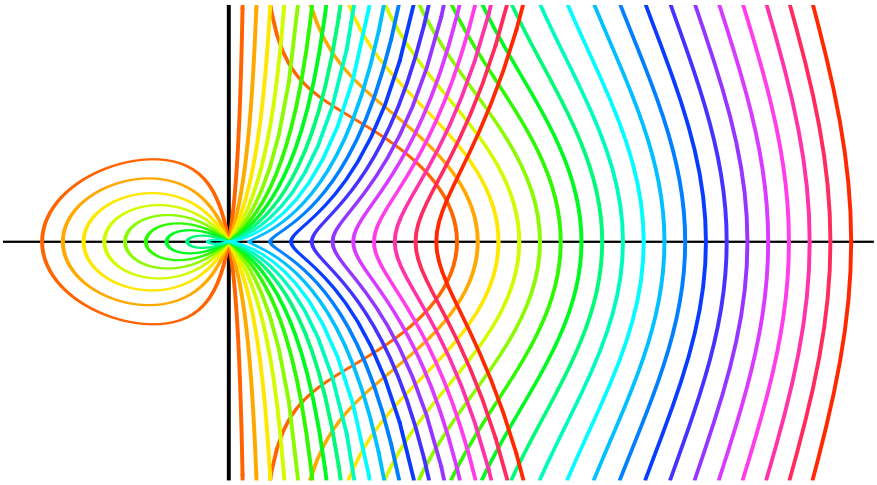


Figure 2.8: Conchoids of Nicomedes representing the family of curves with Equation (2.7), visualized in the domain $[-1.1, 3.1] \times [-2, 2] \subset \mathbb{R}^2$. Picture produced with Mathematica, see [Wolfram, 2000] for more information.

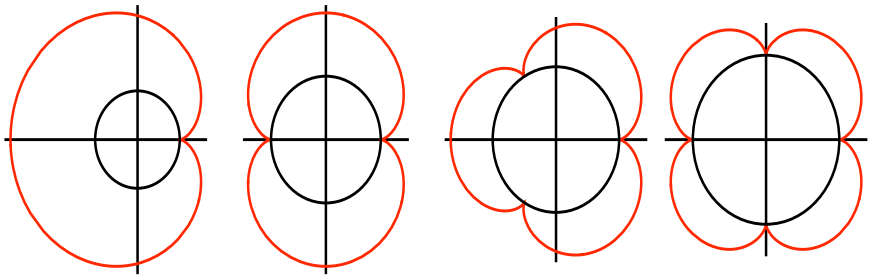


Figure 2.9: Examples of epicycles. Pictures produced with Mathematica, for more information see [Wolfram, 2000].

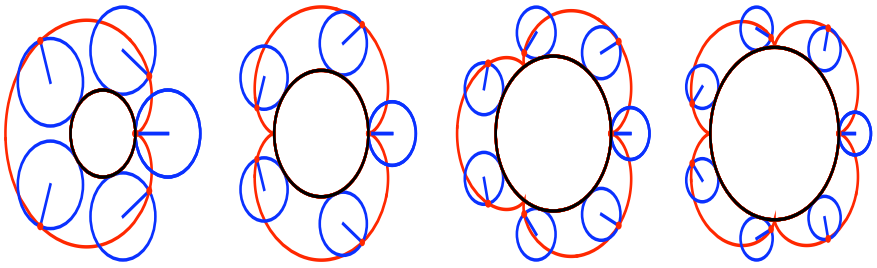


Figure 2.10: Generation of the epicycles from Figure 2.9. In each picture, the epicycle is represented by the path (in red) traced out by a point P on the boundary of a circle of radius r_1 (in blue) rolling without slipping on the outside of a fixed circle (in black) of radius r_2. Pictures produced with Mathematica, see [Wolfram, 2000] for more information.

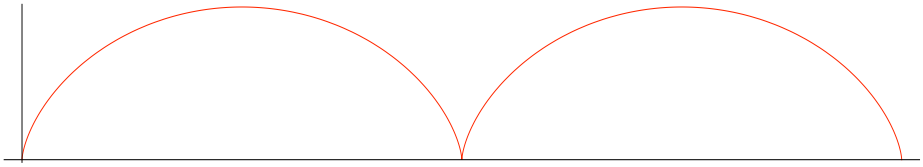


Figure 2.11: Example of a cycloid. Picture produced with Mathematica, for more information see [Wolfram, 2000].

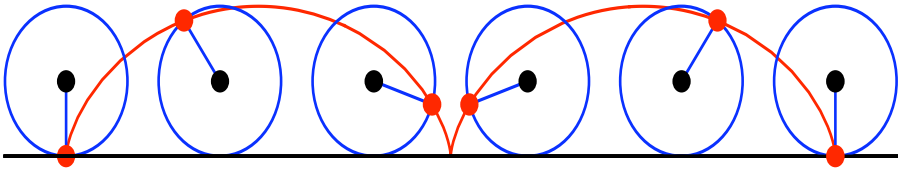


Figure 2.12: Cycloid (in red) generated by a circle (in blue) rolling on a straight line. Picture produced with Mathematica, see [Wolfram, 2000].

or direction. In particular, a gear whose teeth are made up of cycloidal curves is called a cycloidal gear, for an example see Figure 2.13; (ii) another technical application of the cycloids is the trochoidal rotation reciprocator, which represents the geometrical basis for the construction of the Wankel motor. The design of the Wankel engine is complicated enough to be presented here and it includes several engineering terms, therefore we will omit from our overview. For details on this issue, the reader can check [Brieskorn and Knorrer, 1986, p. 38]. Nevertheless, we report on some practical applications of the Wankel engines by mentioning that these types of engines were installed in different types of vehicles and devices, such as for instance cars, aircrafts, etc.

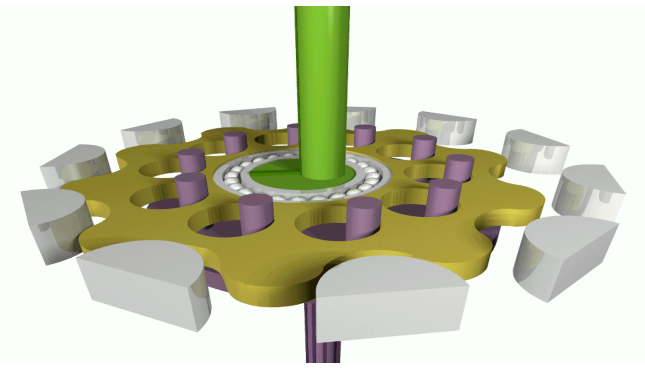


Figure 2.13: Example of a cycloidal gear from http://www.rmhoffman.com.

Another class of important plane algebraic curves, which were generated as a result of a technological discovery, are the Watt curves. These curves were introduced in 1784 by the Scottish engineer J. Watt, who developed the steam engine. Nowadays, the Watt curves are defined by the following implicit equation in Cartesian coordinates:

$$(x^2+y^2)(x^2+y^2-a^2-b^2+c^2)^2+4a^2y^2(x^2+y^2-b^2)=0. \tag{2.8}$$

In Figure 2.14 we include an example of a Watt curve defined by Equation 2.8, where we set the parameters a, b, c to have the following particular values: $a = 2.1,\ b = 2$ and $c = 2.5$.

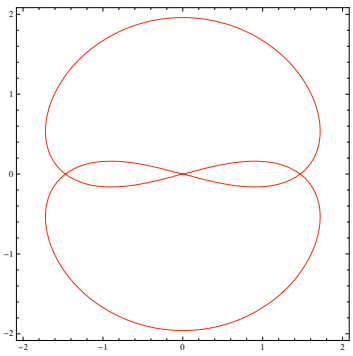


Figure 2.14: Example of a Watt curve defined by the Equation (2.8) with the parameters $a = 2.1,\ b = 2$ and $c = 2.5$. Picture produced with Mathematica, see [Wolfram, 2000] for more information.

We also report on another significant class of plane algebraic curves called the Lissajou curves, which were first studied in 1850 by the French mathematician J. A. Lissajou, while he was investigating different vibration problems. The Lissajou curves are a family of curves represented by the following parametric equations:

$$\begin{aligned} x(t) &= a \sin(\omega t + \delta) \\ y(t) &= b \sin t. \end{aligned} \tag{2.9}$$

The Lissajou curves have applications in physics, astronomy and other sciences. As applications in physics for instance, the Lissajou curves are the family of curves that the oscilloscope, which represents an electronic test instrument, draws when its two inputs (representing horizontal and vertical shifts) are connected to two oscillatory signals. We mention that in Figure 2.15 we visualize several representatives from the Lissajou family of curves with Equation (2.9), where we consider the parameter δ to be 0, we set the parameter b to be 1, and we consider several different values for the parameters a and ω. From Figure 2.15, we notice that the visual forms of some Lissajou curves resemble to 3-dimensional knots. In fact many 3-dimensional knots projects on the 2-dimensional plane as Lissajou curves. For a precise definition of 3-dimensional knots, the reader should consult Subsection 2.3.3, where we introduce and we study in details these mathematical objects, which actually represent one of the key and the central notion of this thesis.

We end here our overview concerning the historical background of plane algebraic curves. We wish to emphasize that this overview only contains some illustrative classes of plane algebraic curves and it is, by no means, exhaustive. Our main purpose in this subsection was to include some of the main reasons for the study of plane algebraic curves and to familiarize the reader with some fundamental examples of plane algebraic curves. Basically, we saw that plane algebraic curves have a history of more than two thousand years, and that they have important technical applications, which connects them to the real word. It is thus essential to introduce a formal study of these mathematical objects in order to understand their mathematical behaviour. We cover these aspects in details in the next sections of this chapter.

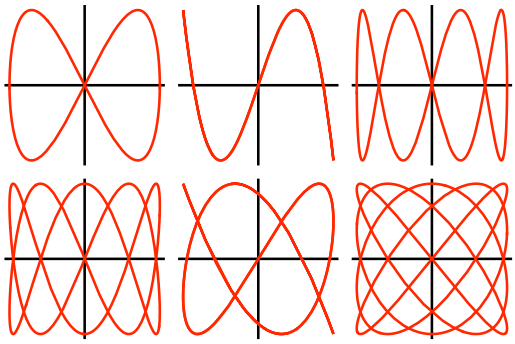


Figure 2.15: Example of representatives from the family of curves with Equation (2.9), defining the Lissajou curves. We set the parameters $\delta = 0$ and $b = 1$ for all the representative. The parameters a and ω are individually chosen as follows. First row from left to right: (1) representative for $a = 1,\ \omega = 1/2$; (2) representative for $a = 1,\ \omega = 1/3$; (3) representative for $a = 1,\ \omega = 1/4$. Second row from left to right: (1) representative for $a = 1,\ \omega = 2/5$; (2) representative for $a = 1,\ \omega = 3/5$; (3) representative for $a = 1,\ \omega = 4/5$. Pictures produced with Mathematica, see [Wolfram, 2000] for more information.

2.1.2 Definitions and Examples

In this subsection, we first recall several notions from algebra, which are essential for defining the objects of our study, i.e. the plane complex algebraic curves. Then, we define the plane complex algebraic curves, which we divide in two main classes: the class of affine plane complex algebraic curves, and the class of projective plane complex algebraic curves. We will handle each of this class in details, including several basic definitions and examples. In our study, we mainly follow the books of [Fulton, 1989], [Kirwan, 1992] and [Winkler, 1996]. We define a squarefree bivariate polynomial in the following way:

Definition 1. Let K be an algebraically closed field of characteristic 0. We say that a polynomial $p(z, w) \in K[z, w]$ is squarefree if there does not exist a polynomial $q(z, w) \in K[z, w]$ of positive degree such that $\big(q(z, w)\big)^2$ divides $p(z, w)$. It follows that $p(z, w)$ is squarefree if we cannot write $p(z, w) = \big(q(z, w)\big)^2 r(z, w)$, where $q(z, w), r(z, w) \in K[z, w]$ and $q(z, w)$ is nonconstant.

An important reason for defining squarefree polynomials is that if $p(z, w)$ is squarefree, then it does not have multiple roots. In several applications from engineering and physics a squarefree polynomial is referred to as a polynomial with no repeated factors. We define an irreducible bivariate polynomial as follows:

Definition 2. Let K be an algebraically closed field of characteristic 0. We say that a polynomial $p(z, w) \in K[z, w]$ is irreducible if there do not exist two nonconstant polynomials $q(z, w), r(z, w) \in K[z, w]$ such that $p(z, w) = q(z, w)r(z, w)$.

We observe that an irreducible polynomial is squarefree, but a squarefree polynomial is not necessarily irreducible. In the literature, the problem of squarefree factorization of a polynomial $p(z, w) \in \mathbb{C}[z, w]$ consists of determining the pairwise relatively prime squarefree polynomials $b_1(z, w), ..., b_s(z, w) \in \mathbb{C}[z, w]$ such that $p(z, w) = \prod_{i=1}^{s} b_i(z, w)^i$. In addition,

the problem of factorization of a polynomial $p(z, w) \in \mathbb{C}[z, w]$ refers to decomposing the polynomial into irreducible polynomials. We familiarize the reader with these two problems in Example 1.

Example 1. We consider the bivariate polynomial $p(z, w) = -1 + 3z - 5z^2 + 7z^3 - 7z^4 + 5z^5 - 3z^6 + z^7 - 2w + 4zw - 4z^2w + 6z^3w - 6z^4w + 4z^5w - 4z^6w + 2z^7w - w^2 - zw^2 + 6z^2w^2 - 6z^3w^2 + 6z^4w^2 - 6z^5w^2 + z^6w^2 + z^7w^2 - 2zw^3 + 4z^2w^3 - 2z^3w^3 + 2z^4w^3 - 4z^5w^3 + 2z^6w^3 - z^2w^4 + 3z^3w^4 - 3z^4w^4 + z^5w^4 \in \mathbb{C}[z, w]$ of degree 7.
The squarefree factorization of $p(z, w)$ into squarefree polynomials is:

$$p(z, w) = (-1 + z)^3(1 + z^2 + w + zw + z^2w + zw^2)^2, \tag{2.10}$$

and the factorization of $p(z, w)$ into irreducible polynomials is:

$$p(z, w) = (-1 + z)^3(1 + w)^2(1 + z^2 + zw)^2. \tag{2.11}$$

We give some intuition on the importance of studying the problem of squarefree factorization. We recall that the squarefree factorization is a relatively inexpensive and simple step, which can be computed just by greatest common divisor operations. Moreover, it is the first step in the problem of factorization of a polynomial and in the problem of finding the roots of a polynomial.
We are now ready to introduce the objects of our study, i.e. the plane complex algebraic curves. First of all, we define the affine plane complex algebraic curves as follows:

Definition 3. Let $\mathbb{C}$ be the field of complex numbers. We recall that $\mathbb{C}$ is an algebraically closed field of characteristic 0. Let $\mathbb{A}^2(\mathbb{C}) = \{(z, w) \in \mathbb{C}^2\}$ be the affine complex plane, and let $p(z, w) \in \mathbb{C}[z, w]$ be an irreducible polynomial in z and w with coefficients in $\mathbb{C}$ of degree m. An (affine) plane complex algebraic curve $\mathcal{C}$ in $\mathbb{A}^2(\mathbb{C})$ of degree m defined by $p(z, w)$ is the set of zeroes of the polynomial $p(z, w)$, i.e. $\mathcal{C} = \{(z, w) \in \mathbb{A}^2(\mathbb{C}) \mid p(z, w) = 0\}$. The curve $\mathcal{C}$ is called irreducible if it has an irreducible polynomial defining it.

Based on Definition 3, we add that curves of degree 1 are called lines, curves of degree 2 are named conics, curves of degree 3 are known as cubics, etc. Since irreducible polynomials are squarefree, without loss of generality we can assume that the defining polynomial of an affine plane complex algebraic curve from Definition 3 is squarefree. The reason for considering squarefree polynomials is an useful consequence of the theorem called the Hilbert's Nullstellensatz [Kirwan, 1992, p. 30], which basically says that two squarefree polynomials define the same affine plane complex algebraic curve in $\mathbb{A}^2(\mathbb{C})$ if and only if they are scalar multiples of each other.

Remark 2. For simplicity reasons, for the rest of this thesis we will denote the affine complex plane by $\mathbb{C}^2$. We make an important observation referring to drawing the affine plane complex algebraic curves. Since $\mathbb{C}^2$ is isomorphic with $\mathbb{R}^4$, we consider a plane complex algebraic curve $\mathcal{C} \subset \mathbb{C}^2$ as a real two-dimensional object in $\mathbb{R}^4$. For visualization purposes, we cannot draw this object in $\mathbb{R}^4$, but we can sketch the equivalent curve in $\mathbb{R}^2$. If not explicitly stated otherwise, in this thesis, we will use this convention for displaying plane complex algebraic curves.

We point out that an affine plane complex algebraic curve in $\mathbb{C}^2$ is never compact, see Example 2. Without going into details concerning notions from topology, we recall at this point that a subset of $\mathbb{C}^2$ is compact if it is closed and bounded. For more details concerning notions from topology used in this thesis, please consult Section 2.3.

Example 2. We give an example, which shows that an affine plane complex algebraic curve is not compact. We consider the affine plane complex algebraic curve $\mathcal{C} = \{(z, w) \in \mathbb{C}^2 \mid z^2 + w^2 = 1\}$, which represents the unit circle in the affine complex plane. We notice that $\mathcal{C}$ is not compact, since it is not bounded. There exist two branches of $\mathcal{C}$ tending to infinity, one tending in the direction i, and the other one in the direction $-i$. We can compactify this curve, by adding two additional "points at infinity" in both directions.

Moreover, for different reasons such as for studying the intersection points of curves, or for using homotopy methods [Sommese and Wampler, 2005] for curves, it is helpful to compactify the affine plane complex algebraic curves by "adding points at infinity" obtaining in this way the projective plane complex algebraic curves, which reside in the projective complex plane. We indicate this compactification process in Example 3. Consequently, for counting the intersection points of curves one can effectively use Bezout's theorem [Kirwan, 1992, p. 52], which says that two projective plane complex algebraic curves of degree m and n intersect in exactly mn points counting multiplicities. A particular result of Bezout's theorem states that any two distinct lines in the projective complex plane will intersect in exactly one point.

Example 3. We include an example, which informally indicates the compactification process of an affine plane complex algebraic curve. We take the affine plane complex algebraic curves $\mathcal{C} = \{(z, w) \in \mathbb{C}^2 \mid z^2 - w^2 - 1 = 0\}$ and respectively $\mathcal{D} = \{(z, w) \in \mathcal{C}^2 \mid z = \pm w\}$, see Figure 2.16. We observe that these curves do not intersect, but they are asymptotic as z and w tend to infinity. We can add "points at infinity" in $\mathbb{C}^2$ such that the two curves will intersect at infinity. In the same way, parallel lines will meet at infinity.

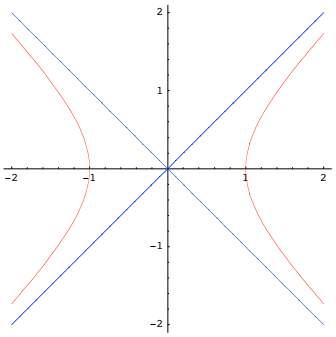


Figure 2.16: The blue curve given by $z^2 - w^2 - 1 = 0$ and the red curve given by $z = \pm w$. The picture represents the hyperbola $z^2 - w^2 - 1 = 0$ together with its two asymptotes $z = \pm w$. Picture produced with Mathematica, see [Wolfram, 2000] for more information.

To make the compactification of the affine plane complex algebraic curves more formal, we introduce the notion of projective complex plane. We consider the projective complex plane denoted with $\mathbb{P}^2(\mathbb{C})$, in the following way:

$$\mathbb{P}^2(\mathbb{C}) = \{(z : w : u) \mid (z, w, u) \in \mathbb{C}^3 \setminus \{(0, ..., 0)\}\},$$

where $(z : w : u) = \{(\alpha z, \alpha w, \alpha u) \mid \alpha \in \mathbb{C} \setminus \{0\}\}$. We identify the affine complex plane $\mathbb{C}^2$ inside $\mathbb{P}^2(\mathbb{C})$ as the subset of points $(z : w : 1)$. In addition, the points $(z : w : 0)$ form a line at infinity. In Example 2, the unit circle in the projective complex plane will contain two new points $(1 : i : 0)$ and $(1 : -i : 0)$.

We now introduce the projective plane complex algebraic curves. We consider the squarefree polynomial $p(z,w) \in \mathbb{C}[z,w]$ of degree m with

$$p(z,w) = p_m(z,w) + p_{m-1}(z,w) + ... + p_0(z,w),$$

where p_k are forms of degree k for all $k \in \{m, ..., 0\}$, i.e. all the terms occurring in p_k are of the same degree and $degree(p_k) = k$. We notice that $p(z,w)$ defines an affine plane complex algebraic curve $\mathcal{C}$ as in Definition 3. We consider $p(z,w,u)$ the homogenized polynomial of $p(z,w)$ in u with

$$p(z,w,u) = p_m(z,w) + p_{m-1}(z,w)u + ... + p_0(z,w)u^m.$$

We recall that a polynomial $p(z,w,u)$ is called homogeneous of degree m if

$$p(\Lambda z, \Lambda w, \Lambda u) = \Lambda^m p(z,w,u),$$

for all $\Lambda \in \mathbb{C}$. We now define a projective plane complex algebraic curve as follows:

Definition 4. A projective plane complex algebraic curve $\tilde{\mathcal{C}}$ (corresponding to $\mathcal{C}$) is the set of zeroes of the squarefree homogeneous polynomial $p(z,w,u)$, i.e.

$$\tilde{\mathcal{C}} = \{(z:w:u) \in \mathbb{P}^2(\mathbb{C}) \mid p(z,w,u) = 0\}.$$

Just as in the case of affine plane complex algebraic curves, two homogeneous squarefree polynomials $p(z,w,u)$ and $q(z,w,u)$ define the same projective plane complex algebraic curves if and only if they are scalar multiples of one another. Furthermore, we define the following notions concerning a projective plane complex algebraic curve:

Definition 5. The degree of a projective plane complex algebraic curve $\tilde{\mathcal{C}}$ in $\mathbb{P}^2(\mathbb{C})$ defined by a homogeneous polynomial $p(z,w,u)$ is the degree m of the polynomial $p(z,w,u)$. The projective curve $\tilde{\mathcal{C}}$ is called irreducible if $p(z,w,u)$ is irreducible, i.e. $p(z,w,u)$ has no nonconstant polynomial factors other than scalar multiples of itself.

Remark 3. We notice that any affine point (z,w) of $\mathcal{C}$ corresponds to a point $(z:w:1)$ of $\tilde{\mathcal{C}}$, and in addition to these points $\tilde{\mathcal{C}}$ contains only finitely many points "at infinity" with coordinates $(z:w:0)$. Even though the affine and the projective curves are different, they are closely related. From an affine plane complex algebraic curve $\mathcal{C}$ we can obtain a projective plane complex algebraic curve $\tilde{\mathcal{C}}$ by adding "points at infinity". More formally, if the squarefree polynomial $p(z,w) = 0$ of degree m defines an affine plane complex algebraic curve $\mathcal{C}$ in $\mathbb{C}^2$, then we can obtain its corresponding projective plane complex algebraic curve $\tilde{\mathcal{C}}$ in $\mathbb{P}^2(\mathbb{C})$ by adding appropriate powers of u to each term from $p(z,w)$ in order to get an homogeneous squarefree polynomial $p(z,w,u)$ of degree m, polynomial that defines the projective curve $\tilde{\mathcal{C}}$. Conversely, if $p(z,w,u) = 0$ is a projective plane complex algebraic curve in $\mathbb{P}^2(\mathbb{C})$ defined by the homogeneous squarefree polynomial $p(z,w,u)$, then we can dehomogenize the polynomial $p(z,w,u)$ with respect to its variables z, w and u to obtain the affine plane complex algebraic curves defined by the polynomials $p(1, \frac{w}{z}, \frac{w}{z}), p(\frac{z}{w}, 1\frac{u}{w})$ and respectively $p(\frac{z}{u}, \frac{w}{u}, 1)$.

In this way, in this subsection we defined the two classes of affine and of projective plane complex algebraic curves, explaining also the main connection between them. We mention that the definitions introduced in this subsection apply in the same way to the class of plane algebraic curves defined over any algebraically closed field. In the literature the theory of plane algebraic curves defined over an algebraic closed field is sometimes called the theory of plane algebraic curves.

2.2 Singularities of Plane Complex Algebraic Curves

When studying plane complex algebraic curves, we are interested in a special type of points called singular points, or simply singularities. Informally, the singularities of a plane complex algebraic curve are those points where the curve has "nasty" behaviour, such as for instance a point of self-intersection or a "cusp". It is useful to mention that the notion of a "cusp" will be precisely defined in the following subsections. In this section we include a formal study of the singularities of both affine and projective plane complex algebraic curves.

2.2.1 Definitions and Examples

In this subsection we define the singularities of plane complex algebraic curves and we introduce several examples for a better understanding of all the notions. We first discuss the singularities of an affine plane complex algebraic curve $\mathcal{C}$ defined by the squarefree polynomial $p(z,w) \in \mathbb{C}[z,w]$, following the textbooks [Brieskorn and Knorrer, 1986, p. 219] and [Walker, 1978, p. 52]. We introduce the singular points of $\mathcal{C}$ as those points $Q(a,b)$ of $\mathcal{C}$ at which the curve has more than one tangent counting multiplicities. We take L the line through the point $Q(a,b)$ of direction vector (λ,μ) and we study the intersections of L and $\mathcal{C}$ at Q. The lines through Q have the following parametric equation:

$$\begin{cases} z = a + \lambda t \\ \\ w = b + \mu t, \end{cases} \tag{2.12}$$

where $t \in \mathbb{C}$, and $(\lambda,\mu) \neq (0,0)$ with $(\lambda,\mu) \in \mathbb{C}^2$. The intersections of L and $\mathcal{C}$ are determined by the solutions of the system of polynomial equation:

$$\begin{cases} p(z,w) = 0 \\ \\ z = a + \lambda t \\ \\ w = b + \mu t, \end{cases} \tag{2.13}$$

or equivalently by the roots of the polynomial equation:

$$p(a + t\lambda, b + t\mu) = 0. \tag{2.14}$$

We expand $p(a + t\lambda, b + t\mu)$ from Equation (2.14) in the Taylor series expansion around t and we obtain the equivalent polynomial equation:

$$\Big(p_z(Q)\lambda + p_w(Q)\mu\Big)t + \Big(p_{zz}(Q)\lambda + p_{zw}(Q)\lambda\mu + p_{ww}(Q)\mu\Big)t^2 + ... = 0, \tag{2.15}$$

where $p_z, p_w, ...$ represent the derivatives of p. We distinguish the following cases:

(i) If in Equation (2.15) $p_z(Q), p_w(Q)$ are not both zero, then every line through Q has a single intersection with $\mathcal{C}$ at Q.

(ii) If in Equation (2.15) all the derivatives of p of order smaller than r vanish at Q but at least one derivative of order r does not vanish, then every line through Q has at least r intersections with $\mathcal{C}$ at Q, and exactly r such lines have more than r intersections.

These exceptional lines called the tangents to $\mathcal{C}$ at the point Q correspond to the roots of the following equation:

$$p_z(Q)\lambda^r + \binom{r}{1} p_{z^{r-1}}(Q)\lambda^{r-1}\mu + ... + \binom{r}{r} p_{w^r}(Q)\mu^r = 0, \tag{2.16}$$

and they are counted with multiplicities equal to the multiplicities of the roots of Equation (2.16). In this case, Q is called a point of multiplicity r of $\mathcal{C}$, or an r-fold point. A point of multiplicity 2 or more is called a singular point. We notice that a necessary and sufficient condition that a point $Q(a, b)$ is singular is that $p(a, b) = p_z(a, b) = p_w(a, b) = 0$, where p_z and p_w denote the partial derivatives of $p(z, w)$ with respect to z and w.

In this way, we introduce the following definition for the singularities of an affine plane complex algebraic curve:

Definition 6. Let $\mathcal{C}$ be an affine plane complex algebraic curve of degree m defined by the squarefree polynomial $p(z, w) \in \mathbb{C}[z, w]$. The set of singular points (or simply singularities) of $\mathcal{C}$ is defined as

$$Sing(\mathcal{C}) = \{(z_0, w_0) \in \mathbb{C}^2 \mid p(z_0, w_0) = \frac{\partial p}{\partial z}(z_0, w_0) = \frac{\partial p}{\partial w}(z_0, w_0) = 0\}.$$

For any singularity, we define its multiplicity as follows:

Definition 7. Let $\mathcal{C}$ be an affine plane complex algebraic curve of degree m defined by the squarefree polynomial $p(z, w) \in \mathbb{C}[z, w]$, and $Q(z_0, w_0) \in Sing(\mathcal{C})$. Let r be such that for any $j + k < r$, the partial derivative $\dfrac{\partial^{j+k} p}{\partial z^j \partial w^k}$ vanishes at the point $Q(z_0, w_0) \in \mathcal{C}$, but at least one of the partial derivatives of order r does not vanish. Then r is the multiplicity of $\mathcal{C}$ at Q. In this case, the polynomial

$$\sum_{j+k=r} \frac{1}{j!k!} \frac{\partial^r p}{\partial z^j \partial w^k}(Q)(z - z_0)^j (w - w_0)^k \tag{2.17}$$

factors completely in linear factors, the tangents of $\mathcal{C}$ at Q.

From Definition 7, we notice that the multiplicity r of $\mathcal{C}$ at Q is the order of the lowest non-vanishing term in the Taylor expansion of p at Q. In addition, the tangents to $\mathcal{C}$ at Q are the lines through Q that cut $\mathcal{C}$ with multiplicity $> r$ at Q. Counting multiplicity, $\mathcal{C}$ has exactly r tangents at Q. In particular:

1. Q is a regular point of $\mathcal{C}$ if $r = 1$.
2. Q is a singular point of $\mathcal{C}$ if $r > 1$.
3. Q is a double point of $\mathcal{C}$ if $r = 2$.
4. Q is a triple point of $\mathcal{C}$ if $r = 3$, etc.

Thus a singularity is also called a multiple point (or an r-fold point) of the (affine) plane complex algebraic curve. Based on Definition 7, we distinguish between ordinary and nonordinary singularities in the following way:

Definition 8. Let $\mathcal{C}$ be an affine plane complex algebraic curve of degree m defined by the squarefree polynomial $p(z, w) \in \mathbb{C}[z, w]$, and $Q(z_0, w_0) \in Sing(\mathcal{C})$. The singularity Q of $\mathcal{C}$ is ordinary if and only if all the r linear factors from Equation (2.17) are different, i.e. all the tangents are different. Otherwise, Q is nonordinary, i.e. some of its corresponding tangents are repeating.

The singular points include multiple points where the curve intersects itself (i.e. double point, triple point, quadruple point as in Figure 2.17) or different types of cusp (i.e. cusp, ramphoid cusp as in Figure 2.18). The points of an affine plane complex algebraic curve that are not singular are called nonsingular or regular points. An irreducible affine plane complex algebraic curve has at most finitely many singular points (in which case it is called a singular affine plane complex algebraic curve), and if it has none it is called nonsingular (or smooth).

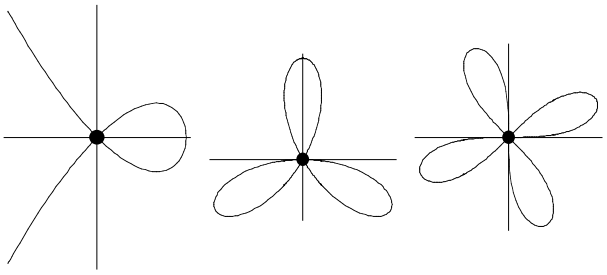


Figure 2.17: Ordinary singularities of some plane algebraic curves. Pictures produced with Mathematica, for more information see [Wolfram, 2000].

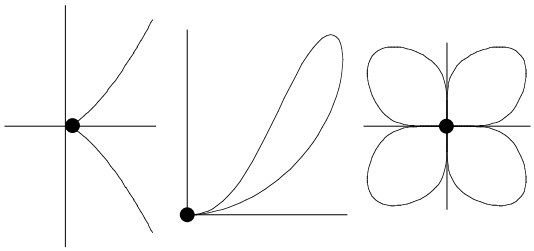


Figure 2.18: Nonordinary singularities of some plane algebraic curves. Pictures produced with Mathematica, see [Wolfram, 2000].

From the previous discussion and definitions, it is clear how we can compute the multiplicity of an affine plane complex algebraic curve $\mathcal{C}$ at its singular point Q and how we can compute the tangents to $\mathcal{C}$ at Q. From this information, we can decide the type of singularity, i.e. ordinary or nonordinary singularity. In Example 4 and Example 5 we familiarize the reader with these computations.

Example 4. We consider the following affine plane complex algebraic curve:

$$\mathcal{C} = \{(z, w) \in \mathbb{C}^2 \mid z^3 - z^2 + w^2 = 0\}.$$

We compute the singularities of $\mathcal{C}$ by solving the system of polynomial equations $z^3 - z^2 + w^2 = 3z^2 - 2z = 2w = 0$ and we obtain $Sing(\mathcal{C}) = \{(0, 0)\}$. We then compute the multiplicity of $\mathcal{C}$ at its singularity $Q(0, 0)$ by extracting the degree of the lowest non-vanishing term in the Taylor expansion of p at Q. In addition, we compute the tangents to $\mathcal{C}$ at Q from equating to zero the lowest non-vanishing term in the Taylor expansion of p at Q. We compute the following Taylor series expansion of p at $Q(0, 0)$:

$$p(0 + zt, 0 + wt) = p(zt, wt) = z^3t^3 + (-z^2 + w^2)t^2. \tag{2.18}$$

We observe that the lowest non-vanishing term in the polynomial from Equation (2.18), which is a polynomial in the indeterminate t and with coefficients z, w, has degree 2. Thus the multiplicity of $\mathcal{C}$ at $Q(0,0)$ is 2. The tangents of $\mathcal{C}$ at $Q(0,0)$ are determined from the following equation:

$$-z^2 + w^2 = 0. \tag{2.19}$$

By replacing $\frac{z}{w} \to x$ in Equation (2.19) and by dividing the equation with $\frac{1}{w^2}$ we obtain the following polynomial equation in the indeterminate x :

$$1 - x^2 = 0.$$

We notice that the greatest common divisor of $1 - x^2$ and its derivative $-2x$ is 1, and therefore the polynomial equation $1 - x^2 = 0$ has no multiple roots. In addition, we observe that $1 - x^2 = (1 - x)(1 + x)$. We obtain two distinct tangents to $\mathcal{C}$ at $Q(0,0)$ with the defining equations $z + w = 0$ and respectively $z - w = 0$. Thus, if we think of the polynomial $q(x) = 1 - x^2$ as a polynomial in one complex variable x, then the equation $q(x) = 0$ has exactly two complex roots $x_1 = 1, x_2 = -1$ counting multiplicity. It follows that there are two distinct tangents to $\mathcal{C}$ at $Q(0,0)$ with the defining equations $w = x_1(z - 0) + 0$ and respectively $w = x_2(z - 0) + 0$, see Figure 2.19.

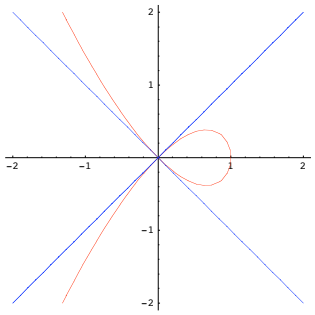


Figure 2.19: Ordinary double point (or node) of the curve (in red) given by $z^3 - z^2 + w^2 = 0$ with two distinct tangents (in blue) $z + w = 0, z - w = 0$. Picture produced with Mathematica, see [Wolfram, 2000] for more information.

Example 5. We consider the affine plane complex algebraic curve:

$$\mathcal{C} = \{(z, w) \in \mathbb{C}^2 \mid z^3 - w^2 = 0\}.$$

We first compute the singularities of $\mathcal{C}$ by solving the system of polynomial equations $z^3 - w^2 = 3z^2 = 2w = 0$ and we obtain $Sing(\mathcal{C}) = \{(0,0)\}$. Secondly, we compute the multiplicity of $\mathcal{C}$ at the singularity $Q(0,0)$ by extracting the degree of the lowest non-vanishing term in the Taylor expansion of p at Q. In addition, we compute the tangents to $\mathcal{C}$ at Q from equating to zero the lowest non-vanishing term in the Taylor expansion of p at Q :

$$p(0 + zt, 0 + wt) = p(zt, wt) = z^3t^3 + (-w^2)t^2. \tag{2.20}$$

We observe that the lowest non-vanishing term in the polynomial from Equation (2.20), which is a polynomial in the indeterminate t and with coefficients z, w, has degree 2. Thus

the multiplicity of $\mathcal{C}$ at $Q(0,0)$ is 2. The tangents of $\mathcal{C}$ at $Q(0,0)$ are determined from the following equation:

$$-w^2 = 0. \tag{2.21}$$

By replacing $\frac{w}{z} \to x$ in Equation (2.21) and by dividing the equation with $\frac{1}{z^2}$ we obtain the following polynomial equation in the indeterminate x :

$$-x^2 = 0.$$

We notice that the greatest common divisor of $-x^2$ and its derivative $-2x$ is $x \neq 1$, and therefore the equation $-x^2 = 0$ has multiple roots $x_1 = x_2 = 0$. It follows that the two tangents to $\mathcal{C}$ at $Q(0,0)$ are equal with the defining equation $w = 0$, see Figure 2.20.

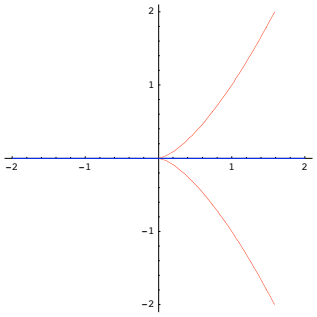


Figure 2.20: Nonordinary double point (cusp) of the curve (in red) given by $z^3 - w^2 = 0$ with two equal tangents (in blue) $w = 0$. Picture produced with Mathematica, see [Wolfram, 2000] for more information.

In the following examples, we present several ordinary and nonordinary singularities of some plane complex algebraic curves. We mention that these examples are mainly taken from the book of [Fulton, 1989]. We remind that for deciding the type of singularity, i.e. ordinary or nonordinary, one has to compute the multiplicity and the number of corresponding tangent lines of the singularity itself, as indicated in Example 4 and in Example 5.

Example 6. In Figure 2.21 we include the ordinary singularities of some plane complex algebraic curves. In the indicated figure, the plane complex algebraic curves are depicted in red, whereas the corresponding tangent lines of their singularities are rendered in black. From left to right we observe the following plane complex algebraic curves, for which we indicate their corresponding singularities, multiplicities and tangent lines:

- the plane complex algebraic curve defined by $z^3 - z^2 + w^2 = 0$. This curve has a singularity in the origin $(0,0)$. From Example 4, we computed the multiplicity m of this singularity and the number of its corresponding tangent lines t and we obtained $m = 2$ and $t = 2$, where the defining equations of the two tangent lines are given by $z + w = 0$, $z - w = 0$. Thus, this singularity is an *ordinary double point*.
- the plane complex algebraic curve defined by $(z^2 + w^2)^2 + 3z^2w - w^3 = 0$. This curve has a singularity in the origin $(0,0)$. By a straightforward computation as in Example 4, we obtain the multiplicity of this singularity to be $m = 3$ and the number of its tangent lines to be $t = 3$, where the three tangent lines have the following defining equations: $w = 0$, $w = z/\sqrt{3}$, $w = -z/\sqrt{3}$. We call this singularity an *ordinary triple point*.

- the plane complex algebraic curve defined by $zw(z^2 - w^2) + (z^2 + w^2)^3 = 0$. This curve has a singularity in the origin $(0, 0)$. By a straightforward computation as in Example 4, we get the multiplicity of this singularity to be $m = 4$ and the number of its tangent lines to be $t = 4$, where the four tangent lines have the following defining equations: $z = 0,\ w = 0,\ z + w = 0,\ z - w = 0$. We notice that this singularity is an *ordinary quadruple point*.

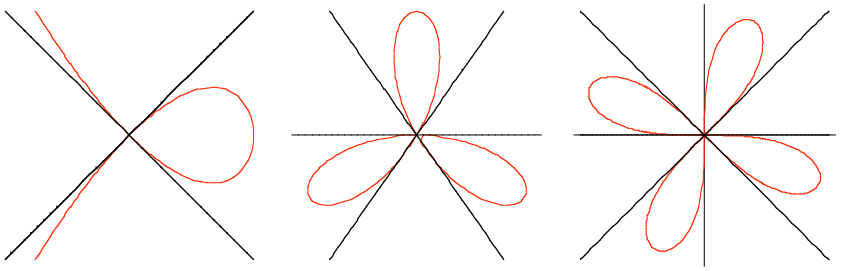


Figure 2.21: Ordinary singularities with their corresponding tangents lines. From left to right: (1) ordinary double point; (2) ordinary triple point; (3) ordinary quadruple point. Pictures produced with Mathematica, see [Wolfram, 2000] for more information.

Example 7. In Figure 2.22 we indicate the nonordinary singularities of different plane complex algebraic curves. In the mentioned figure, the plane complex algebraic curves are drawn in red, whereas the corresponding tangent lines of their singularities are marked in black. From left to right we observe the following plane complex algebraic curves, for which we indicate their corresponding singularities, multiplicities and tangent lines:

- the plane complex algebraic curve defined by $z^3 - w^2 = 0$. This curve has a singularity in the origin $(0, 0)$. From Example 5, we computed the multiplicity m of this singularity and the number of its corresponding tangent lines t and we obtained $m = 2$ and $t = 1$, where the tangent line has the defining equation $w = 0$. Thus, this singularity is a nonordinary double point called also a *cusp*.

- the plane complex algebraic curve defined by $z^4 + z^2w^2 - 2z^2w - zw^2 + w^2 = 0$. This curve has a singularity in the origin $(0, 0)$. By a straightforward computation as in Example 5, we obtain the multiplicity of this singularity to be $m = 2$ and the number of its tangent lines to be $t = 1$, where the tangent line has the defining equation $w = 0$. This singularity is also a nonordinary double point and it is called a *ramphoid cusp*. A ramphoid cusp is a cusp of a curve, which has both branches of the curve on the same side of the common tangent, as we can easily notice from Figure 2.22.

- the plane complex algebraic curve defined by $(z^4 + w^4)^3 - z^2w^2 = 0$. This curve has also a singularity in the origin $(0, 0)$. By a straightforward computation as in Example 5, we get the multiplicity of this singularity to be $m = 4$ and the number of its tangent lines to be $t = 2$, where the two tangent lines have the following defining equations: $z = 0$ and $w = 0$. We call this singularity a *nonordinary quadruple point*.

An important observation is that computing the singularities of an affine plane complex algebraic curve $\mathcal{C} = \{(z, w) \in \mathbb{C}^2 \mid p(z, w) = 0\} \subset \mathbb{C}^2$ defined by a squarefree polynomial $p(z, w) \in \mathbb{C}[z, w]$ is an ill-posed problem. This means that the solution of the problem

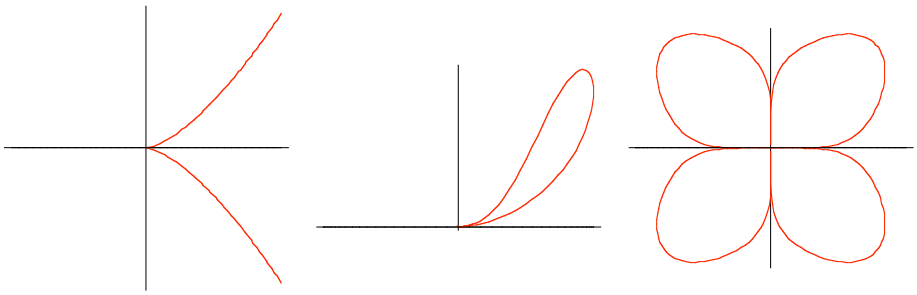


Figure 2.22: Nonordinary singularities with their corresponding tangents lines. From left to right: (1) cusp; (2) ramphoid cusp; (3) nonordinary quadruple point. Pictures produced with Mathematica, see [Wolfram, 2000] for more information.

does not depend continuously on the input data, i.e. the solution is not stable under small changes of the input data. For a better understanding of this situation, we consider a slightly perturbed polynomial $q(z,w)$ of $p(z,w)$ and the affine plane complex algebraic curve $\mathcal{D}$ defined by $q(z,w)$, i.e. $\mathcal{D} = \{(z,w) \in \mathbb{C}^2 \mid q(z,w) = 0\}$. We identify $\mathbb{C}^2$ with $\mathbb{R}^4$ and we intend to visualize the topology (i.e. shape) of the affine plane complex algebraic curves $\mathcal{C}$ and $\mathcal{D}$ around their singularities. As discussed in Remark 2 on drawing techniques for affine plane complex algebraic curves, we cannot draw these real two-dimensional objects in $\mathbb{R}^4$ but we can sketch the equivalent curves in the two-dimensional plane. In Example 8 and Example 9 we illustrate the ill-posedness notion for the problem of computing the singularities of plane complex algebraic curves.

Example 8. We consider the following affine plane complex algebraic curves:

$$\mathcal{C} = \{(z,w) \in \mathbb{C}^2 \mid -z^3 - zw + w^2 = 0\} \text{ and}$$

$$\mathcal{D} = \{(z,w) \in \mathbb{C}^2 \mid -z^3 - zw + w^2 - 0.01 = 0\},$$

defined by the squarefree polynomial $p(z,w) = -z^3 - zw + w^2 \in \mathbb{C}[z,w]$ and its perturbed squarefree version $q(z,w) = -z^3 - zw + w^2 - 0.01 \in \mathbb{C}[z,w]$. We sketch the equivalent affine plane algebraic curves in $\mathbb{R}^2$ denoted with $\mathcal{C}'$ and respectively with $\mathcal{D}'$, i.e.

$$\mathcal{C}' = \{(z,w) \in \mathbb{R}^2 \mid -z^3 - zw + w^2 = 0\} \text{ and}$$

$$\mathcal{D}' = \{(z,w) \in \mathbb{R}^2 \mid -z^3 - zw + w^2 - 0.01 = 0\}.$$

We visualize the topology of $\mathcal{C}'$ and $\mathcal{D}'$. We notice that the singular point $(0,0)$ of the red algebraic curve $\mathcal{C}'$ disappears for small perturbations of its defining polynomial obtaining the smooth blue curve $\mathcal{D}'$, as in Figure 2.23. Thus for small changes in the input data, we obtain huge changes in the output solution. The same situation happens in $\mathbb{R}^4$, but we cannot visualize it.

Example 9. By using the same notations as in Example 8, we consider the following affine plane complex algebraic curves:

$$\mathcal{C} = \{(z,w) \in \mathbb{C}^2 \mid z^3 + z^2 - w^3 = 0\} \text{ and}$$

$$\mathcal{D} = \{(z,w) \in \mathbb{C}^2 \mid z^3 + z^2 - w^3 - 0.1w^2 = 0\},$$

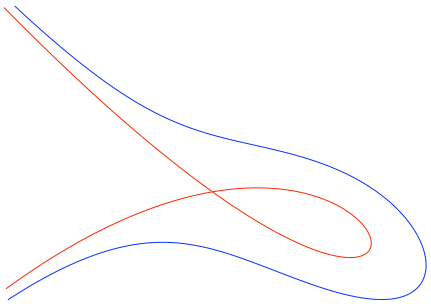


Figure 2.23: Example of ill-posedness of the singularity $(0,0)$ of the red inner curve given by $-z^3 - zw + w^2 = 0$. Picture produced with Axel, see Chapter 5 for more information.

defined by the squarefree polynomial $p(z,w) = z^3 + z^2 - w^3 \in \mathbb{C}[z,w]$ and its perturbed squarefree version $q(z,w) = z^3 + z^2 - w^3 - 0.1w^2 \in \mathbb{C}[z,w]$. We visualize the topology of the equivalent affine plane algebraic curves in $\mathbb{R}^2$ denoted with $\mathcal{C}^{'}$ and respectively with $\mathcal{D}^{'}$, i.e.

$$\mathcal{C}^{'} = \{(z,w) \in \mathbb{R}^2 \mid z^3 + z^2 - w^3 = 0\} \text{ and}$$
$$\mathcal{D}^{'} = \{(z,w) \in \mathbb{R}^2 \mid z^3 + z^2 - w^3 - 0.1w^2 = 0\}.$$

In Figure 2.24 the red inner curve represents the topology of $\mathcal{C}^{'}$, whereas the blue outer curve represents the topology of $\mathcal{D}^{'}$. The curves $\mathcal{C}^{'}$ and $\mathcal{D}^{'}$ have a singularity in the origin, i.e. $\mathcal{C}^{'}$ has a cusp in the origin and $\mathcal{D}^{'}$ has an ordinary double point in the origin. We notice that the singularity of $\mathcal{C}^{'}$ changes its type under small perturbations of its defining polynomial obtaining the singularity of $\mathcal{D}^{'}$. The same situation happens in $\mathbb{R}^4$, but we cannot visualize it.

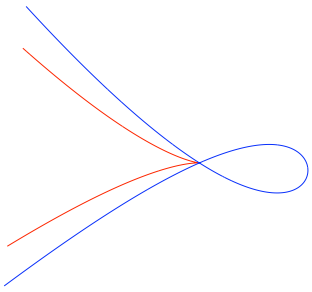


Figure 2.24: Example of ill-posedness of the singularity $(0,0)$ of the red inner curve given by $z^3 + z^2 - w^3 = 0$. Picture produced with Axel, see Chapter 5 for more information.

In the previous definitions we used affine coordinates, referring to the class of singularities of affine plane complex algebraic curves. Still, the definitions are independent of the choice of coordinates and we can extend them for homogeneous coordinates, referring thus to the class of singularities of projective plane complex algebraic curves as follows:

Definition 9. Let $\tilde{\mathcal{C}}$ be a projective plane complex algebraic curve in $\mathbb{P}^2(\mathbb{C})$ defined by the squarefree homogeneous polynomial $p(z,w,u)$, and let $\tilde{Q}(a:b:c)$ be a point on the projective curve $\tilde{\mathcal{C}}$. Then $\tilde{Q}$ is called a simple (or regular) point if $\frac{\partial p}{\partial z}(\tilde{Q}) \neq 0$ or $\frac{\partial p}{\partial w}(\tilde{Q}) \neq 0$ or $\frac{\partial p}{\partial u}(\tilde{Q}) \neq 0$. If $\tilde{Q}$ is not simple, then $\tilde{Q}$ is called a multiple point or a singularity on $\tilde{\mathcal{C}}$.

Let r be such that for all $j+k+l<r$ the partial derivative $\dfrac{\partial^{j+k+l}p}{\partial z^j w^k u^l}$ vanishes at $\tilde{Q}$, but at least one of the partial derivatives or order r does not vanish at $\tilde{Q}$. Then r is called the multiplicity of $\tilde{Q}$ on $\tilde{\mathcal{C}}$ or $\tilde{Q}$ is an r-fold point on $\tilde{\mathcal{C}}$. A projective plane complex algebraic curve with no singularities is called smooth (or nonsingular). If a projective plane complex algebraic curve has singularities, then it is called singular.

From Euler's formula we know that any homogeneous polynomial $p(z,w,u)\in\mathbb{C}[z,w,u]$ of degree m satisfies the equation:

$$m\cdot p(z,w,u)=z\frac{\partial p}{\partial z}(z,w,u)+w\frac{\partial p}{\partial w}(z,w,u)+u\frac{\partial p}{\partial u}(z,w,u).$$

It follows that a point $\tilde{Q}(z:w:u)$ is a singularity of the projective plane complex algebraic curve $\tilde{\mathcal{C}}$ defined by the homogeneous polynomial $p(z,w,u)$ of degree m if Q satisfies the relation $\dfrac{\partial p}{\partial z}(\tilde{Q})=\dfrac{\partial p}{\partial w}(\tilde{Q})=\dfrac{\partial p}{\partial u}(\tilde{Q})=0$. We make the following remark concerning the connection between the singularities of affine and projective plane complex algebraic curves:

Remark 4. We consider $\tilde{\mathcal{C}}$ to be a projective plane complex algebraic curve defined by the squarefree homogeneous polynomial $p(z,w,u)$, and we consider $\tilde{Q}(a:b:c)$ to be a point on the projective curve $\tilde{\mathcal{C}}$. The point $\tilde{Q}$ is a singularity on the projective curve $\tilde{\mathcal{C}}$ depending on whether $Q(a,b)$ is a singularity on the affine curve $\mathcal{C}$ defined by $p(z,w,1)$. Moreover if $\tilde{\mathcal{C}}$ is nonsingular then so is $\mathcal{C}$. However, the converse is not necessarily true: it may happen that $\tilde{\mathcal{C}}$ has singular points even if $\mathcal{C}$ is nonsingular.

2.2.2 Applications of Singularities

Singularities have applications in many different areas such as: catastrophe theory, optimization and control problems, wavefront propagation, etc. For more details on these particular topics, see the detailed book of [Arnold, 2004].

In this subsection, as an example, we will shortly cover the applications of singularities in catastrophe theory, following the books of [Arnold et al., 1985], [Arnold et al., 1998], [Arnold, 2004], [Gilmore, 1981]. In catastrophe theory, a special type of points of a function, which are called degenerate critical points, play an important role. In the sequel, we shortly introduce these types of points. For this purpose, we consider the function $f:\mathbb{R}^n\to\mathbb{R},(x_1,...,x_n)\mapsto f(x_1,...,x_n)$, and the point of local coordinates $D(x_1,...x_n)\in\mathbb{R}^n$. In addition, we assume that the function f in the variables $x_1,...,x_n$ is infinitely differentiable, denoted by $f(x_1,...,x_n)\in C^\infty(\mathbb{R}^n)$. We remember that a function is said to be infinitely differentiable (or smooth or of class C^∞) if it has derivatives of all orders. First of all, we define the gradient of the function f at the point D as follows:

$$(\nabla f)(D)=\Big(\frac{\partial f}{\partial x_1}(D),...,\frac{\partial f}{\partial x_n}(D)\Big). \tag{2.22}$$

Secondly, we define the Hessian matrix of the function f at the point D denoted by $Hf(D)$ as the Jacobian matrix of the derivatives $\dfrac{\partial f}{\partial x_1},...,\dfrac{\partial f}{\partial x_n}$ of the function $f(x_1,...,x_n)$ with

respect to the variables $x_1, ..., x_n$ at the point D, i.e.:

$$Hf(D) = \begin{pmatrix} \frac{\partial f^2}{\partial x_1^2}(D) & \frac{\partial f^2}{\partial x_1 x_2}(D) & ... & \frac{\partial f^2}{\partial x_1 x_n}(D) \\ ... & ... & ... & ... \\ \frac{\partial f^2}{\partial x_n x_1}(D) & \frac{\partial f^2}{\partial x_n x_2}(D) & ... & \frac{\partial f^2}{\partial x_n^2}(D) \end{pmatrix}. \tag{2.23}$$

Under these assumptions, we are now prepared to introduce the following types of points for the function f :

- we say that D is a critical point for f if the gradient of f at D is identically zero, i.e. $(\nabla f)(D) = 0$. Otherwise, we say that D is a regular point of f, i.e. D is a regular point of f if there exists $j \in \{1, ..., n\}$ such that $\frac{\partial f}{\partial x_j}(D) \neq 0$. Furthermore, we distinguish between the following types of critical points:
  - we say that D is a non-degenerate critical point (or a Morse critical point) of f if the gradient of f at D is identically zero, i.e. $(\nabla f)(D) = 0$, and the Hessian of f at D is non-singular, i.e. $\det\big(Hf(D)\big) \neq 0$.
  - we say that D is a degenerate critical point (or a catastrophe point or simply a catastrophe) of f if the gradient of f at D is identically zero, i.e. $(\nabla f)(D) = 0$, and the Hessian of f at D is singular, i.e. $\det\big(Hf(D)\big) = 0$.

We notice that the non-degenerate and the degenerate critical points (also called catastrophes) are found at points where the gradient vanishes. However, a degenerate critical point (or a catastrophe) differs from a non-degenerate critical point in the following way: a catastrophe has a degenerate (or singular) Hessian matrix, i.e. the determinant of the Hessian matrix vanishes.

We recall that a diffeomorphism is a smooth invertible function with smooth inverse. We are now ready to define the notion of structural equivalence of two smooth functions at a given point. We say that two smooth functions $f_1, f_2 \in \mathbb{C}^\infty$ are locally structural equivalent at a point $D \in \mathbb{R}^n$ if there exists a neighborhood V of the point D and a diffeomorphism ϕ such that a change of the coordinate system for one function with this diffeomorphism causes the two functions to be equal in V, i.e. $f_1, f_2 \in \mathbb{C}^\infty$ are locally structural equivalent at $D \in \mathbb{R}^n$ if there exists V a neighborhood of D and a diffeomorphism $\phi : \mathbb{R}^n \to \mathbb{R}^n$ with the property that $(f_1 \circ \phi)(x) = f_2(x)$, for all $x \in V$. For an intuitive illustration of this notion see Example 10. Informally, if two smooth functions are locally structural equivalent at a given point D, then this fact implies that the two functions have the same local topological structure (or type) at the given point.

Example 10. This example is from [Arnold et al., 1998, p. 12]. We consider the two smooth functions $f_1 : \mathbb{R}_{>0} \to \mathbb{R}_{>0}, x \mapsto f_1(x) = x^2$ and $f_2 : \mathbb{R}_{>0} \to \mathbb{R}_{>0}, x \mapsto f_2(x) = cx^2$, with $c \neq 0$. We notice that the two functions f_1, f_2 are locally structural equivalent at the non-degenerate critical point $x = 0$, since $f_1 \circ g = f_2$, where g is the diffeomorphism defined by $g(x) = \sqrt{c}x$.

By Morse lemma [Arnold et al., 1985, p. 119] we know that a non-degenerate critical point is stable. This means that a small perturbation of the function will not modify the local topological structure of the function at its non-degenerate critical point. Moreover, the Morse lemma indicates us that at the non-degenerate critical points the local topological

Table 2.1: List of *A*-*D*-*E* singularities from [Arnold et al., 1985]

$A_k,\ k \geq 1$	$D_k,\ k \geq 4$	E_6	E_7	E_8
$x^{k+1} + y^2$	$x^2y + y^{k-1}$	$x^3 + y^4$	$x^3 + xy^3$	$x^3 + y^5$

structure of the function is given by the quadratic part of the Taylor series expansion of the function itself.

On the contrary, degenerate critical points are not stable. It follows that a small perturbation of the function will produce a huge change in the local topological structure of the function in its degenerate critical points. Furthermore, a higher order Taylor series expansion is required for describing the local topological structure of the given function at its non-degenerate critical points.

The main observation is that small perturbations of a given function preserve its local topological structural at its non-degenerate critical points, but they produce changes in the local topological structure of the function at its degenerate critical points. An important result, i.e. the Thom splitting lemma [Gilmore, 1981, p. 9] says that for a smooth function f with one or more degenerate critical points there exists a smooth change of coordinates such that the function f can be split into two functions: (1) one function containing the non-degenerate critical points of f, which represents a Morse function, i.e. a function for which all the critical points are non-degenerate; (2) and another function containing the degenerate critical points of f, which represents a non-Morse function. Furthermore, the Taylor series expansion of this non-Morse function begins with at least third-degree terms. This non-Morse function is the main topic of research in basic catastrophe theory. In addition, according to Thom classification theorem [Gilmore, 1981], [Arnold et al., 1985, p. 191] under a smooth change of coordinates this non-Morse function splits into two other functions: (2.1) one function, which contains the degenerate critical point (or non-Morse critical points or catastrophes) and which depends only on the variables of f. These variables are in fact called the state variables. In the literature, this function is called the catastrophe germ. For the purpose of this subsection, we do not explain the notion of a germ in particular, but we accept it as part of our terminology; (2.2) and another function, which indicates how the function behaves at its degenerate critical point under the most general perturbation. This function is called the perturbation function and it depends both on the state variables and on one or more control parameters. A slight change in the control parameters produces a huge change in the local topological structure of the function itself. In fact, this perturbation function indicates how the non-Morse function changes its local structure in a neighborhood of its degenerate critical points when the control parameters are changing.

A list of fundamental catastrophes was given by Thom in 1975. Still a complete list of fundamental catastrophes is presented in the book of [Arnold et al., 1985, p. 246]. In this book, the author classifies the critical points of smooth functions by introducing the notion of equivalence class of a function germ at a critical point, which is called a singularity. We recall here, in Table 2.1, only the *A*-*D*-*E* list of singularities, by indicating the catastrophe germ of each type of singularity. From the table, we observe that the catastrophe germ of the A_k singularity depends on one state variable x, whereas the catastrophe germs of the D_k, E_6, E_7 and E_8 singularities depend on two state variables x and y. The names of the *A*-*D*-*E* singularities come from the Lie groups of type A, D, E since there is a close connection between the classification of elementary catastrophes and the classification of simple Lie algebras as indicated in [Arnold et al., 1985, p.184].

For a better understanding of the notion of catastrophe, in Figure 2.25 we illustrate an

example of a fold catastrophe from [Gilmore, 1981]. For this purpose, we consider the A_2 singularity with the following equation of its catastrophe germ that depends on one state variable x:

$$f(x) = x^3.$$

The fold catastrophe is given by the following equation:

$$f(x, c) = x^3 + cx,$$

where $f(x) = x^3$ represents the catastrophe germ, which depends on one state variable x, and $p(x, c) = cx$ represents the perturbation function, which depends on one state variable x and one control parameter c.

From Figure 2.25 we notice that for different values of the control parameter c, the function $f(x, c)$ changes its local structure, i.e.: (1) for $c = 0$, $f(x, c)$ has a doubly degenerate critical point at $x = 0$; (2) for $c > 0$, $f(x, c)$ has no critical point at $x = 0$; (3) and for $c < 0$, $f(x, c)$ has two isolated non-degenerate critical points at $x_1 = \sqrt{-c}$ and $x_2 = -\sqrt{-c}$. We recall that a point is an isolated critical point if there is a neighborhood V of p such that there is no other critical point in $V \setminus \{p\}$. Thus we observe that the perturbations $f(x, c) = x^3 + cx$ of the function $f(x) = x^3$ with a doubly degenerate critical point in $x = 0$ either split the critical points in two non-degenerate critical points for $c < 0$ or they annihilate the critical points completely for $c > 0$.

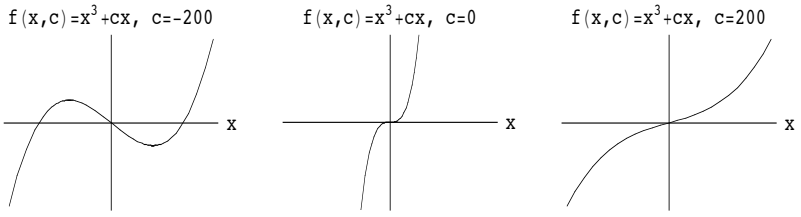


Figure 2.25: Fold catastrophe: perturbations of type $f(x, c) = x^3+cx$ of the function $f(x) = x^3$ with doubly degenerate critical points at $x = 0$ cause changes in the local topological structure of the function $f(x)$ itself either by splitting the degenerate critical points in two non-degenerate critical points when $c < 0$ or by totally annihilating the degenerate critical points when $c > 0$. Picture produced with Mathematica, see [Wolfram, 2000].

We end this subsection with making the following important remarks:

- We notice that there is a correspondence between the singularities of plane complex algebraic curves defined in Subsection 2.2.1 and the critical points of smooth functions defined in this subsection. If we consider the smooth function $f : \mathbb{R}^2 \to \mathbb{R}, (x, y) \mapsto f(x, y)$ with the point $D(a, b) \in \mathbb{R}^2$, then the set $f^{-1}(0) = \{(x, y) \in \mathbb{R}^2 \mid f(x, y) = 0\}$ defines the plane algebraic curve $\mathcal{C} \subset \mathbb{R}^2$. In addition, if $f(D) = 0$, the derivatives of f with respect to x and y are both zero in the point D and f is squarefree, then the point $D(a, b)$ is an isolated singularity of $\mathcal{C}$, i.e. $f(D) = \dfrac{\partial f}{\partial x}(D) = \dfrac{\partial f}{\partial y}(D) = 0$.

- Moreover we observe that a singular point is also a critical point, but the converse statement is not necessarily true, i.e. not all the critical points are singular points. For instance, we consider the smooth function $f : \mathbb{R}^2 \to \mathbb{R}, (x, y) \mapsto x^3 - y^2 + x^2$. We notice that the function f has two non-degenerate critical points in $(0, 0)$ and in $(-\dfrac{2}{3}, 0)$ since these two points are the solutions of the system $3x^2 + 2x = 2y = 0$,

which is formed by the two partial derivatives of f with respect to x and y, and $(Hf)(0,0) \neq 0, (Hf)(-\frac{2}{3},0) \neq 0$. Furthermore, we see that the set $f^{-1}(0) = \{(x,y) \in \mathbb{R}^2 \mid x^3 - y^2 + x^2 = 0\}$ defines the plane algebraic curve $\mathcal{C} \subset \mathbb{R}^2$. We remark that the non-degenerate critical point $(0,0)$ is also a singularity for $\mathcal{C}$ since $f(0,0) = \frac{\partial f}{\partial x}(0,0) = \frac{\partial f}{\partial y}(0,0) = 0$. However, the non-degenerate critical point $(-\frac{2}{3},0)$ is not a singularity of $\mathcal{C}$ since $f(-\frac{2}{3},0) \neq 0$, i.e. this point is not on the curve.

Thus we can notice that there exist clear connections between singularity theory as introduced in Subsection 2.2.1 and catastrophe theory as shortly discussed in this subsection. This is the main reason for which in this subsection we made a short survey concerning catastrophe theory. In this thesis, we will thoroughly report on an important application of singularity theory in algebraic geometry, for more details see Section 2.3, Section 2.4, Section 2.5 and Chapter 3.

2.3 Topology of Plane Complex Algebraic Curves

2.3.1 Preliminaries

In this subsection, we briefly recall some fundamental aspects from the broad field of topology following [Lee, 2000], which are needed for the purpose of this thesis. The goal of this subsection is thus to review important concepts from topology. For a deeper introspection into the captivating field of topology, the reader is advised to consult [Jänich, 1984], [Mumkres, 2000].
We first introduce the Euclidean spaces, since most of topology is modelled from the behaviour of Euclidean spaces. We define the Euclidean space in the following way:

Definition 10. The Cartesian product $\mathbb{R}^n = \mathbb{R} \times \mathbb{R} \times ... \times \mathbb{R}$ of n copies of the real line is called the n-dimensional Euclidean space. Thus $\mathbb{R}^n$ is the set of ordered n-tuples of real numbers. A point (or vector) in $\mathbb{R}^n$ is denoted by $(x_1, ..., x_n)$ or simply by x. The numbers x_k, for any $k \in \{1, ..., n\}$, are called the components (or coordinates) of x. The 0-dimensional Euclidean space $\mathbb{R}^0$ is by convention the singleton $\{0\}$.

The n-dimensional Euclidean space is sometimes called Cartesian space or simply n-space. In the following remark, we recall the main properties of the n-dimensional Euclidean space:

Remark 5. The Euclidean space $\mathbb{R}^n$ has the following properties:

1. $\mathbb{R}^n$ is an n-dimensional real vector space with the usual operations of scalar multiplication and vector addition. The elements of $\mathbb{R}^n$ are called n-vectors.

2. The geometric properties of $\mathbb{R}^n$ are derived from the Euclidean dot product. We define the dot product on $\mathbb{R}^n$ in the following way:

$$x \cdot y = \sum_{i=1}^{n} x_i y_i = x_1 y_1 + ... + x_n y_n, \text{ for any } x, y \in \mathbb{R}^n.$$

It follows that $(\mathbb{R}^n, \cdot)$ is an inner space over $\mathbb{R}$.

3. In particular, we define the Euclidean norm (or length) of a vector $x \in \mathbb{R}^n$ as follows:

$$||x|| = \sqrt{x \cdot x} = \sqrt{x_1^2 + ... + x_n^2}, \text{ for any } x \in \mathbb{R}^n.$$

   Consequently, $(\mathbb{R}^n, ||\cdot||)$ is a normed space over $\mathbb{R}$.

4. In particular, we define the Euclidean metric (or distance) on $\mathbb{R}^n$:

$$d(x, y) = ||x - y|| = \sqrt{(x_1 - y_1)^2 + ... + (x_n - y_n)^2}, \text{ for any } x, y \in \mathbb{R}^n.$$

   In this setting, $(\mathbb{R}^n, \cdot)$ is a metric space over $\mathbb{R}$.

5. The angle between two non-zero vectors x and y is determined by the following formula $cos^{-1}(\frac{x \cdot y}{||x|| \cdot ||y||})$.

6. Given two points $x, y \in \mathbb{R}^n$, the line segment between them is the set $\{tx + (1-t)y : 0 \leq t \leq 1\}$.

7. The notions of continuity and convergence in Euclidean spaces are defined in the usual ways. A map $f : U \to V$ between subsets of Euclidean spaces is continuous if for any $x \in U$ and any $\epsilon > 0$ there exists $\delta > 0$ such that for all y, $||x - y|| < \delta$ implies $||f(x) - f(y)|| < \epsilon$. We refer to this definition as the ϵ-δ definition for continuity of functions.

8. A sequence $(x_k)_k$ of points in $\mathbb{R}^n$ converges to $x \in \mathbb{R}^n$ if for any $\epsilon > 0$ there exists N such that $k \geq N$ implies $||x_k - x|| < \epsilon$.

We next familiarize the reader with metric spaces, which are generalizations of Euclidean spaces. It is helpful to specify that in metric spaces none of the vector space properties are true and only the distance function is preserved. In fact for our purpose, metric spaces and their properties represent the motivation for introducing topological spaces. We define a metric space as follows:

Definition 11. Let M be any set. A metric on M is a function $d : M \times M \to \mathbb{R}$, also called a distance function, satisfying the following three properties:

1. For all $x, y \in M$, $d(x, y) \geq 0$ and $d(x, y) = 0 \Leftrightarrow x = y$ (positivity).
2. For all $x, y \in M$, $d(x, y) = d(x, y)$ (symmetry).
3. For all $x, y, z \in M$ $d(x, z) \leq d(x, y) + d(y, z)$ (triangle inequality).

The pair (M, d) is called a metric space.

We give some simple examples of metric spaces, see Example 11.

Example 11. The following pairs are metric spaces:

1. If M is any subset of $\mathbb{R}^n$, then the function $d(x, y) = ||x - y||$ is a metric on M called the Euclidean metric. Thus the pair $(M, ||\cdot||)$ is a metric space, for any $M \subseteq \mathbb{R}^n$. In this thesis, we make the following convention: whenever we consider a subset of $\mathbb{R}^n$, we assume that the subset is being given together with the Euclidean metric.
2. If (M, d) is a metric space and $X \subset M$, then X inherits a metric by restricting d to $X \times X$.

We now give some standard definitions in metric spaces:

Definition 12. Let (M, d) be a metric space with d the metric function on M.

- For any $x \in M$ and $r > 0$ the open ball of radius r around x is the set

$$B_r(x) = \{y \in M : d(y, x) < r\},$$

  and the closed ball of radius r around x is

$$\bar{B}_r(x) = \{y \in M : d(x, y) \leq r\}.$$

- Given a subset $A \subset M$, a point $x \in M$ is said to be a limit point (or accumulation point or cluster point) of A if every open ball around x contains a point of A other than x.
- A set $A \subset M$ is open if it contains an open ball around each of its points.
- A set $A \subset M$ is closed if it contains all its limit points. The closure of a set $A \subset M$ consists of all the points in A plus its limit points.
- The diameter of a set $A \subset M$ is $sup\{d(x, y) : x, y \in A\}$, which may be infinite. A set $A \subset M$ is said to be bounded if it has finite diameter.

The notions of continuity and convergence in metric spaces are generalizations of the Euclidean definitions:

Definition 13. If $(M_1, d_1), (M_2, d_2)$ are metric spaces, then a map $f : M_1 \to M_2$ is continuous if for every $x \in M_1$ and every $\epsilon > 0$, there exists $\delta > 0$ such that for all y, $d_1(x, y) < \delta$ implies $d_2\big(f(x) - f(y)\big) < \epsilon$. If $(x_k)_k$ is a sequence of points in a metric space (M, d), then it is said to converge to $x \in M$, written $x_k \to x$ or $\lim\limits_{k\to\infty} x_k = x$, if for any $\epsilon > 0$ there exists N such that $k \geq N$ implies $d(x_k, x) < \epsilon$.

We define the notion of compactness in metric spaces as follows:

Definition 14. Let (M, d) be a metric space and $K \subset M$. An open cover of K is a collection of open subsets of M whose union contains K. A subcover is a subcollection that is still an open cover of K. A subset of M is said to be compact if every open cover has a finite subcover.

The following proposition is a direct consequence of Definition 14 of compacts sets:

Proposition 1. *Any compact subset of a metric space is closed and bounded.*

In the n-dimensional Euclidean space, the converse of Proposition 3 is true and it is called the Heine-Borel theorem:

Theorem 1. *Every closed and bounded subset of $\mathbb{R}^n$ is compact, i.e. every subset of $\mathbb{R}^n$ is compact if and only if it is closed and bounded.*

An important property of compactness is given by the following theorem:

Theorem 2. *If M and N are metric spaces, $f : M \to N$ is continuous, and $K \subset M$ is compact, then $f(K)$ is compact.*

The following lemma, called the open set criterion for continuity [Lee, 2000, p. 350], basically proves that continuous functions between metric spaces can be identified knowing only the open sets:

Lemma 1. *A map $f : M_1 \to M_2$ between metric spaces is continuous if and only if the inverse image of every open set is open, i.e. whenever U is an open subset of M_2, then $f^{-1}(U)$ is open in M_1.*

Remark 6. From Lemma 1 we deduce that a function between metric spaces is continuous in the sense specified in the open set criterion for continuity if and only if it is continuous in the sense specified in the ϵ-δ definition for continuity of functions.

Actually, Lemma 1 is the main reason for introducing topological spaces, which are abstract spaces without distances in which continuous functions are defined. We define topological spaces in the following way:

Definition 15. A topology on a set X is a collection $\mathcal{T}$ of subsets of X, called open sets, satisfying the following properties:

1. X and $\emptyset$ are elements of $\mathcal{T}$;
2. $\mathcal{T}$ is closed under finite intersections, i.e. if $U_1, ..., U_n \in \mathcal{T}$, then their intersection $U_1 \cap U_2 \cap ... \cap U_n \in \mathcal{T}$;
3. $\mathcal{T}$ is closed under arbitrary unions, i.e. if $\{U_\alpha\}_{\alpha \in A}$ is any (finite or infinite) collection of elements of $\mathcal{T}$, then their union $\cup_{\alpha \in A}$ is in $\mathcal{T}$.

In addition, a pair $(X, \mathcal{T})$ consisting of a set X and a topology $\mathcal{T}$ on X is called a topological space. The sets of $\mathcal{T}$ are called open sets.

We include several examples of topological spaces, see Example 12.

Example 12. The following are topological spaces:

1. Any set X, with $\mathcal{T} = \{\emptyset, X\}$. This is called the trivial topology on X.
2. Any metric space (M, d), with $\mathcal{T}$ being equal to the collection of all open subsets of M. This is called the metric topology on M.

In this thesis, we use the following convention: if the topology is understood from the context, then we will omit it from the notation and simply refer to X as a topological space. For our purpose, we work with the topological spaces indicated in Example 13.

Example 13. The following are topological spaces:

1. The real Euclidean space $\mathbb{R}^n$ and the complex space $\mathbb{C}^n$ defined as:

$$\mathbb{R}^n = \big\{(x_1, ..., x_n) \mid x_k \in \mathbb{R} \text{ for all } k \in \{1, ..., n\}\big\},$$

$$\mathbb{C}^n = \big\{(z_1, ..., z_n) \mid z_k \in \mathbb{C} \text{ for all } k \in \{1, ..., n\}\big\},$$

where we consider $z_k = x_k + iy_k$, with $x_k, y_k \in \mathbb{R}$, for all $k \in \{1, ..., n\}$. These spaces are provided with the Euclidean metric denoted by $||\cdot||$ and with the topology induced by the Euclidean metric. Moreover we observe that $\mathbb{C}^n$ is isomorphic to $\mathbb{R}^{2n}$.

2. The n-dimensional sphere in $\mathbb{R}^{n+1}$, the n-dimensional open ball in $\mathbb{R}^n$, and the n-dimensional closed ball in $\mathbb{R}^n$, which are all centered in the origin, have radius 1, and they are defined as follows:

$$S_1^n(\vec{0}_{n+1}) = \{(x_1, ..., x_{n+1}) \mid x_1^2 + x_2^2 + ... + x_{n+1}^2 = 1\} \subset \mathbb{R}^{n+1},$$

$$B_1^n(\vec{0}_n) = \{(x_1, ..., x_n) \mid x_1^2 + x_2^2 + ... + x_{n+1}^2 < 1\} \subset \mathbb{R}^n,$$

$$\overline{B_1^n}(\vec{0}_n) = \{(x_1, ..., x_n) \mid x_1^2 + x_2^2 + ... + x_{n+1}^2 \leq 1\} \subset \mathbb{R}^n,$$

where we denote by $\vec{0}_{n+1}$ the origin vector in $\mathbb{R}^{n+1}$ and by $\vec{0}_n$ the origin vector in $\mathbb{R}^n$. We observe that $S_1^n(\vec{0}_{n+1})$ is a compact topological space, whereas $B_1^n(\vec{0}_n)$, $\mathbb{R}^n$ and $\mathbb{C}^n$ are not compact topological spaces. We make the following remark concerning the terminology used for a n-dimensional ball in $\mathbb{R}^n$: the notion of "disk" is usually the generalization to the metric space of the ball from the Euclidean space. Still, in the literature depending on the authors disk is sometimes used to mean ball. In this thesis, we use the notion of a n-dimensional ball in the Euclidean space $\mathbb{R}^n$. In Figure 2.26 we include an example of a sphere in the 3-dimensional Euclidean space. In addition, the interior of this sphere is a ball in the 3-dimensional Euclidean space.

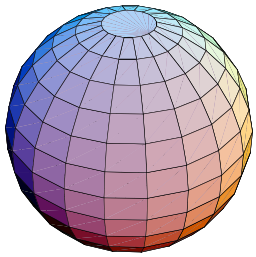


Figure 2.26: Example of a sphere in the 3-dimensional Euclidean space. The interior of this sphere is a ball in the 3-dimensional Euclidean space. Picture produced with Mathematica, see [Wolfram, 2000] for more information.

3. The n-dimensional torus constructed in the following way: we assume $X_1, ..., X_n$ to be topological spaces and we define a basis in their Cartesian product as $\mathcal{B} = \{U_1 \times ... \times U_n \mid U_k \text{ is open in } X_k, \text{ for all } k \in \{1, ..., n\}\}$. The topology generated by $\mathcal{B}$ is called product topology. The space $X_1 \times ... \times X_n$ together with the product topology is called a product space. The basis open sets of the form $U_1 \times ... \times U_n$ are called product open sets. We notice that the product topology on $\mathbb{R}^n = \underbrace{\mathbb{R} \times ... \times \mathbb{R}}_{n \text{ times}}$ is the same as the metric topology induced by the Euclidean distance function. Furthermore, we consider S^1 to be the unit circle in the 2-dimensional Euclidean plane, i.e. a 1-dimensional sphere. Then the product space of n unit circles denoted by $\mathbb{T}^n = \underbrace{S^1 \times ... \times S^1}_{n \text{ times}}$ is called an n-dimensional torus. As an example, in Figure 2.27 we visualize a torus in the 3-dimensional Euclidean space $\mathbb{R}^3$.

   In the following we introduce the implicit and the parametric equations of a torus in the 3-dimensional Euclidean space. We consider R to be the radius from the center of the hole to the center of the torus tube, and we consider r to be the radius of the

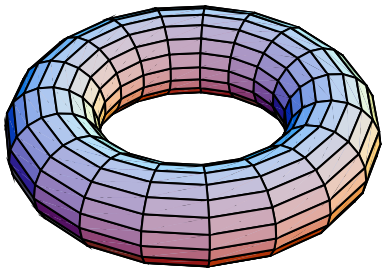


Figure 2.27: Example of a torus in the 3-dimensional Euclidean space. Picture produced with Mathematica, see [Wolfram, 2000] for more information.

tube. With these notations, the equation of a torus in $\mathbb{R}^3$ in Cartesian coordinates x, y, z is the following:

$$(R - \sqrt{x^2 + y^2})^2 + z^2 = r^2, \tag{2.24}$$

whereas the parametric equations are the following:

$$x = (R + r\cos v)\cos u \tag{2.25}$$
$$y = (R + r\cos v)\sin u \tag{2.26}$$
$$z = r\sin v, \tag{2.27}$$

where $u, v \in [0, 2\pi)$.

Moreover, depending on the relative sizes of the two radii R and r of the torus, we distinguish between the following types of tori in the 3-dimensional space:

- if $R > r$, then we obtain a ring torus;
- if $R = r$, then we get a horn torus;
- if $R < r$, then we obtain a spindle torus. In Figure 2.28 we include three examples for each type of torus. In this thesis we make the general convention concerning the types of tori in the 3-dimensional space: if no special explanation is given, then by a torus we mean a ring torus.

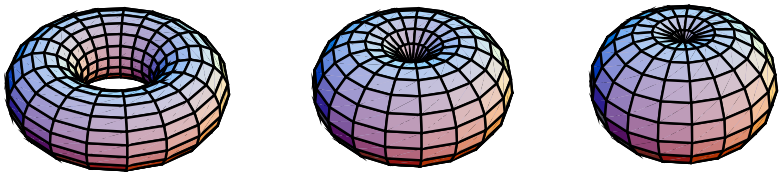


Figure 2.28: Example of different types of tori in the 3-dimensional Euclidean space. From left to right: (1) Ring torus; (2) Horn torus; (3) Spindle torus. Pictures produced with Mathematica, see [Wolfram, 2000] for more information.

4. The n-dimensional complex projective space $\mathbb{P}^n(\mathbb{C})$, which we introduce in details in the next paragraphs following [Kirwan, 1992, p. 36].

We define the n-dimensional complex projective space denoted by $\mathbb{P}^n(\mathbb{C})$ (or simply by $\mathbb{P}^n$) as the set of 1-dimensional complex subspaces of the complex vector space $\mathbb{C}^{n+1}$. If $n = 1$, then we obtain the complex projective line $\mathbb{P}^1$, and if $n = 2$, then we get the complex projective plane $\mathbb{P}^2$.

A 1-dimensional linear subspace V of $\mathbb{C}^{n+1}$ is spanned by any non-zero vector $v \in V$. Thus we can define $\mathbb{P}^n$ as the set of equivalence classes for the equivalence relation denoted by $\sim$, and which is defined in the following way: $z \sim w$ if there exists some $\lambda \in \mathbb{C} \setminus \{0\}$ such that $z = \lambda w$.

Therefore we can introduce the more formal definition for $\mathbb{P}^n$ as follows:

Definition 16. Since any non-zero vector $(z_0, ..., z_n)$ from $\mathbb{C}^{n+1}$ is an element of $\mathbb{P}^n$, we call $(z_0, ..., z_n)$ the homogeneous coordinates of z, where we use the notation $z = (z_0 : z_1 : ... : z_n)$ for denoting the equivalence class defined as:

$$(z_0 : z_1 : ... : z_n) = (w_0 : w_1 : ... : w_n)$$

if there exists some $\lambda \in \mathbb{C} \setminus \{0\}$ such that $z_j = \lambda w_j$ for all $j \in \{0, ..., n\}$. Under these assumptions we define the n-dimensional complex projective space $\mathbb{P}^n$ in the following way:

$$\mathbb{P}^n = \big\{(z_0 : z_1 : ... : z_n) \mid (z_0, ..., z_n) \in \mathbb{C}^{n+1} \setminus \{\vec{0}_{n+1}\}\big\}, \tag{2.28}$$

where we denote by $\vec{0}_{n+1}$ the origin vector in $\mathbb{C}^{n+1}$.

We notice that the definition of the n-dimensional complex projective space is the generalization for the definition of the complex projective plane $\mathbb{P}^2$ as introduced in Subsection 2.1.2. We next make $\mathbb{P}^n$ a topological space. We define the following function:

$$\begin{aligned} &\Pi : \mathbb{C}^{n+1} \setminus \{\vec{0}_{n+1}\} \to \mathbb{P}^n, \\ &(z_0, z_1, ..., z_n) \mapsto (z_0 : z_1 : ... : z_n). \end{aligned} \tag{2.29}$$

We observe that the function Π is continuous. We give $\mathbb{P}^n$ the quotient topology induced from the usual topology on $\mathbb{C}^{n+1} \setminus \{\vec{0}_{n+1}\}$, i.e. we say that $A \subset \mathbb{P}^n$ is open if $\Pi^{-1}(A) \subset \mathbb{C}^{n+1} \setminus \{\vec{0}_{n+1}\}$ is open.

Remark 7. We make the following important observations: (1) for X a topological space, the function $f : \mathbb{P}^n \to X$ is continuous if and only if the composition $f \circ \Pi : \mathbb{C}^{n+1} \setminus \{\vec{0}_{n+1}\} \to X$ is continuous; (2) and more generally for $A \subset \mathbb{P}^n$, the function $f : A \to X$ is continuous if and only if the composition $f \circ \Pi : \Pi^{-1}(A) \to A$ is continuous.

We next define the subsets $U_0, ..., U_j \subset \mathbb{P}^n$, for all $j \in \{0, ..., n\}$ as follows:

$$U_j = \{(z_0 : z_1 : ... : z_n) \in \mathbb{P}^n \mid z_j \neq 0\} \subset \mathbb{P}^n. \tag{2.30}$$

We notice that for all $j \in \{0, ..., j\}$ the following subset:

$$\Pi^{-1}(U_j) = \{(z_0, ..., z_n) \subset \mathbb{C}^{n+1} \mid z_j \neq 0\} \subset \mathbb{C}^{n+1} \setminus \{\vec{0}_{n+1}\} \tag{2.31}$$

is open. Thus we obtain that $U_j \subset \mathbb{P}^n$ is open. We next define the following function:

$$\begin{aligned} &\phi_0 : U_0 \to \mathbb{C}^n, \\ &(z_0 : z_1 : ... : z_n) \mapsto \left(\frac{z_1}{z_0}, ..., \frac{z_n}{z_0}\right), \end{aligned} \tag{2.32}$$

with its inverse:

$$\begin{aligned} &\phi_0^{-1} : \mathbb{C}^n \to U_0, \\ &(w_1, ..., w_n) \mapsto (1 : w_1 : ... : w_n). \end{aligned} \tag{2.33}$$

By using Remark 7, we notice that the function $\phi_0 : U_0 \to \mathbb{C}^n$ is continuous since the composition of ϕ_0 with the function $\Pi : \Pi^{-1}(U_0) \to U_0$ is continuous. Using a similar argument, we remark that the inverse ϕ_0^{-1} is also continuous because it is the composition of Π with the following continuous function:

$$\begin{aligned} &g : \mathbb{C}^n \to \mathbb{C}^{n+1} \setminus \{\vec{0}_{n+1}\}, \\ &(w_1, ..., w_n) \mapsto (1, w_1, ..., w_n). \end{aligned} \tag{2.34}$$

We thus obtain that ϕ_0 is a continuous function with a continuous inverse, which is in fact the definition for the homeomorphism as we will introduce it later in this subsection. In the same way, for all $j \in \{1, ..., n\}$ there exists the following homeomorphisms:

$$\begin{aligned} &\phi_j : U_j \to \mathbb{C}^n, \\ &(z_0 : z_1 : ... : z_n) \mapsto \left(\frac{z_0}{z_j}, ..., \frac{z_{j-1}}{z_j}, \frac{z_{j+1}}{z_j}, ..., \frac{z_n}{z_j}\right). \end{aligned} \tag{2.35}$$

We observe that the complement of U_n in $\mathbb{P}^n$ is the following hyperplane:

$$\{(z_0 : z_1 : ... : z_n) \in \mathbb{P}^n \mid z_n = 0\},$$

which can be identified with $\mathbb{P}^{n-1}$. We remember that a hyperplane is any codimension-1 vector subspace V of a vector space W. If we assume that the dimension of V is m and the dimension of W is n, then the codimension of V is defined as $n - m$. It follows that we can construct the complex projective spaces $\mathbb{P}^n$ inductively: (i) $\mathbb{P}^0$ is the single point; (ii) $\mathbb{P}^1$ is $\mathbb{C}$ together with the point ∞, which is a copy of $\mathbb{P}^0$. Therefore $\mathbb{P}^1$ can be identified with the Riemann sphere $\mathbb{C} \cup \{\infty\}$, which we will discuss later in this subsection; (iii) $\mathbb{P}^2$ is $\mathbb{C}^2$ together with a line at infinity, which is a copy of $\mathbb{P}^1$; (iv) $\mathbb{P}^n$ is $\mathbb{C}^n$ together with a copy of $\mathbb{P}^{n-1}$ at infinity. We also notice that $\{U_j \mid 0 \leq j \leq n\}$ is an open cover of $\mathbb{P}^n$ with $\phi_j : U_j \to \mathbb{C}^n$ being an homeomorphism for each $j \in \{0, ..., n\}$.
We are now ready to make a more formal the connection between affine and projective plane complex algebraic curves discussed in Subsection 2.2.1 in Remark 3.

Remark 8. From the previous discussion it follows that we can identify $\mathbb{C}^2$ with the open subset

$$U = \{(z : w : u) \in \mathbb{P}^2 \mid u \neq 0\} \subset \mathbb{P}^2$$

via the following homeomorphism:

$$\phi : U \to \mathbb{C}^2, (z : w : u) \mapsto \left(\frac{z}{u}, \frac{w}{u}\right)$$

with the following inverse:

$$\phi^{-1} : \mathbb{C}^2 \to U, (z, w) \mapsto (z : w : 1).$$

The complement of U in $\mathbb{P}^2$ is the projective line defined by $\{(z : w : 0) \in \mathbb{P}^2\}$, which we can identify with $\mathbb{P}^1$ via the following map:

$$h : \mathbb{P}^2 \to \mathbb{P}^1, (z : w : 0) \mapsto (z : w).$$

Therefore the projective complex plane $\mathbb{P}^2$ is the disjoint union of a copy of $\mathbb{C}^2$ and a copy of $\mathbb{P}^1$. We now consider the projective curve $\tilde{\mathcal{C}}$ defined by the nonconstant homogeneous polynomial $p(z, w, u)$ of degree m. If we identify U with $\mathbb{C}^2$ as described before, then the intersection $U \cap \tilde{\mathcal{C}}$ is the affine curve $\mathcal{C}$ in $\mathbb{C}^2$ defined by the inhomogeneous polynomial in two variables $p(z, w, 1)$. Conversely, we consider the affine curve $\mathcal{C}$ in $\mathbb{C}^2$ defined by the inhomogeneous polynomial $q(z, w)$ of degree m. If we identify U with $\mathbb{C}^2$ as described before, then $\mathcal{C}$ is the intersection $U \cap \tilde{\mathcal{C}}$, where $\tilde{\mathcal{C}}$ represents the projective curve in $\mathbb{P}^2$ defined by the homogeneous polynomial $u^m q(\frac{z}{u}, \frac{w}{u})$.

The main reason for introducing topological spaces is to provide a general setting for introducing the notions of convergence and continuity. However, before we give the formal definitions for these notions, we present the definition for the notion of neighbourhood in a topological space:

Definition 17. If X is a topological space and $q \in X$, then we call a neighbourhood of q, denoted $\mathcal{V}_q$, an open set containing q. A neighbourhood of a subset $K \subset X$ is an open set containing K.

We are now prepared to define the notion of convergence in a topological space:

Definition 18. If X is a topological space and $(q_k)_k$ is any sequence of points in X, then we say that the sequence converges to $q \in X$, and q is the limit of the sequence, if for every neighbourhood U of q there exists N such that $q_k \in U$ for all $k \geq N$. We denote this assertion by $q_k \to q$ or by $\lim\limits_{k\to\infty} q_k = q$.

We introduce the notion of continuous functions between topological spaces:

Definition 19. If X and Y are topological spaces, a function $f : X \to Y$ is called continuous if for every open set $U \subset Y$, $f^{-1}(U)$ is open in X.

From Remark 6, it follows that all the continuous functions between metric spaces are also continuous functions between topological spaces. We next define an homeomorphism in the following way:

Definition 20. Let X, Y be topological spaces. We say that X and Y are topologically equivalent (or homeomorphic) if there exists a bijective function $f : X \to Y$ such that both f and its inverse f^{-1} are continuous. In this case, the function f is called a homeomorphism.

We can give the following equivalent definition for an homeomorphism:

Definition 21. We consider X, Y to be two topological spaces. We say that X and Y are homeomorphic and we denote this by $f \cong g$ if there exists a continuous function $f : X \to Y$ and a continuous function $g : Y \to X$ such that $f \circ g = id_Y$ and $g \circ f = id_X$, where we denote by the $\circ$ symbol the composition of functions, by the id_Y symbol the identity function on Y, and by the id_X symbol the identity function on X. This means that the identity functions are defined as follows: $id_Y : Y \to Y, y \mapsto y$ and $id_X : X \to X, x \mapsto x$.

From Definition 21 we notice the following two important facts: (1) the functions f and g are inverse to each other, i.e. $f = g^{-1}$ and $g = f^{-1}$; (2) a function is continuous if the inverse image of an open set is open, which actually represents the lemma criterion for continuity.
We now define the notion of homotopy, which is also important for our study:

Definition 22. Let X, Y be topological spaces, and $f, g : X \to Y$ be continuous functions. We say that f is homotopic to g and we denote this by $f \approx g$ if there exists a continuous function $G : X \times [0, 1] \to Y$ such that for all $x \in X$ the following property holds: $G(x, 0) = f(x), G(x, 1) = g(x)$. In addition, G is called a homotopy.

We can introduce an equivalent definition for the notion of homotopy in the following way:

Definition 23. We consider X, Y to be two topological spaces. We say that X and Y are homotopic if there exists a continuous function $f : X \to Y$ and a continuous function

$g : Y \to X$ such that $f \circ g \approx id_Y$ and $g \circ f \approx id_X$, where we denote by the $\circ$ symbol the composition of functions, by the id_Y symbol the identity function on Y, and by the id_X symbol the identity function on X. This means that the identity functions are defined as follows: $id_Y : Y \to Y, y \mapsto y$ and $id_X : X \to X, x \mapsto x$.

The reason for which we introduce the notion of homotopy is that we will use it in the next chapters of this thesis and it is closely connected to the notion of homeomorphism by the following property: if f is a homeomorphism between two topological spaces, then f is also a homotopy. The converse statement is not necessarily true. For instance if we consider the two topological spaces $X = \{P\}$ and $Y = \mathbb{R}^n$, then we observe that X and Y are homotopic topological spaces since for $f : X \to Y, P \mapsto (\underbrace{(0, 0, ..., 0)}_{\text{n times}})$ and $g : Y \to X, (y_1, ...y_n) \mapsto P$, we obtain that $g \circ f \approx id_X$ and $f \circ g \approx id_Y$. However we observe that the two topological spaces X and Y are not homeomorphic because there does not exist any bijection between them since X and Y have different cardinality, i.e. $|X| = 1, |Y| = \aleph_c = 2^{\aleph_0}$.

We now return to the notion of homeomorphism and make several important observations. We notice that the notion of homeomorphism generates an equivalence relation. Consequently, the resulting equivalence classes are called homeomorphism classes. The notion of homeomorphism is very important in topology, as it identifies the "topological properties" in a topological space as being those properties that are preserved by homeomorphisms. More informally, the "topological properties" are those properties from a topological space that remain unchanged under continuous deformations, i.e. deformations where one is not allowed to cut objects or to glue them together.

We continue with presenting some examples of homeomorphisms.

Example 14. The following are explicit examples of homeomorphisms:

1. Any open ball in $\mathbb{R}^n$ is homeomorphic to any other open ball. In the same way, all spheres in $\mathbb{R}^{n+1}$ are homeomorphic to one another. This example shows that "size" is not a topological property.

2. We consider the 2-dimensional sphere in $\mathbb{R}^3$ centered in the origin and of radius 1 defined as $S_1^2(0, 0, 0) = \{(x, y, z) \in \mathbb{R}^3 \mid x^2 + y^2 + z^2 = 1\}$. We take the cube centered in the origin defined as $C = \{(x, y, z) \in \mathbb{R}^3 \mid max(|x|, |y|, |z|) = 1\}$. In addition, we consider $\phi : C \to S^2, (x, y, z) \mapsto \phi(x, y, z) = \dfrac{(x, y, z)}{\sqrt{x^2 + y^2 + z^2}}$ to be the function that projects each point of the cube C onto the sphere S^2. We observe that the function ϕ is bijective and continuous with its inverse $\phi^{-1} : S^2 \to C, \phi^{-1}(x, y, z) = \dfrac{(x, y, z)}{max(|x|, |y|, |z|)}$ being a continuous function as well. This example illustrates that "corners" are not a topological property. In Figure 2.29 we roughly depict the continuous deformation of a sphere into a cube in the 3-dimensional Euclidean space, which basically represents the homeomorphism introduced in this example.

3. We consider the following homeomorphism: $\phi : (0, 1) \to \mathbb{R}, x \mapsto \dfrac{2x - 1}{x(1 - x)}$, which shows that the open interval $(0, 1)$ is homeomorphic to the real line $\mathbb{R}$. We recall that a subset S of a topological space $(\mathbb{R}^n, || \cdot ||)$ is bounded if it is contained in a ball of finite radius, i.e. if there is $x \in \mathbb{R}^n$ and $r > 0$ such that for all $s \in S$ we have that $||x - s|| < r$. We observe that the open interval $(0, 1)$ is bounded, whereas the real line $\mathbb{R}$ is not bounded and therefore "boundedness" is not a topological property.

From Example 14 we notice that "size", "boundnedness" and "corners" are not topological properties. In the next example we give some examples of topological properties.

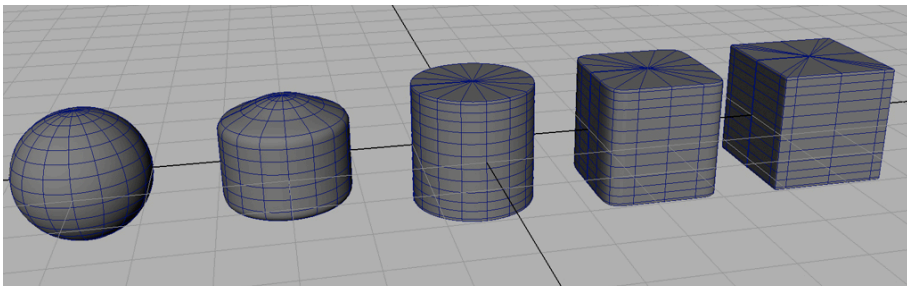


Figure 2.29: Deformation of a sphere into a cube in the 3-dimensional Euclidean space, which represents the homeomorphism defined in Example 14. Pictures produced with Blender, see [Foundation, 2004] for more information.

Example 15. The following represents a list of topological properties:

- the dimension;
- the existence of boundary, e.g. an open ball versus a closed ball;
- the orientability, e.g. the Möbius strip versus the sphere in the 3-dimensional space. We recall that a surface is called orientable if a loop going around one way of the surface can never be continuously deformed without overlapping itself to a loop going the opposite way of the surface. In a rough sense, we can decide that a surface is non-orientable in the following way: (i) we consider an oriented circle (called an indicatrix) placed on a surface; (ii) we then move this indicatrix around an arbitrarily closed path on the surface; (iii) if there exists a curve that brings the indicatrix back with its orientation reversed then the surface is non-orientable. In Figure 2.30 we include examples of non-orientable surfaces such as the Möbius strip (or band) and the Klein bottle. Another example of a non-orientable surface is the complex projective plane $\mathbb{P}^2(\mathbb{C})$. As examples of orientable surfaces we mention the sphere in the 3-dimensional space, see Figure 2.26, and the torus in the 3-dimensional space, see Figure 2.27.

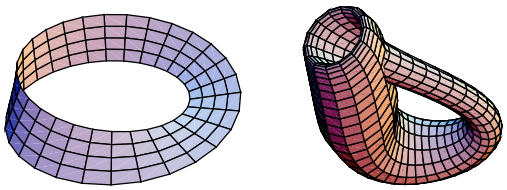


Figure 2.30: Example of non-orientable surfaces. From left to right: (1) a Möbius strip; (2) a Klein bottle. Pictures produced with Mathematica, see [Wolfram, 2000] for more information.

- the connectedness, and the pathconnectedness. We say that a topological space is connected if it cannot be written as a disjoint union of two nonempty open subsets, i.e. there does not exist any decomposition $X = U_1 \sqcup U_2$, where U_1, U_2 are open subsets of X and $\sqcup$ denotes the symbol for the disjoint union. For instance the interval $[0, 1]$ is connected. Moreover, we define a path in X as a continuous map $\varphi : [0, 1] \to X$. Then X is path connected if for any two points $x_1, x_2 \in X$ there exists a path φ with $\varphi(0) = x_1$ and $\varphi(1) = x_2$, i.e. X is path connected if any two points from X can

be joined by a path. A path connected set is also connected, but the converse is not necessarily true. Still, in the Euclidean space the notion of connectedness and path connectedness are equivalent.

- the connectivity, e.g. simply connected versus multiply connected. We say that a path-connected set is simply connected (or 1-connected) if any simple closed curve can be continuously shrunk to a point in the set. If the domain is not simply connected, it is said to be multiply connected. For example, a 2-dimensional sphere in $\mathbb{R}^3$ is simply connected, while a torus in $\mathbb{R}^3$, the Möbius strip, the Klein bottle and the projective complex (or real) plane are not simply connected.

- the compactedness, e.g. a 2-dimensional sphere in $\mathbb{R}^3$ is compact, whereas the affine real Euclidean plane $\mathbb{R}^2$ is not compact.

We next define the notion of "being locally Euclidean":

Definition 24. A subset M of some Euclidean space $\mathbb{R}^k$ is locally Euclidean of dimension n if every point of M has a neighborhood in M that is topologically equivalent to a ball in $\mathbb{R}^n$.

At this point, we establish the definition of a topological manifold (or simply manifold) in the following way:

Definition 25. An n-dimensional manifold (or n-manifold) is a subset of some Euclidean space $\mathbb{R}^k$ that is locally Euclidean of dimension n.

We now introduce the notion of a differential manifold following [Jänich, 2000]. For more information on differentiable manifolds, the reader can consult more advanced books on differentiable geometry such as for instance [DoCarmo, 1976].
We introduce the notion of a coordinate chart and of an n-dimensional atlas as follows:

Definition 26. We define a coordinate chart and an atlas in the following way:

- If M is a n-manifold, then a homeomorphism from an open subset $U \subset M$ to an open subset of $\mathbb{R}^n$ is called a coordinate chart (or a chart) on U.

- A collection of charts that cover the manifold M, i.e. whose union is the whole manifold, is called an atlas. This means that an n-dimensional atlas on M is a collection of coordinate charts $\{U_{\alpha,\varphi_\alpha}\}_{\alpha\in I}$ such that (i) M is covered by the $\{U_\alpha\}_{\alpha\in I}$; (ii) for each $\alpha, \beta, \varphi(U_{\alpha\cap U_\beta})$ is open in $\mathbb{R}^n$; (iii) the map $\varphi_\beta\varphi_\alpha^{-1} : \varphi(U_\alpha \cap U_{beta}) \to \varphi_\beta(U_\alpha \cap U_\beta)$ is C^∞ (or infinitely differentiable or smooth) with its inverse being also C^∞. We recall that $f(x_1, ..., x_n) \in \mathbb{R}^n$ is C^∞ if it has derivatives of all orders.

We introduce the following definition of compatibility for two atlases:

Definition 27. Two atlases $\{(U_\alpha, \varphi_\alpha)\}, \{(V_\beta, \psi_\beta)\}$ are compatible if their union is an atlas, i.e. all the maps $\psi_\beta\varphi_\alpha^{-1}$ are smooth.

We notice that the compatibility notion from the previous definition is an equivalence relation and therefore we can introduce the notion of a differentiable structure on a manifold M as follows:

Definition 28. A differentiable structure $\mathcal{D}$ on the manifold M is an equivalence class of atlases.

We remember that a topological space M is said to be Hausdorff space if given any pair of distinct points $q_1, q_2 \in M$, there exist neighbourhoods U_1 of q_1 and U_2 of q_2 with $U_1 \cap U_2 = \emptyset$. In addition, we say that a topological space M satisfies the second axiom of countability if it has a countable topological basis. We define a topological basis as introduced in [Mumkres, 2000]: if M is a set, then a basis for a topology on M is a collection B of subsets of M (called basis elements) satisfying the following properties: (i) for each $x \in M$, there is at least one basis element B containing x; (ii) if x belongs to the intersection of two basis elements B_1 and B_2, then there is a basis element B_3 containing x such that $B_3 \subset B_1 \cap B_2$. We can now define a differentiable manifold as follows:

Definition 29. An n-dimensional differentiable manifold is a pair $(M, \mathcal{D})$ consisting of a Hausdorff space M that satisfies the second axiom of countability and a differentiable structure $\mathcal{D}$ on M.

Remark 9. We recall the definition of the projective complex plane as introduced in Subsection 2.1.2. If we denote the coordinates of $\mathbb{P}^2(\mathbb{C})$ by $[z : w : u]$, then $\mathbb{P}^2(\mathbb{C})$ is covered by the open sets U_z, U_w, U_u, where

$$U_z = \{(1 : \frac{w}{z} : \frac{u}{z}) \mid z \neq 0\},$$

$$U_w = \{(\frac{z}{w} : 1 : \frac{u}{w}) \mid w \neq 0\},$$

$$U_u = \{(\frac{z}{u} : \frac{w}{u} : 1) \mid u \neq 0\}.$$

Each of the open sets U_z, U_w, U_u is homeomorphic to $\mathbb{C}^2$. Thus U_z, U_w, U_u are all coordinate charts. In fact the collection $\{U_z, U_w, U_z\}$ forms an atlas, which covers $\mathbb{P}^2(\mathbb{C}) = U_z \cup U_w \cup U_u$.

2.3.2 Topological Properties of Plane Complex Algebraic Curves

In this subsection, we briefly establish the global topological properties of projective plane complex algebraic curves. We recall from Subsection 2.1.2, Definition 4 that a projective plane complex algebraic curve $\tilde{\mathcal{C}}$ is defined as the set of zeroes of the squarefree homogeneous polynomial $p(z, w, u)$, and that is $\tilde{\mathcal{C}} = \{(x : y : z) \in \mathbb{P}^2(\mathbb{C}) \mid p(z, w, u) = 0\}$, where $\mathbb{P}^2(\mathbb{C})$ represents the projective complex plane.

The first result in establishing the global topology of projective plane complex algebraic curves refers to the compactness property of the curves. In Subsection 3 we discussed the compactification process of affine plane complex algebraic curves, process that generates projective plane complex algebraic curves. The following theorem states that any curve in $\mathbb{P}^2(\mathbb{C})$ is compact, i.e.:

Theorem 3. *([Kirwan, 1992, p.42]) Any projective plane complex algebraic curve $\tilde{\mathcal{C}}$ in the projective complex plane $\mathbb{P}^2(\mathbb{C})$ is compact.*

The second important result with important consequence in determining the global topology of projective plane complex algebraic curves concerns the connectedness property of the curves, formulated in the following way:

Theorem 4. *([Kendig, 1977, p. 87]) Any projective plane complex algebraic curve $\tilde{\mathcal{C}}$ in the projective complex plane $\mathbb{P}^2(\mathbb{C})$ is connected.*

We mention that Theorem 4 basically says that it is impossible for a projective plane complex algebraic curve in $\mathbb{P}^2(\mathbb{C})$ to consist of different parts, hence all the parts of a projective plane complex algebraic curve must touch each other.

The third essential result, which is needed for specifying the global topology of projective plane complex algebraic curves refers to the orientability property of the curves. We recall that intuitively a manifold is orientable if we can specify in a consistent way a direction of spinning in the manifold at each of its points. The orientability property of nonsingular projective plane complex algebraic curves is stated by the following theorem:

Theorem 5. *([Kendig, 1977, p. 96]) Any nonsingular projective plane complex algebraic curve $\tilde{\mathcal{C}}$ in the projective complex plane $\mathbb{P}^2(\mathbb{C})$ is orientable.*

The next important results for specifying the global topology of projective plane complex algebraic curves are related to the theory of Riemann surfaces. We mention that a Riemann surface is defined as a 1-dimensional complex manifold. We add that a Riemann surface is a special kind of surface on which it makes sense to introduce holomorphic functions and to apply complex analysis. In addition, we recall that any Riemann surface is orientable. We note that the detailed study of Riemann surfaces is not one of the main goal of this thesis. Therefore for more information on this subject the reader can consult [Kirwan, 1992, p. 111]. However, for our purpose we do revise some essential results concerning Riemann surfaces. We first mention that a compact surface is a surface that is also a compact set. From [Kirwan, 1992] we know that any nonsingular projective plane complex algebraic curve $\tilde{\mathcal{C}}$ in the projective complex plane $\mathbb{P}^2(\mathbb{C})$ is a compact Riemann surface. An essential result for our study is thus the classification theorem of compact Riemann surfaces (i.e. of compact 1-dimensional complex manifolds), classification theorem that is formulated in the following way:

Theorem 6. *([Fischer, 2001, chap. 9]) Any compact Riemann surface is either a sphere or a torus with g-holes (i.e. a sphere with g-handles). The number g of holes is called the genus of the compact Riemann surface.*

From the previous observations and from Theorem 6 it follows that every nonsingular projective plane complex algebraic curve $\tilde{\mathcal{C}}$ in the projective complex plane $\mathbb{P}^2$ is topologically either a sphere or a g-holed torus (i.e. a sphere with g-handles). The number g of holes is called the genus of the nonsingular curve $\tilde{\mathcal{C}}$, see Figure 2.31. For an irreducible singular projective plane complex algebraic curve we can associate a compact Riemann surface called the resolution of singularities of the curve, and the genus of the singular curve is defined as the genus of this compact Riemann surface, i.e. the genus of the resolution of singularities. We make some necessary remarks concerning the resolution of singularities problem, which is one of the classical problems in algebraic geometry and which is formulated in the following way: given V an algebraic variety, decide whether there exists a nonsingular variety W with $\varphi : W \to V$ a birational map (i.e. a rational map with a rational inverse). For algebraic varieties defined over a field of characteristic 0 in any dimension (and thus for plane complex algebraic curves defined over the field of complex numbers), the resolution of singularities problem is solved. We remark that the resolution of singularities problem is still an open problem for algebraic varieties defined over a field of positive characteristic in dimension at least 4. For further details concerning the resolution of singularities problem, the interested reader can consult for instance [Hauser, 2000].

In this subsection, we thus introduced the genus of a projective plane complex algebraic curve as a global topological invariant of the Riemann surface associated to the projective plane complex algebraic curve. In Subsection 2.4.5 we derive a formula for the genus of a projective plane complex algebraic curve depending on the degree of the curve and on the set of singularities of the projective plane complex algebraic curve. We add that the genus of a projective plane complex algebraic curve has an important role in the classification of irreducible projective plane complex algebraic curves.

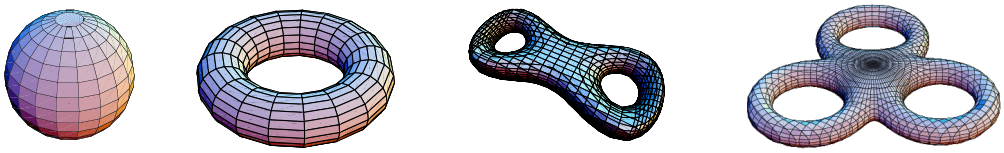


Figure 2.31: Visualization for the genus of several nonsingular projective plane complex algebraic curves. Every nonsingular projective plane complex algebraic curve is topologically a torus with g-holes. The number g is called the genus of the nonsingular curve. From left to right: (1) a sphere; (2) a torus with 1-hole; (3) a torus with 2-holes; (4) a torus with 3-holes. Pictures produced with Mathematica, see [Wolfram, 2000] for more information.

2.3.3 Singularities and Knot Theory

In this subsection we discuss the local topological properties of plane complex algebraic curves. In particular, we introduce the notion of a link of a singularity, which is one of the key ingredient of this thesis. For defining the link of a singularity and explaining its relation to the local topology of plane complex algebraic curves, we need to introduce some fundamental notions from knot theory. Therefore we divide this subsection into two main parts: one part in which we first introduce the basic knot theory, and a second part in which we then discuss the concept of a link of a singularity by connecting the singularity theory with the knot theory.

Basic Knot theory

In the following paragraphs, we include a short overview concerning basic knot theory, which turns out important for our study. We follow the basic books on knot theory of [Adams, 2004], [Livingston, 1993], [Rolfsen, 1976] and [Kauffman, 1991].
We define a knot in the following way:

Definition 30. A knot $K \subset \mathbb{R}^3$ or $K \subset S^3$ is a subset of points homeomorphic to the circle denoted with S^1. We add that the circle can be arranged as a smooth curve or as a polygonal curve.

The simplest knot of all is just the unknotted circle, which we call the unknot or the trivial knot, for an example see Figure 2.32.

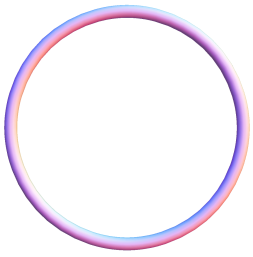


Figure 2.32: Example of the unknot, also called the trivial knot. Picture produced with Mathematica, see [Wolfram, 2000] for more information.

We recall that a homeomorphism is a continuous bijective function with a continuous inverse. We define the equivalence of two knots as follows:

Definition 31. We say that two knots are equivalent if there exists an orientation preserving homeomorphism on $\mathbb{R}^3$ that maps one knot onto the other. This equivalence is called (ambient) isotopy.

We can specify a particular knot using the following definition:

Definition 32. A knot is a continuous simple closed curve in $\mathbb{R}^3$.

The continuity in Definition 32 allows the existence of infinitely knotted loops called wild knots, as seen in Figure 2.33. To eliminate the wild points in the wild knots (i.e. the point where the small knots bunch up) we can introduce the notion of differentiability. This eliminates the wild points, since there is no continuous way to define a tangent direction at this point. This remedy is possible but difficult and therefore we will not describe it in this thesis.

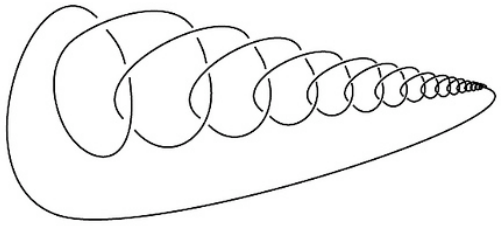


Figure 2.33: Example of a wild knot. Picture from [Livingston, 1993].

Alternatively, we can use polygonal curves to define knots. Since polygonal curves are finite by nature we can thus eliminate the wild points in the wild knots. We can think of knots as build up of straight lines and give the coordinates of the corners of these lines. More formally we define a simple closed polygonal closed as follows:

Definition 33. Let $p, q \in \mathbb{R}^3, p \neq q$. Let $[p, q]$ denote the segment line joining p, q. Let $(p_1, ..., p_n)$ be an ordered set with $p_i \neq q_j, i, j \in \{1, 2, ..., n\}$. Then $P = \bigcup_{i=1}^{n-1} [p_i, p_{i+1}] \cup [p_n, p_1]$ is a closed polygonal curve. In addition P is simple if each segment intersects exactly two other segments only at their endpoints.

Informally, we notice that a polygonal curve is simple if each segment intersects exactly two other segments only at their endpoints. In addition, a polygonal curve is closed if its first vertex coincides with its last vertex. We can now introduce the definition of a knot using a simple closed polygonal curve:

Definition 34. A knot K is a simple closed polygonal curve P in $\mathbb{R}^3$. The line segments of P are called the edges of the knots, and the corners of P are called the vertices of the knot. A knot is called tame if it has a polygonal representative.

We now introduce the notion of a link:

Definition 35. A link is a finite disjoint union of knots $L = K_0 \cup K_1 \cup ... \cup K_n$. Each knot K_j with $j \in \{0, ..., n\}$ is called a component of the link. The number of components of a link is called the multiplicity of the link. A subset of the components of L embedded in the same way is called a sublink.

Based on the former definition we make the following remarks: (1) a knot is a link with one component; (2) the unlink (trivial link) is the union of unknots all lying in the same plane; (3) links can be oriented (each component is assigned an orientation). An unoriented n-component link can be assigned orientations in 2^n ways.

We introduce the equivalence of links as follows:

Definition 36. We say that two links L_1, L_2 are equivalent if there exists an orientation-preserving homeomorphism on $\mathbb{R}^3$ that maps one link onto the other. This equivalence is called (ambient) isotopy.

We mention that the ambient isotopy introduced in the previous definition has to preserve any orientations or labeling on the links. Without this requirement, the definition is weak as it does not impose any restriction on the isotopy: there is a free choice of how to match the components of L_1 with those of L_2. In Figure 2.34 we include several examples of knots and links, i.e. the trefoil knot, the Hopf link and the Borromean rings.

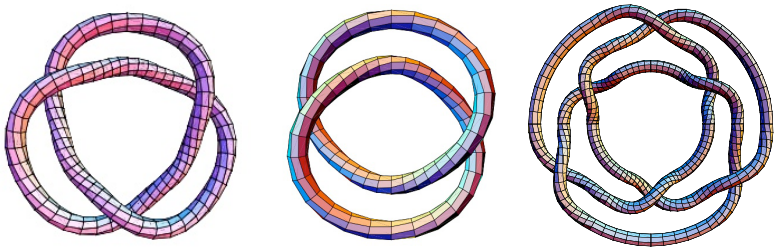


Figure 2.34: Examples of knots and links. From left to right: the trefoil knot, the Hopf link, the Borromean rings. Pictures produced with Mathematica, see [Wolfram, 2000] for more information.

In particular, we are interested in a special type of links called algebraic links. Without going too much into details concerning the algebraic links, at this point we mention that a link is called algebraic if it arises as the intersection of an algebraic curve with a sufficiently small sphere. We will thoroughly introduce the notion of an algebraic link in the second part of this subsection and in Subsection 2.4.2. From now on we will basically refer only to the class of links from knot theory since a knot is a link with one component.

In our study we use the following convention: we approximate links by simple closed polygonal curves in $\mathbb{R}^3$, but we usually draw them as smooth curves that do not intersect themselves in $\mathbb{R}^3$. In fact, in our study, we approximate a differentiable link by a piecewise linear link. Another important observation is that when we work with links, we actually work with their projection in the 2-dimensional Euclidean plane. In fact we work with their regular projection, which we define in the following way for the piecewise linear case:

Definition 37. A regular projection of a link L is a linear projection for which no three points on the link project to the same point, and no vertex projects to the same point as any other point on the link. A crossing point of the link L is an image of two knot points of such a regular projection from $\mathbb{R}^3$ to $\mathbb{R}^2$.

In Figure 2.35 we visualize the figure eight knot with its regular projection in $\mathbb{R}^2$.

As we can notice from Figure 2.35, regular projections of a link have the disadvantage that they contain double points (i.e. crossings) for which it is not clear which of the two paths goes under the other. To remove this problem we change the regular projection close to these double points, drawing the projection such that it has been cut. In this way, regular

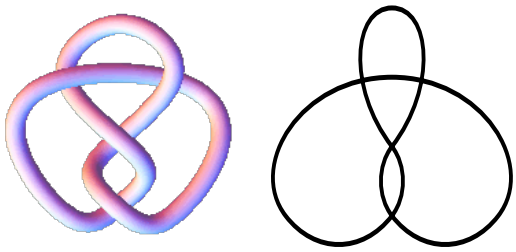


Figure 2.35: Example of a figure eight knot and its regular projection. From left to right: (1) the figure eight knot, which is a picture produced with Mathematica, see [Wolfram, 2000] for more information; (2) regular projection of the figure eight knot.

projections of a link to the plane allow the representation of a link as a special type of projection called a diagram, which we formally define as follows:

Definition 38. We consider the regular projection of a link as introduced in Definition 37. Then:

1. A link diagram (or simply diagram) of a link is the image under regular projection of the link, together with the information on each crossing point telling which branch goes over and which goes under, see Figure 2.36 and Figure 2.37. 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. Otherwise it is called unoriented. An arbitrary orientation of a knot is defined by choosing a direction to travel around the knot. This direction is denoted by placing coherently directed arrows along the projection of the knot in the chosen direction. In this case, we say that the knot is oriented.

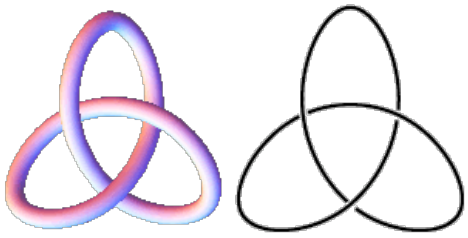


Figure 2.36: Example of a trefoil knot together with its diagram. From left to right: (1) the trefoil knot in $\mathbb{R}^3$; (2) the corresponding diagram of the trefoil knot in $\mathbb{R}^2$. Pictures produced with Mathematica, see [Wolfram, 2000] for more information.

We are interested in the following elements of a diagram:

Definition 39. For each diagram of a link we define the following elements:

1. A crossing is a double point of a regular projection together with the extra information telling which branch goes under and which goes over. In addition, for an oriented diagram we distinguish between lefthanded and righthanded crossings. A crossing of an oriented diagram 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

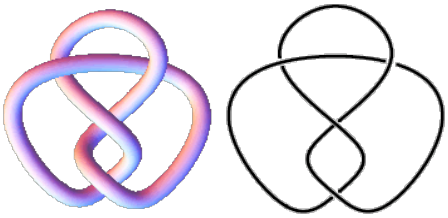


Figure 2.37: Example of a figure eight knot together with its diagram. From left to right: (1) the eight figure knot in $\mathbb{R}^3$; (2) the corresponding diagram of the figure eight knot in $\mathbb{R}^2$. Pictures produced with Mathematica, see [Wolfram, 2000] for more information.

crossing with -1 and a righthanded crossing with $+1$ as indicated by the dotted round arrow in Figure 2.38.

2. An arc is the part of a diagram between two undercrossings, see Figure 2.39. We notice that the number of arcs in a diagram equals the number of crossings in the same diagram, see Figure 2.39.

Figure 2.38: Types of crossings: lefthanded crossing (-1) and righthanded crossing (+1).

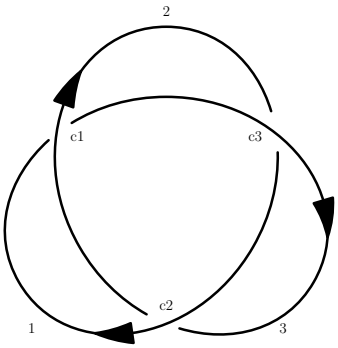


Figure 2.39: Oriented diagram of the trefoil with 3 arcs denoted with $\{1, 2, 3\}$ and 3 lefthanded crossings denoted with $\{c_1, c_2, c_3\}$.

In the following we include a list containing different types of knots:

- The first type of knots that we mention in this thesis are the torus knots. In the following we explain the method used to generate this type of knots. As introduced in Subsection 2.3.1 in Example 13 a torus is generated by taking a circle in the yz

plane of radius r centered on the y-axis at distance $R + r$ from the origin, and then by rotating it around the z-axis. If we parametrize the circle by angle $v \in [0, 2\pi]$ and the rotation by angle $u \in [0, 2\pi]$, then we can express the torus as:

$$\begin{pmatrix} \cos u & -\sin u & 0 \\ \sin u & \cos u & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ R + r\cos v \\ r\sin v \end{pmatrix} = \begin{pmatrix} -(R + r\cos v)\sin u \\ (R + r\cos v)\cos u \\ r\sin v \end{pmatrix}.$$

The parameters r, R control the geometry of the torus: r is the radius of the tube, R is the radius of the hole. The angles form a coordinate system: any point on the torus can be labelled by a pair (v, u).
The subset of points defined by the equation $pv = qu$ for coprime integers p, q winds its way around the torus and forms a knot, called a (p, q)-torus knot. A (p, q)-torus knot is equivalent with a (q, p)-torus knot. Since they lie in the surface of a standard torus, torus knots are some of the simplest knots to describe parametrically. Thus we can give the following definition for a torus knot:

Definition 40. The following definitions are equivalent:

1. A (p, q)-torus knot is obtained by looping a string through the hole of a torus p times with q revolutions before joining its ends, where p and q are relatively prime.
2. A (p, q)-torus knot is a curve on the torus, which is specified by winding p times around the main axis of the torus and q times around the tube of the torus.

In Figure 2.40 we include several examples of torus knots. We notice that the trefoil knot is a $(3, 2)$-torus knot.

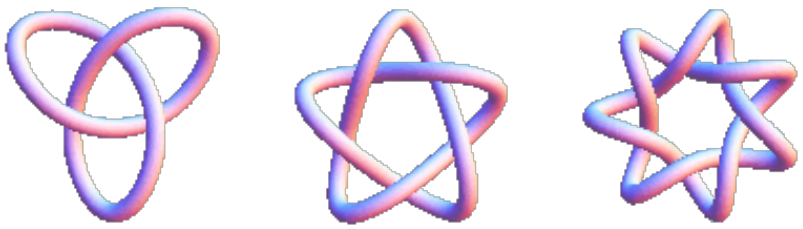


Figure 2.40: Examples of torus knots. From left to right: (1) a $(3, 2)$ torus knot; (2) a $(5, 2)$ torus knot; (3) a $(7, 2)$ torus knot. Pictures from Wolfram Research, see [E W. Weisstein, 1999].

- The second type of knots that we present in our study are the prime knots. Informally, two oriented knots (or links) can be summed by placing them side by side and joining them by straight bars so that orientation is preserved in the sum. The knot sum is also known as composition in [Adams, 2004] or connected sum in [Rolfsen, 1976]. We add that given any 2 knots K and J one can form their connected sum denoted by $K\#J$. In Figure 2.41 we visualize the connected sum $K_1\#K_2$ of two unknots denoted with K_1 and K_2.

  Under these assumptions, we say that a knot is prime if it cannot be decomposed as a connected sum of nontrivial knots. As examples we mention that all torus knots are prime knots. In addition the figure eight knot (also known as the Flemish knot and savoy knot) is the unique prime knot of four crossings, see Figure 2.42. An important

Figure 2.41: Connected sum of 2 unknots. From left to right: (1) two unknots in $\mathbb{R}^3$ denoted K_1, K_2 : (2) the connected sum $K_1 \# K_2$. Pictures from Wolfram Research, see [E W. Weisstein, 1999].

result was proven by Schubert in 1974. Basically he showed that any knot can be decomposed uniquely as the connected sum of prime knots. This is an analogy to positive integers.

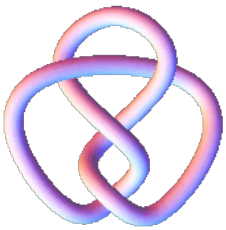


Figure 2.42: Figure eight knot, which is the unique prime knot of four crossings. Picture produced with Mathematica, see [Wolfram, 2000] for more information.

- The third type of knots that we remember are the amphicheiral knots (or achiral knots). We say that a knot is amphicheiral (or achiral) if it is equivalent to its mirror image. As examples, we add that the trefoil not is not amphicheiral, whereas the figure eight knot is amphicheiral, see Figure 2.43 and Figure 2.44. In addition all the torus knots are not amphicheiral.

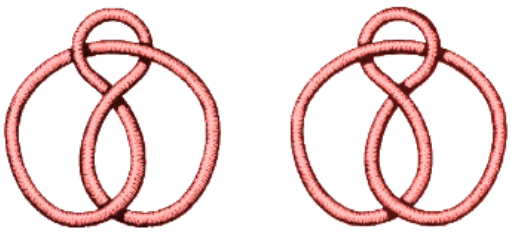


Figure 2.43: Figure eight knot and its mirror image, from [University of Wales, 2004].

- The last type of knots that we present in this overview are the alternating knots. We define an alternating knot as a knot that possesses a diagram in which crossings alternate between under and over crossings. Not all the knot diagrams of an alternating knot need to be alternating diagrams. We add that the trefoil, the figure eight knot and all the prime knots with 7 or fewer crossings are alternating knots. In Figure 2.45 we visualize an alternating projection of the figure eight knot, which thus shows that

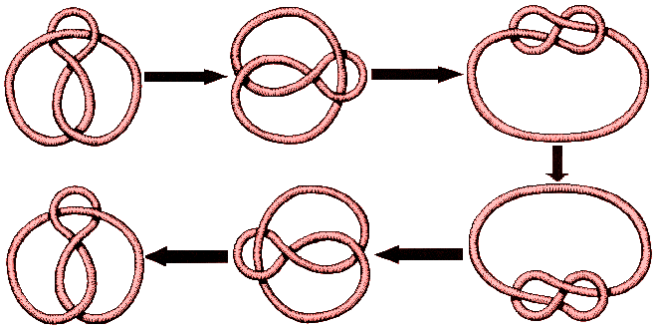


Figure 2.44: Figure eight knot deformed to its mirror image by a sequence of moves, which shows that the figure eight knot is an amphicheiral knot. Pictures from [University of Wales, 2004].

the figure eight knot is an alternating knot. We finish here our list containing different types of knots.

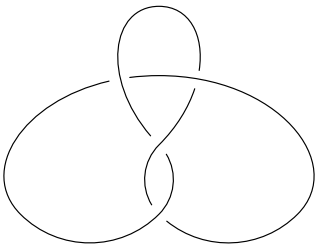


Figure 2.45: Alternating projection of the figure eight knot, which shows that the figure eight knot is an alternating knot.

The main problem in knot theory is to distinguish between different links and to establish whether two links are equivalent or not. A simplified version of this problem is formulated as follows: given a projection of a knot, decide whether it is the unknot or not. To show that 2 diagrams represent the same knot diagram we can use Reidemeister theorem, that basically says that 2 diagrams representing the same knot are always related by a sequence of three special moves called the Reidemeister moves. We define the Reidemeister moves in the following way:

Definition 41. A Reidemeister move is one of the 3 ways to change a diagram of a knot that will modify the relation between its crossings see Figure 2.46, i.e.:

1. the Reidemeister move of type I allows to put in or to take out a twist in a knot;
2. the Reidemeister move of type II allows to either add two crossings or to remove two crossings of the knot;
3. and the Reidemeister move of type III allows to slide a strand of the knot from one side of the crossing to the other side of the crossing.

Thus to show that two diagrams represent the same knot we can use Reidemeister theorem (1926), which is formulated as follows:

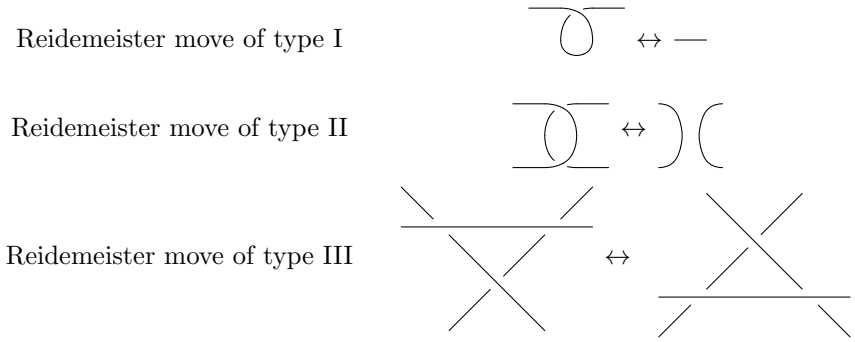


Figure 2.46: Reidemeister moves of type I, II, III.

Theorem 7. *Two links are equivalent if and only if any diagram of one can be transformed into a diagram of the other by a sequence of Reidemeister moves.*

However if two links are equivalent it can be difficult to decide the smallest number of moves that can be used to transform one diagram into the other. If after some time one cannot transform one diagram into another using the Reidemeister moves, maybe one is not smart enough. Or it might just happen that the knots are really different. To prove that two links are not equivalent we use the notion of link invariants:

Definition 42. 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 2.46).

Some link invariants are: the crossing number, the unknotting number, the tricolorability, the Alexander polynomial, the Jones polynomial, etc. We make some basic remarks concerning some of these link invariants:

- The crossing number and the unknotting number are the most natural invariants to study knots. They are simple to define, but their computation is difficult. We define the crossing number of a knot K, denoted with $c(K)$, as the least number of crossings that occur in any diagram of the knot. Moreover, we say that a knot K has unknotting number n $\big($denoted $u(K) = n\big)$ if there exists a diagram of K such that changing n crossings in the diagram turns the knot into the unknot and there is no other diagram such that fewer changes would have turned it into the unknot. As an example we mention that the trefoil knot has its crossing number equal to 3 and its unknotting number equal to 1, as seen in Figure 2.47.

- Another link invariant is the tricolorability. We say that a diagram of a knot is tricolorable (or simply colorable) if each arc can be drawn using one of the 3 colors such that at each crossing either 3 different colors came together or the same color comes together and at least 2 of the colors are used. From Figure 2.48 we observe that the trefoil knot is colorable. From Figure 2.49 we observe that the unknot is not colorable since only one color is used to draw the unknot, while the notion of colorability requires that at least 2 colors are being used. Since the trefoil is colorable

Figure 2.47: Trefoil knot has its crossing number equal to 3 as 3 is the least number of crossings that occur in any diagram of the trefoil; and its unknotting number equal to 1 since changing 1 crossing in the diagram change the trefoil into the unknot. Pictures from [University of Wales, 2004].

and the unknot is not colorable, we conclude that the trefoil and the unknot are different knots. Furthermore, any colorable knot is nontrivial. From Figure 2.50 we remark that the figure eight knot is also not colorable because there is a crossing for which 2 different colors meet. From the definition of colorability this is impossible as either 3 or 1 colors can meet at one crossing. Since the figure eight knot and the unknot are not colorable, it follows that the colorability cannot be used to show that figure eight knot is different from the unknot. Therefore we conclude that colorability is not a complete invariant for knots. Another important theorem, which we include here without a proof, says that if a diagram of a knot is colorable then all of its diagrams are colorable. A generalization of the notion of tricolorability is a more subtle kind of labelling, which gives new invariants, i.e. mod p labellings, where p is a prime positive integer. The problem of finding mod p labellings of a knot diagram can be reduced to solving a system of linear equations mod p. The dimension of the solution space for this system of equations is called the mod p rank of the knot. It is proved that if K has mod p rank n, then the number of the mod p labellings of the knot diagram is $p(p^n - 1)$, see [Livingston, 1993] for details.

Figure 2.48: Trefoil knot is tricolorable. Picture from [University of Wales, 2004].

- The most successful invariants to tell knots apart are the polynomial invariants. We mention here the following polynomial invariants:
  - The Alexander Polynomial introduced by J. W. Alexander in 1928. This polynomial distinguishes all knots of 8 crossings or fewer, but it does not distinguish a knot from its mirror image, i.e. it does not distinguish the amphicheiral knots.
  - The Jones polynomial introduced by Vaughan F. R. Jones in 1984. It distinguishes all knots of 10 crossings or fewer, it also distinguishes a knot from its mirror image, but it does not distinguish the mutant knots, which is another class of special knots, see [Adams, 2004] for information on the definition of mutant knots.

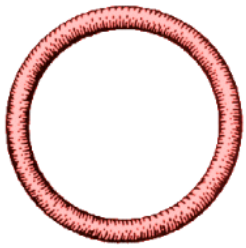


Figure 2.49: Unknot is not tricolorable as only one color is used for drawing the knot. Picture from [University of Wales, 2004].

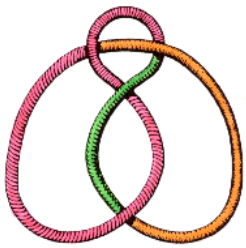


Figure 2.50: Figure eight knot is not tricolorable since there is a crossing for which only 2 different colors meet. From the definition of colorability this is impossible, since either 3 or 1 colors can meet at one crossing. Picture from [University of Wales, 2004].

– The HOMFLY polynomial was introduced in 1985/1987. This polynomial is a generalization of both the Alexander and the Jones polynomials. It is named after its inventors J. Hoste, A. Ocneanu, K. Millett, P. J. Freyd, W. B. R. Lickorish, and D. N. Yetter. Independently also J. H. Przytycki and P. Traczyk discovered the same polynomial. It does not distinguish mutant knots.

For further details and information on link invariants the reader can consult the detailed books of [Adams, 2004], [Livingston, 1993], [Burde and Zieschang, 1985]. As an important observation we mention that at present, there exists no complete invariant for links, i.e. an invariant that distinguishes all the links.

In this paper we will extensively use the Alexander polynomial, therefore at this point in our study we make several important remarks concerning the Alexander polynomial as a link invariant in knot theory. The Alexander polynomial of a link can be computed in several ways:

- by using Alexander's combinatorial method (1928), [Alexander, 1928]. This method uses the diagram of the knots, and the Reidemeister moves. This method will be presented in detail in Subsection 2.4.3;

- by using Fox's method (1963). This technique uses a representation of the fundamental group of the complement of the knot. It was also mentioned in Alexander's original paper [Alexander, 1928] in his "Miscellaneous" section, but Fox's description is more detailed. For a detailed and comprehensive overview of this method see [Crowell and Fox, 1963];

- or by using Conway's skein relation (1969). This method uses skein relation, some special equations that connect the crossings of different knot diagrams. It was also

mentioned in Alexander's original paper, but Conway's presentation is clearer and thus it paved the discovery for the Jones polynomial 15 years later.

In the following, we shortly present Fox's method for computing the Alexander polynomial of a link by following the book of [Crowell and Fox, 1963]. Fox's method for computing the Alexander polynomial is based on the study of the properties of the fundamental groups of the complementary spaces of knots. The complementary space of a knot K denoted with $\mathbb{R}^3 \setminus K$ consists of all the points of $\mathbb{R}^3$ that do not belong to K. We recall that $K \subset \mathbb{R}^3$ is a knot if there exists a homeomorphism of the unit circle S^1 into $\mathbb{R}^3$ whose image is K. The main idea in Fox's method for computing the Alexander polynomial is to define the fundamental group of the knot. For this purpose, we have to introduce the notion of a fundamental group of a topological space. We introduce the following preliminaries definitions:

Definition 43. A path in the topological space X is a continuous map $p : [0,1] \to X$. The continuous function $f : [0,1] \to X$ with the property that $f(0) = f(1)$ is called a loop. In addition, $f(0) = f(1)$ is called the base point of the loop.

We introduce the homotopy relation between two loops in a topological space as follows:

Definition 44. Let $x \in X$ and denote with $P(X,x)$ the set of all loops in X with the base point x. Let $f, g \in P(X,x)$ be two loops in X. We say that the loops f, g are homotopic denoted with $f \approx g$ if there exists a continuous function $H : [0,1] \times [0,1] \to X$, i.e. the family of maps $H_t = H|[0,1] \times \{t\}, t \in [0,1]$ is continuous, such that:

1. $H_0(y) := H(y,0) = f(y)$ and $H_1(y) := H(y,1) = g(y)$ for all $y \in [0,1]$;
2. the base point $H_t(0) := H(0,t) = H(1,t) := H_t(1) = x$ is independent of t.

We notice that the homotopy relation from the previous definition is an equivalent relation. We denote with $\pi(X,x) = P(X,x)/_{\approx}$ the collection of all equivalence classes of loops in the topological space X with the base point x. We define the following operations on $\pi(X,x)$:

- the inverse:

$$\begin{array}{l} i : \pi(X,x) \to \pi(X,x), \\ s \in \pi(X,x) \mapsto i(s)(t) := s^{-1}(t) = s(1-t) \in \pi(X,x), \end{array} \tag{2.36}$$

- and the multiplication:

$$\begin{array}{l} m : \pi(X,x)^2 \to \pi(X,x), \\ (s_1, s_2) \in \pi(X,x)^2 \mapsto m(s_1,s_2) := s_1 \circ s_2 = s_3 \in \pi(X,x), \end{array} \tag{2.37}$$

$$\text{where } s_3(t) = \begin{cases} s_1(2 \cdot t), t \in [0, \frac{1}{2}] \\ s_2(2 \cdot t - 1), t \in [\frac{1}{2}, 1]. \end{cases}$$

We call the group $\big(\pi(X,x), m, i\big) := \big(\pi(X,x), \circ, {}^{-1}\big)$ the fundamental group of the topological space X relative to the base point x.

We remember that a topological space X is connected if any two of its points can be joined by a path lying in X. As a remark we add that $\mathbb{R}^3$ is a connected topological space. We include the following important theorem concerning the fundamental group of a connected topological space:

Theorem 8. *If X is a connected topological space with $x, y \in X$, then $\pi(X, x)$ is isomorphic to $\pi(X, y)$, which we denote by $\pi(X, x) \simeq \pi(X, y)$. In other words, in a connected topological space X, the fundamental group of X is independent on the base point.*

The fundamental group of a knot $K \subset \mathbb{R}^3$ for some choice of the basepoint $x \in K$ is denoted with $\pi(K, x)$. However, the fundamental group of the complement of the knot K denoted with $\pi(\mathbb{R}^3 \setminus K, x)$ is more interesting to study. Since $\mathbb{R}^3 \setminus K$ is a connected topological space, we use simply the notation $\pi(\mathbb{R}^3 \setminus K)$ for the fundamental group of the complement of the knot because $\pi(\mathbb{R}^3 \setminus K)$ is independent of the base point. The fundamental group of the complement of the knot K denoted with $\pi(\mathbb{R}^3 \setminus K) := G(K)$ is simply called the knot group of K, or simply the group of the knot K. In this thesis, we will use both of these terminology.

We now need to introduce the notion of a presentation of a group. For this purpose, we let $S = |n|$ be a set of cardinality n. Then any $a \in S$ is called a letter of S, any a^n with $a \in S, n \in \mathbb{Z}$ is called a syllable, and a finite ordered sequence of syllables such as $b^{-3}a^0a^1c^2c^2a^0c^1$ is called a word. The unique word 1 is called the empty word. We denote with $W(S)$ the set of all words formed on S. On $W(S)$ we define the following operations:

- the product of 2 words formed by writing one word after the other;
- the elementary expansions and contractions: If $w_1, w_2 \in W(S)$ then:
  $u = w_1a^0w^2 \Leftrightarrow u = w_1w_2$,
  $u = w_1a^pa^qw^2 \Leftrightarrow u = w_1a^{p+q}w_2$.
- If $u, v \in W(S)$, then $u \sim v$ if one can be obtained from the other by a finite sequence of elementary expansions and contractions.

We denote by $F(S) = W(S)/_\sim$ the set of equivalence classes of words. On $F(S)$ we define the following two operations:

- the multiplication operation inherited from $W(S)$ defined as $[u][v] = [uv]$;
- the inverse operation denoted by $[u]^{-1}$ and representing the word obtained from u by reversing the order of its syllables and changing the sign of each exponent.

Under these assumptions, $(F(S), \cdot, {}^{-1})$ is a group, called the free group on the set S. We give the intuitive significance of the free group: a free group on a set S denoted by $F(S)$ is a group in which each element can be uniquely described as a finite length product of the form $s_1^{a_1} \cdot s_2^{a_2} \cdot ... \cdot s_n^{a_n}$, where s_j are distinct elements of S and $a_j \in \mathbb{Z}^*$ for all $j \in \{1, ..., n\}$. The set S is called the set of generators. In addition, a useful theorem from group theory says that any group is the homomorphic image of some free group.

Informally for the presentation of a group one specifies a set S of generators of G such that every element of the group can be written as a product of some of these generators, and a set R of relations among those generators. In addition, we say that the group G has a presentation denoted by $G = \langle S \mid R \rangle$. Formally we say that a group G has the presentation $G = \langle S \mid R \rangle$ if it is isomorphic (denoted by $\simeq$) to the quotient of the free group $F(S)$ by the normal subgroup N of $F(S)$ generated by the relations R, i.e.

$$G = \langle S \mid R \rangle \simeq F(S)/N.$$

To construct the presentation of the group $G = \langle S \mid R \rangle$ the idea is to take the smallest quotient of $F(S)$ such that each element of R gets identified with the identity element. We mention that R may not be a subgroup (or a normal subgroup), so we cannot take a

quotient by R. Therefore we take N the normal closure of R in $F(S)$, i.e. the smallest normal subgroup in $F(S)$ that contains R.

We recall that the knot group $G(K)$ of a knot K is the fundamental group of the knot complement $G(K) := \pi(\mathbb{R}^3 \setminus K)$. A presentation of the knot group $G(K)$ of the knot K is the Wirtinger presentation, which can be constructed from the diagram of the knot $K \subset \mathbb{R}^3$. We add that the knot group of the unknot is isomorphic to $\mathbb{Z}$, but the knot group of any nontrivial knot is infinite and non-abelian, as we will soon see later in this subsection. A proof that the Wirtinger presentation describes $G(K)$ is found in [Rolfsen, 1976]. The proof mainly uses the van Kampen theorem. We do not insist on the proof of this theorem, but we do describe the Wirtinger presentation of $G(K)$ as described also in [Harris and Quenell, 1999]:

- an element of $G(K)$ is represented by a loop, which begins at some fixed base point $x_0 \notin K$, winds through the space around K and returns to x_0;
- the composition operation in $G(K)$ corresponds to the concatenation of loops;
- the identity element is represented by a path that never leaves x_0, or by a loop at x_0, which never gets tangled up with any part of K such that it can be shrunk back to x_0 without getting caught anywhere;
- the inverse of the group element represented by a loop σ is represented by the same path traced in the opposite direction.

As an example in Figure 2.51 we visualize three loops in the complement of the figure eight knot, i.e. three loops in the group of the figure eight knot. If the loop σ_1 represents the group element g, then σ_2 represents g^2, and σ_3 represents the identity.

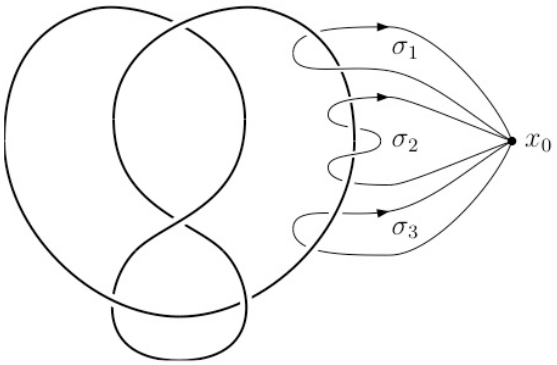


Figure 2.51: Three loops in the group of the figure eight knot represented by $\sigma_1, \sigma_2, \sigma_3$. Picture and example from [Harris and Quenell, 1999].

To obtain the generators S of the knot group $G(K)$ of a knot K we proceed in the following way:

- we consider x_0 somewhere off to the side of the oriented diagram of the knot K. We denote the arcs of the diagram with labels represented by non-negative integers. For each arc denoted with the label i in the diagram, we write down a group element g_i, represented by a loop that begins at x_0, crosses under arc i from right to left, then crosses over arc i and returns to x_0 without getting tangled up anywhere else in the knot;

- all the elements g_i generate the knot group $G(K)$ such that $g_i \in S$, i.e. any x_0-based loop through the space around K can be deformed into a sequence of loops each of which leaves x_0, circles one arc of K and returns to x_0.

In Figure 2.52 we can see the generators g_1, g_2, g_3 of the knot group $G(K)$ of the trefoil knot.

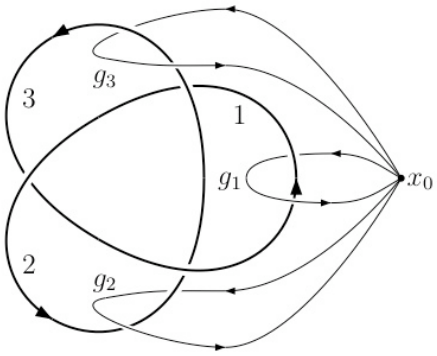


Figure 2.52: Generators of the knot group of the trefoil knot, i.e. $S = \{g_1, g_2, g_3\}$. Picture from [Harris and Quenell, 1999].

To get the relations S of the knot group $G(K)$ of the knot K we proceed as follows:

- for each crossing in the diagram, we have a relation among all the generators from S. For the type of crossing from Figure 2.53, we generate the following relation: the loop that passes under all the 3 arcs and circles the crossing once represents the group element $g_i g_k g_i^{-1} g_j^{-1}$ since it can be deformed into 4 loops representing these generators. Since this loop can be pulled clear of K, it is the identity element, so we get $g_i g_k g_i^{-1} g_j^{-1} = 1$.

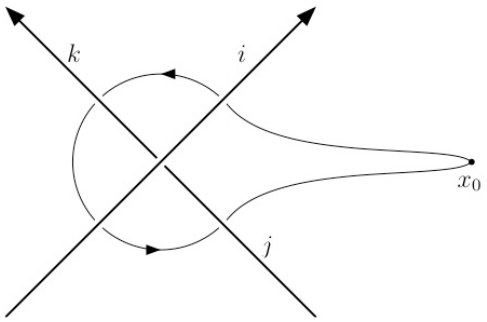


Figure 2.53: Crossing for the diagram of the trefoil knot, which for the knot group $G(K)$ of the trefoil knot with generators $S = \{g_i, g_j, g_k\}$ produces the relation $g_i g_k g_i^{-1} g_j^{-1} = 1$ in $G(K)$. Picture from [Harris and Quenell, 1999].

Thus for the diagram of the trefoil knot K from Figure 2.52, we obtain the following presentation of its knot group: $G(K) = \langle S \mid R \rangle$, where the set of generators S is given by $S =$

$\{g_1, g_2, g_3\}$, and the set of relations is represented by $R = \{g_2 g_1 g_2^{-1} g_3^{-1} = 1,\ g_3 g_2 g_3^{-1} g_1^{-1} = 1,\ g_1 g_3 g_1^{-1} g_2^{-1} = 1\}$.
The existence and the uniqueness of the Alexander polynomials (and of other knot polynomials) depend on the abelianized group of a knot group. In the following we introduce the abelianization of a group G. For any two elements $g, h \in G$, the commutator of g, h is given by $[g, h] = g^{-1}h^{-1}gh$. The commutator group denoted by $[G, G]$ is the subgroup of G generated by all the commutators, and $[G, G]$ is a normal subgroup of G. The quotient $G/[G, G]$ is an abelian group, called the abelianization of G. We introduce the notion of a free derivative as follows:

Definition 45. Let $F\big(\langle x_1, ..., x_n\rangle\big)$ be the free group of the generators $\langle x_1, ..., x_n\rangle$ and $\mathbb{Z}[F]$ the group ring of F. We define the free derivative for every generator x_i from $\langle x_1, ..., x_n\rangle$ as follows: $D_i : F\big(\langle x_1, ..., x_n\rangle\big) \to \mathbb{Z}[F], D_i := \dfrac{\partial}{\partial x_i}$ with the following properties:

$$\begin{gathered}
\frac{\partial}{\partial x_i} 1 = 0, \\
\frac{\partial x_j}{\partial x_i} = \delta_{ij} = \begin{cases} 1, \text{ if } i = j \\ 0, \text{ if } i \neq j \end{cases}, \\
\frac{\partial x_j^{-1}}{\partial x_i} = -\delta_{ij} x_j^{-1}, \\
\text{and for any word } w = u x_j \in F\big(\langle x_1, ..., x_n\rangle\big) : \frac{\partial}{\partial x_i} u x_j = \frac{\partial}{\partial x_i} u + u \frac{\partial x_j}{\partial x_i}.
\end{gathered} \tag{2.38}$$

Let $F(S)$ be the free group on $S = \langle x_1, ..., x_n\rangle$, and consider the following knot group presentation: $G(K) = \big(\langle x_1, ..., x_n\rangle \mid \langle r_1, ..., r_n\rangle\big)$. We consider the following function to be a group homomorphism:

$$\begin{gathered}
\phi : F(S) \to G, \\
s \mapsto s.
\end{gathered} \tag{2.39}$$

We recall that any group is a homomorphic image of some free group. Then $\phi : \mathbb{Z}[F] \to \mathbb{Z}[G]$ is a ring homomorphism.
Let $\psi : G \to G'$ be a group homomorphism with $G' = \psi(G)$ an abelian group, i.e. ψ is the abelianization of G. We include the following important theorem concerning the abelianized group of a knot group:

Theorem 9. *The abelianized group of every knot group is infinite cyclic, i.e. the generators of a presentation of a knot group are all mapped into the same generator.*

Based on the previous theorem, $\psi(G)$ is an infinite cyclic group generated by one element. We choose this element to be t and so it follows that the following function is a ring homomorphism:

$$\begin{gathered}
\psi : \mathbb{Z}[G] \to \mathbb{Z}[G'] = \mathbb{Z}[t, t^{-1}], \\
x_i \mapsto t.
\end{gathered} \tag{2.40}$$

We now define the Alexander matrix of the presentation of a knot group $G(K)$ as follows:

Definition 46. Let G be an infinite cyclic group. Then to any knot group presentation $G(K) = \big(\langle x_1, ..., x_n\rangle \mid \langle r_1, ..., r_m\rangle\big)$ we associate a Jacobian matrix of dimension $m \times n$ called the Alexander matrix of $G(K)$ whose ijth entry is given by $\psi\phi(\dfrac{\partial r_i}{\partial x_j})$ with $i \in \{1, ..., m\}, j \in \{1, ..., n\}$.

Before we introduce the main elements that lead to the computation of the Alexander polynomial of a link, we recall some fundamental concepts from algebra:

- A ring is called a greatest common divisor domain if it is an integral domain and every finite set of elements has a greatest common divisor. Moreover a unique factorization domain is an integral domain with identity in which every element that is neither zero nor a unit has an essentially unique factorization into primes. The relation between a unique factorization domain and a greatest common divisor domain is given by the following proposition: every unique factorization domain is a greatest common divisor domain.

- In a group ring, all group elements and their negatives are obviously units. They are called trivial units of the group ring.

- An ideal is called a principal ideal if it is generated by a single element. In addition, in a greatest common divisor domain with identity, the greatest common divisor of any finite set of elements is the generator of the smallest principal ideal that contains them.

For defining the Alexander polynomial of a link we now systematically introduce the following terminology and basic results from [Crowell and Fox, 1963]:

- Let R be an arbitrary commutative ring with a nonzero multiplicative identity 1. We consider a m (row) $\times$ n (column) matrix A with entries in R that we denote by $A \in \mathcal{M}_{m\times n}(R)$. For any non-negative integer $k \in \mathbb{Z}_+$ we define the kth elementary ideal $E_k(A)$ of A as follows:

  - If $0 < n - k \leq m$, then $E_k(A)$ is the ideal generated by the determinants of all $(n-k) \times (n-k)$ submatrices of A.
  - If $n - k > m$, then $E_k(A) = 0$.
  - If $n - k \geq 0$, then $E_k(A) = R$.

  Moreover, the elementary ideals of A form an ascending chain:

$$E_0(A) \subset E_1(A) \subset ... \subset E_n(A) = E_{n+1}(A) = ... = R.$$

- For any finite group representation of a knot group $G(K)$ and non-negative integer k, we define the kth elementary ideal of $G(K)$ to be the kth elementary ideal of the Alexander matrix of $G(K)$ as introduced in Definition 46. The elementary ideals, defined for any finite group presentation, represent a generalization of the knot polynomials. There are several differences between elementary ideals and knot polynomials. In this study, we underline some of these differences. The elementary ideals are defined for arbitrary finitely presented groups, whereas knot polynomials exist and are unique only for a more restricted class of groups satisfying certain algebraic conditions of the abelianized group of a knot group. In fact it can be shown that any tame knot group satisfies these algebraic conditions. Even when the knot polynomials exist, the elementary ideals contain more information. For instance, there exists knots that are not distinguishable by their polynomials, but which have different elementary ideals, for specific details see [Crowell and Fox, 1963, p. 128]. The invariance theorem basically says that the elementary ideals are invariants of any finitely presented group G, i.e. any two finite representations of G have the same chain of ideals.

- We recall that the abelianized group of any knot group is infinite cyclic, i.e. the generators of a presentation of a knot group are all mapped into the same generator. We establish the necessary algebraic properties of the group ring of an infinite cyclic group: the group ring of an infinite cyclic group is a greatest common divisor domain. For more information on the group ring of an infinite cyclic group see [Crowell and Fox, 1963, p.113].

- The group ring of an infinite cyclic group becomes upon selection of a generator t of the group, the ring of Laurent polynomials in t.

- For any integer $k \geq 0$, the kth knot polynomial Δ_k of a finite presentation $G(K) = \big(\langle x_1, ..., x_n \rangle | \langle r_1, ..., r_m \rangle\big)$ of a knot group is the greatest common divisor of the determinants of all $(n-k) \times (n-k)$ submatrices of the Alexander matrix of $G(K)$ as introduced in Definition 46. In addition:

$$\Delta_k = \begin{cases} 0, \text{ if } n-k > m \\ 1, \text{ if } n-k \leq 0. \end{cases}$$

- The group ring of the abelianized group of any finite presentation of a knot group is a greatest common divisor domain with only trivial units, i.e. the powers of a generator t and their negatives. It follows that the knot polynomials exist and are unique to within $\pm t^n$, where n is any integer and t is a generator of the infinite cyclic abelianized group of the finite presentation of the knot group.

- Each knot polynomial Δ_k is the generator of the smallest principal ideal containing the elementary ideal E_k.

- The 0th elementary ideal and polynomial of a knot group are trivial, i.e. $E_0 = \Delta_0 = 0$.

- The 1st elementary ideal of a knot group is a principal ideal generated by the 1st knot polynomial $\Delta_1(t)$. This is the most important member of the sequence of knot polynomials, it is denoted by $\Delta(t)$ and it is called the Alexander polynomial of the knot. In addition it is proven that the determinant of any one of the $(n-1) \times (n-1)$ submatrices of A may be taken to be the Alexander polynomial $\Delta(t)$.

- The 1st elementary ideal of a knot group is an invariant for the knot and does not depend on the presentation of the knot group.

In the following we illustrate the computation of the Alexander polynomial of the trefoil knot by considering the presentation of the knot group of the trefoil from Figure 2.52.

- We recall that a presentation of the knot group of the trefoil knot from Figure 2.52 is given by:

$$G = (S \mid R), \text{ where}$$

$$S = \langle g_1, g_2, g_3 \rangle,$$

$$R = \langle g_2 g_1 g_2^{-1} g_3^{-1} = 1,\ g_3 g_2 g_3^{-1} g_1^{-1} = 1,\ g_1 g_3 g_1^{-1} g_2^{-1} = 1 \rangle.$$

- In this case, based on Definition 46 the Alexander matrix of the trefoil knot K is the

Jacobian defined as:

$$J = \Big\{(a_{ij}) = \psi\phi\big(\frac{\partial r_i}{\partial g_j}\big) \mid i,j \in \{1,2,3\}\Big\},$$

$$\text{with } \phi : \mathbb{Z}[F] \to \mathbb{Z}[G],\ \phi(s) = s,$$

$$\text{and } \psi : \mathbb{Z}[G] \to \mathbb{Z}[t,t^{-1}],\ \psi(g_j) = t,\ \text{for } j \in \{1,2,3\}.$$

- By using the properties of the derivatives from the Equations 2.38, we first compute the terms $\dfrac{\partial r_i}{\partial g_j}$ for all $i,j \in \{1,2,3\}$. We compute these terms as follows:
  - we first compute the term $\partial r_1/\partial g_1$ in the following way:

$$\frac{\partial r_1}{\partial g_1} = \frac{\partial g_2 g_1 g_2^{-1} g_3^{-1}}{\partial g_1} = \frac{\partial g_2 g_1 g_2^{-1}}{\partial g_1} + g_2 g_1 g_2^{-1} \cdot \frac{\partial g_3^{-1}}{\partial g_1} =$$

$$= \frac{\partial g_2 g_1 g_2^{-1}}{\partial g_1} + g_2 g_1 g_2^{-1} \cdot (-\delta_{31} g_3^{-1}) = \frac{\partial g_2 g_1 g_2^{-1}}{\partial g_1} + g_2 g_1 g_2^{-1} \cdot 0 =$$

$$= \frac{\partial g_2 g_1}{\partial g_1} + g_2 g_1 \cdot \frac{\partial g_2^{-1}}{\partial g_1} = \frac{\partial g_2 g_1}{\partial g_1} + g_2 g_1 \cdot (-\delta_{21} g_2^{-1}) =$$

$$= \frac{\partial g_2 g_1}{\partial g_1} + g_2 g_1 \cdot 0 = \frac{\partial g_2}{\partial g_1} + g_2 \cdot \frac{\partial g_1}{\partial g_1} = 0 + g_2 \cdot 1 = g_2.$$

  - in the same way we obtain the next terms:

$$\frac{\partial r_1}{\partial g_2} = \frac{\partial g_2 g_1}{\partial g_2} + g_2 g_1 g_2^{-1} = 1 - g_2 g_1 g_2^{-1},$$

$$\frac{\partial r_1}{\partial g_3} = -g_2 g_1 g_2^{-1} g_3^{-1}.$$

  - we continue in the same manner and we compute all the terms $\partial r_i/\partial g_j$ for $i \in \{2,3\}$ and $j \in \{1,2,3\}$.

- By using the fact that $\phi(s) = s$, we thus obtain the Jacobian as follows:

$$J = \left(\begin{array}{c|ccc} & g_1 & g_2 & g_3 \\ \hline \dfrac{\partial r_1}{\partial g_j} & \psi(g_2) & \psi(1 - g_2 g_1 g_2^{-1}) & \psi(-g_2 g_1 g_2^{-1} g_3^{-1}) \\ \dfrac{\partial r_2}{\partial g_j} & \psi(-g_3 g_2 g_3^{-1} g_1^{-1}) & \psi(g_3) & \psi(1 - g_3 g_2 g_3^{-1}) \\ \dfrac{\partial r_3}{\partial g_j} & \psi(1 - g_1 g_3 g_1^{-1}) & \psi(-g_1 g_3 g_1^{-1} g_2^{-1}) & \psi(g_1) \end{array}\right)$$

- By using the information that $\psi(g_j) = t$ for all $j \in \{1,2,3\}$ we obtain the following expression for the Jacobian:

$$J = \begin{pmatrix} t & 1 - ttt^{-1} & -ttt^{-1}t^{-1} \\ -ttt^{-1}t^{-1} & t & 1 - ttt^{-1} \\ 1 - ttt^{-1} & -ttt^{-1}t^{-1} & t \end{pmatrix},$$

and thus we obtain:

$$J = \begin{pmatrix} t & 1-t & -1 \\ -1 & t & 1-t \\ 1-t & -1 & t \end{pmatrix}.$$

- The 2×2 minors of the Jacobian J generate the elementary ideal $\langle t^2 - t + 1 \rangle$, which is a principal ideal and which is generated by the Alexander polynomial of the trefoil knot $\Delta(t) = t^2 - t + 1$.

Another classical problem in knot theory is the classification of knots and links. At present there exist knot tables constructed by using Dowker notation, a notion that we will not discuss in our study. However the Dowker notation for knots and links does not allow the classification of all knots and links. For information on the Dowker notation, the reader can consult [Adams, 2004]. Another way in which knots can be classified is by using the notion of a braid, which we will not discuss in details in this thesis, but which can be studied also from [Adams, 2004]. We mention that the braids were originally studied by J. W. Alexander. Some basic results of Alexander concerning braids show that every knot can be converted into a braid, that braids can be classified and that braids can be manipulated by a sequence of arithmetical rules. Still, these results are not enough to allow a complete classification of knots and links.

Summary of knot theory. As a summary we recall the basic facts from knot theory introduced so far, which are in particular useful for our study:

- A knot is a piecewise linear or a differentiable simple closed curve in the 3-dimensional space $\mathbb{R}^3$.
- 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.
- A link is called algebraic if it arises as the intersection of an algebraic curve with a sufficiently small sphere.
- In our study we approximate a differentiable algebraic link by a piecewise linear algebraic link.
- Two links are equivalent if there exists an orientation preserving homeomorphism on $\mathbb{R}^3$ that maps one link onto the other. This equivalence is called (ambient) isotopy.
- When we work with links we actually work with a special type of their projection in $\mathbb{R}^2$ called a diagram, which contains information on each of its crossing by telling which branch goes under and which goes over. Therefore we distinguish between overcrossings and undercrossings.
- A diagram with an arbitrary orientation on it is called an oriented diagram. The elements of an oriented diagram are its crossings and its arcs.
- A crossing of an oriented diagram can either be righthanded or lefthanded. In particular, 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. An arc is the part of a diagram between two undercrossings. A crossing is always determined by three arcs. The number of crossings always equals the number of arcs in an (oriented) diagram.

- 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.
- Examples of link invariants are: the unknotting number, the crossing number, the colorability, the Alexander polynomial, the Jones polynomial, etc. However at present there is no complete invariant for links, i.e. there exists no invariant that distinguishes among all the links.
- An essential link invariant for our study is the Alexander polynomial, which can be computed in several ways. In this section, following [Crowell and Fox, 1963], we presented the computation of the Alexander polynomial by using Fox's method, which in fact uses the fundamental group of the complement of the knot in $\mathbb{R}^3$. In Subsection 2.4.3 we will fully describe the computation of the Alexander polynomial by using Alexander's method, a method that basically uses the diagram of the knot and the Reidemeister moves.

Local Topology of Plane Complex Algebraic Curves

The study of the local topology of isolated singularities of plane complex algebraic curves has been studied by many authors and it is closely related to knot theory. As mentioned in [Neumann, 2003], [Brauner, 1928], [Brieskorn and Knorrer, 1986] and in the detailed paper of [Eisenbud and Neumann, 1985], the local topology of a plane complex algebraic curve around its singular point can be described as follows: we consider a plane complex algebraic curve $\mathcal{C}$ with a singularity in the origin $(0,0)$ and we take a disk D of small enough radius centered in the origin $(0,0)$. We denote with S the 3-dimensional sphere centered in the origin, i.e. S represents the boundary of the small disk D. This small disk will intersect the curve $\mathcal{C}$ in a set that is homeomorphic to the cone over $\mathcal{C} \cap S$. In addition, the set $\mathcal{C} \cap S$ is a link in the 3-dimensional sphere S and it determines completely the local topology of $\mathcal{C}$ at its singularity $(0,0)$. In the literature, the links that arise as intersections of a plane complex algebraic curve with a small sphere are called links of plane curve singularities. It follows that to understand the topology of a plane complex algebraic curve around its singularity Q it suffices to understand the link of the singularity Q. To understand the local topology of a plane complex algebraic curve near its singular point, we also need to study the fundamental group of the complement of the link of the singularity in the 3-dimensional sphere. Consequently, we are interested in studying invariants, which determine completely the topology of plane curve singularities. We describe this issue in details in Subsection 2.4.3.

To make the description for the link of a plane curve singularity more formal, we next consider the 3-dimensional sphere of radius ϵ centered in the origin $(0,0)$ denoted

$$S_\epsilon = \{(z,w) \in \mathbb{C}^2 \mid |z|^2 + |w|^2 = \epsilon^2\} \subset \mathbb{C}^2,$$

and a plane complex algebraic curve

$$\mathcal{C} = \{(z,w) \in \mathbb{C}^2 \mid f(z,w) = 0\} \subset \mathbb{C}^2$$

defined by the squarefree polynomial $f(z,w) \in \mathbb{C}[z,w]$ with an isolated singularity in the origin $(0,0) \in \mathbb{C}^2$. Since $\mathbb{C}^2$ is isomorphic to $\mathbb{R}^4$, we replace the complex variables $z = a + ib$ and $w = c + id$ in the defining equations of the sphere S_ϵ and of the curve $\mathcal{C}$ and we obtain:

$$S_\epsilon = \{(a,b,c,d) \in \mathbb{R}^4 \mid a^2 + b^2 + c^2 + d^2 = \epsilon^2\} \subset \mathbb{R}^4,$$

and respectively

$$C = \{(a,b,c,d) \in \mathbb{R}^4 \mid p(a,b,c,d) = 0\}.$$

For small enough radius ϵ, the intersection $X_\epsilon = \mathcal{C} \cap S_\epsilon$ is a knot or a link in the 3-dimensional sphere $S_\epsilon \subset \mathbb{R}^4$ and it determines completely the local topology of $\mathcal{C}$ near its singularity $(0,0)$. We identify this 3-dimensional sphere S_ϵ topologically with $\mathbb{R}^3$ by using the stereographic projection. For the time being, we will not give a formal definition of the stereographic projection. For more information on the stereographic projection see Subsection 2.4.1. At this point, we only mention that the stereographic projection of the sphere from its north point $N(0,\epsilon)$ maps each point (z,w) of the sphere different from $(0,\epsilon)$ to the point of intersection of the line joining (z,w) and $(0,\epsilon)$ with the 3-dimensional real space $\{(a,b,c,d) \in \mathbb{R}^4 \mid d = 0\}$. The point $N(0,\epsilon)$ is mapped to ∞. In the literature, the stereographic projection is used to project the link X_ϵ from the 3-dimensional sphere $S_\epsilon \subset \mathbb{R}^4$ to $\mathbb{R}^3$. In particular we consider the stereographic projection π defined by the following mapping:

$$\begin{gathered} \pi : S_\epsilon \setminus \{N\} \subset \mathbb{R}^4 \to \mathbb{R}^3 \\ (a,b,c,d) \mapsto \begin{cases} \left(x = \dfrac{a}{\epsilon - d}, y = \dfrac{b}{\epsilon - d}, z = \dfrac{c}{\epsilon - d}\right), \text{ if } d \neq \epsilon \\ \infty, \text{ if } d = \epsilon, \end{cases} \end{gathered} \tag{2.41}$$

with inverse π^{-1} given by:

$$\begin{gathered} \pi^{-1} : \mathbb{R}^3 \to S_\epsilon \setminus \{N\} \subset \mathbb{R}^4 \\ \begin{cases} (x,y,z) \mapsto \left(a = \frac{2x\epsilon}{\kappa}, b = \frac{2y\epsilon}{\kappa}, c = \frac{2z\epsilon}{\kappa}, d = \frac{(x^2+y^2+z^2-1)\epsilon}{\kappa}\right) \\ \infty \mapsto (0,\epsilon), \end{cases} \end{gathered} \tag{2.42}$$

where $\kappa = 1 + x^2 + y^2 + z^2$. As mentioned earlier, the stereographic projection π allows us to project the link $X_\epsilon = \mathcal{C} \cap S_\epsilon$ from $S_\epsilon \subset \mathbb{R}^4$ to $\mathbb{R}^3$ by preserving its topological properties. Thus we compute the image $\pi(X_\epsilon)$ of X_ϵ through stereographic projection as follows:

$$\begin{aligned} \pi(X_\epsilon) &= \left\{(x,y,z) \in \mathbb{R}^3 \mid \exists\ (a,b,c,d) \in \mathcal{C} \cap S_\epsilon :\ \pi(a,b,c,d) = (x,y,z)\right\} = \\ &= \left\{(x,y,z) \in \mathbb{R}^3 \mid \exists\ \pi^{-1}(x,y,z) \in \mathcal{C} \cap S_\epsilon\right\}. \end{aligned} \tag{2.43}$$

Since the stereographic projection preserves all the topological properties of X_ϵ, it follows that $\pi(X_\epsilon)$ coincide with the link of the singularity for small values of the radius ϵ. In the following paragraph we include an example, in which we present the computation of the link of a singularity of a plane complex algebraic curve using the stereographic projection.

Example 16. We consider the following plane complex algebraic curve:

$$\mathcal{C} = \{(z,w) \in \mathbb{C}^2 \mid z^3 - w^2 = 0\},$$

defined by the squarefree polynomial $f(z,w) = z^3 - w^2 \in \mathbb{C}[z,w]$ with a singularity in the origin $(0,0) \in \mathbb{C}^2$. In addition, we consider the sphere of radius ϵ centered in the singularity $(0,0)$ denoted with:

$$S_\epsilon = \left\{(z,w) \in \mathbb{C}^2 \mid |z|^2 + |w|^2 = \epsilon^2\right\}.$$

Without loss of generality in this case we assume $\epsilon = 1$ and thus we obtain:

$$S_1 = \left\{(z,w) \in \mathbb{C}^2 \mid |z|^2 + |w|^2 = 1\right\}.$$

To compute the link of the singularity $(0,0)$ of the curve $\mathcal{C}$ we proceed in the following way:

- We replace the complex variables $z = a + ib$ and $w = c + id$ in the defining equations of the curve $\mathcal{C}$ and of the sphere S_ϵ identifying in this way $\mathbb{C}^2$ with $\mathbb{R}^4$. We obtain:

$$\begin{array}{l} \mathcal{C} = \{(a,b,c,d) \in \mathbb{R}^4 \mid (a+ib)^3 - (c+id)^2 = 0\} \text{ and} \\ S_1 = \left\{(a,b,c,d) \in \mathbb{R}^4 \mid a^2+b^2+c^2+d^2 = 1\right\}. \end{array} \tag{2.44}$$

- We next intersect the curve $\mathcal{C}$ with the sphere S_1 and we obtain a 1-dimensional set X_1 in $\mathbb{R}^4$:

$$\begin{array}{l} X_1 = \mathcal{C} \cap S_1 = \{(a,b,c,d) \in \mathbb{R}^4 \mid (a+ib)^3 - (c+id)^2 = 0 \text{ and} \\ \qquad\qquad a^2+b^2+c^2+d^2 = 1\}. \end{array} \tag{2.45}$$

  For $\epsilon = 1$, the intersection $X_1 = \mathcal{C} \cap S_1$ is a link, i.e. a disjoint union of embedded circles, called the link of the singularity $(0,0)$ of the plane complex algebraic curve $\mathcal{C}$.

- We project the set X_1 from $\mathbb{R}^4$ to $\mathbb{R}^3$ by using the stereographic projection π defined by the Equations (2.41) for $\epsilon = 1$. We basically need to compute the image of X_1 through the stereographic projection π by using the inverse π^{-1} defined by the Equations 2.42 for $\epsilon = 1$. We notice that by using the inverse π^{-1} of the stereographic projection π, the coordinates (a,b,c,d) of the 4-dimensional space are expressed in terms of the coordinates (x,y,z) of the 3-dimensional space, i.e. $\left(a = \frac{2x}{\kappa}, b = \frac{2y}{\kappa}, c = \frac{2z}{\kappa}, d = \frac{x^2+y^2+z^2-1}{\kappa}\right)$, where $\kappa = 1 + x^2 + y^2 + z^2$. We next replace these formulas for the variables (a,b,c,d) in terms of (x,y,z) in the defining equation of the curve $\mathcal{C}$ to compute the image $\pi(X)$ of X through the stereographic projection π. We thus obtain:

$$\begin{array}{l} \pi(X_1) = \left\{(x,y,z) \in \mathbb{R}^3 \mid \left(\frac{2x}{\kappa} + \frac{2y}{\kappa}i\right)^3 - \right. \\ \qquad\qquad \left. -\left(\frac{2z}{\kappa} + \frac{x^2+y^2+z^2-1}{\kappa}i\right)^2 = 0\right\}, \end{array} \tag{2.46}$$

  where $\kappa = 1 + x^2 + y^2 + z^2$. For simplicity reasons, we denote the defining polynomial of $\pi(X_1)$ from Equation (2.46) with $q(x,y,z) \in \mathbb{C}[x,y,z]$. Next we eliminate the denominators in the defining equation $q(x,y,z) = 0$ of $\pi(X_1)$. Since the defining polynomial $q(x,y,z) \in \mathbb{C}[x,y,z]$ of $\pi(X_1)$ is a complex polynomial, it follows that we obtain in fact two defining equations for $\pi(X_1)$ denoted with $q_1(x,y,z) = 0, q_2(x,y,z) = 0$, where $q_1(x,y,z), q_2(x,y,z) \in \mathbb{R}[x,y,z]$, given by the real and the imaginary part of $q(x,y,z)$, where by a straightforward computation we obtain the following expressions for q_1, q_2 :

$$\begin{array}{l} q_1(x,y,z) := 1 - 2x^2 + x^4 + 2y - 2x^2y - 2y^2 + +2x^2y^2 - 2y^3 + y^4 + 4xz - \\ \qquad -6z^2 + 2x^2z^2 - 2yz^2 + 2y^2z^2 + z^4 = 0 \text{ and} \\ q_2(x,y,z) := 1 - 2x^2 + x^4 + 2y - 2x^2y - 2y^2 + 2x^2y^2 - 2y^3 + y^4 + 4xz - \\ \qquad -6z^2 + 2x^2z^2 - 2yz^2 + 2y^2z^2 + z^4 = 0. \end{array}$$

  We thus obtain:

$$\pi(X_1) = \left\{(x,y,z) \in \mathbb{R}^3 \mid q_1(x,y,z) = 0, q_2(x,y,z) = 0\right\} \subset \mathbb{R}^3. \tag{2.47}$$

  We obtain that $\pi(X_1)$ is a space algebraic curve in $\mathbb{R}^3$ given as the intersection of two algebraic surfaces S_1 and S_2 in $\mathbb{R}^3$ with their defining polynomials $q_1(x,y,z) \in$

$\mathbb{R}[x, y, z]$ and respectively $q_2(x, y, z) \in \mathbb{R}[x, y, z]$. We use an algebraic geometric modeler as Axel [Wintz et al., 2006] to display the space algebraic curve $\pi(X_1)$. More information on Axel and on the methods used to display the space algebraic curve $\pi(X_1)$ are discussed in Chapter 3 and in Chapter 5. In Figure 2.54 we visualize the space algebraic curve $\pi(X_1)$.

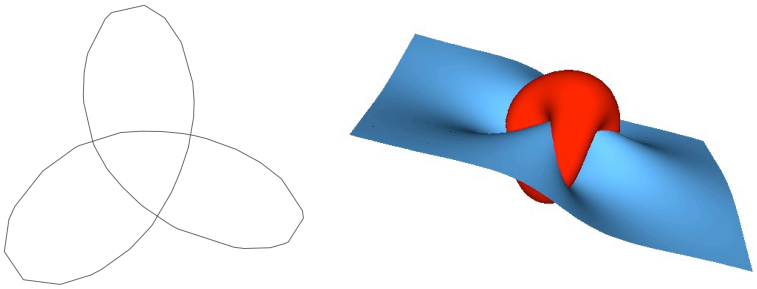


Figure 2.54: Link of the singularity $(0, 0)$ of the plane complex algebraic curve $\mathcal{C}$ given by $z^3 - w^2 = 0$. From left to right: (1) the link L of the singularity $(0, 0)$ of $\mathcal{C}$, represented by the trefoil knot; (2) the two algebraic surfaces that define as their intersection the link L. Pictures produced with Axel, see Chapter 5 for more information.

We observe that $\pi(X_1)$ is a closed curve in $\mathbb{R}^3$ that does not intersect itself, i.e. $\pi(X_1)$ is a smooth space algebraic curve in $\mathbb{R}^3$, which represents the link of the singularity $(0, 0)$ of the curve $\mathcal{C}$. In addition, we notice that the link $L := \pi(X_1)$ of the singularity $(0, 0)$ of $\mathcal{C}$ is the trefoil knot. Only two points on this trefoil knot can be seen in the real picture from Figure 2.55, points that are represented by the intersection of the plane real algebraic curve defined by $z^3 - w^2 = 0$ with the unit circle $z^2 + w^2 = 1$. In Subsection 2.4.2 we include a precise formal description for links of plane curve singularities.

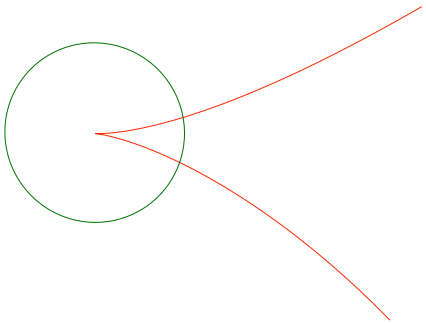


Figure 2.55: Real points of the curve $z^3 - w^2$, which are represented by the two intersection points of the plane real algebraic curve defined by $z^3 - w^2 = 0$ with a small sphere around the origin represented by the unit circle $z^2 + w^2 = 1$. Pictures produced with Axel, see Chapter 5 for more information.

2.4 Invariants of Plane Complex Algebraic Curves

2.4.1 Preliminaries

In this subsection, we introduce and we define the topological invariants of plane complex algebraic curves, which are the main topic of research in this thesis. We recall that two subsets $U \subset \mathbb{R}^k, V \subset \mathbb{R}^n$ are topologically equivalent or homeomorphic if there exists a bijective function $\varphi : U \to V$ such that both φ and its inverse are continuous. In this case, φ is called an homeomorphism. We now introduce the concept of homeomorphism of pairs in the following way:

Definition 47. A pair (X, A) of spaces is a topological space together with a subspace $A \subseteq X$. A mapping $\varphi : (X, A) \to (Y, B)$ of pairs is a continuous mapping $\varphi : X \to Y$ with $\varphi(A) \subseteq B$. A homeomorphism $\varphi : (X, A) \to (Y, B)$ of pairs is a mapping of pairs, which is a homeomorphism $\varphi : X \to Y$ and which induces a homeomorphism $\varphi/_A : A \to B$.

In this thesis, the topological invariants of a plane complex algebraic curve $\mathcal{C}$ are those topological properties of $\mathcal{C}$ and its singularities that are unchanged under homeomorphism of small disks around 0 mapping the first curve onto the second curve.

An important notion for our study is the stereographic projection. Firstly, we consider the stereographic projection from $\mathbb{R}^3$ to $\mathbb{R}^2$ as a mapping that projects a sphere onto a plane. It is constructed as in Figure 2.56: we take a sphere; we draw a line from the north pole N of the sphere to a point $\hat{P}$ in the equator plane to intersect the sphere at a point P. We say that the stereographic projection of $\hat{P}$ is P. Since the stereographic projection is a bijective map, it follows that this map works in both ways. So we can think of projecting points from the sphere down to the plane using the same type of intersecting line. Each point from the sphere has a correspondence point in the plane, except for the north pole of the sphere. Since points close to the north pole of the sphere map into the plane far from the sphere, the north pole is said to represent the plane's "point at infinity". We remark that 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 R^{n+1} in the following way: we consider a n-sphere in $\mathbb{R}^{n+1}$ of radius 1 denoted with

$$S^n = \{(x_1, x_2, ..., x_{n+1}) \subset \mathbb{R}^{n+1} \mid x_1^2 + x_2^2 + ... + x_{n+1}^2 = 1\},$$

and we consider the north point of the n-sphere to be $N(0, ..., 1) \in S^n$. If H is a hyperplane in $\mathbb{R}^{n+1}$ not containing N, then the stereographic projection of the point $P \in S^n \setminus N$ is the point $\hat{P}$ of the intersection of the line NP with H. The stereographic projection is an homeomorphism from $S^n \setminus N \subset \mathbb{R}^{n+1}$ to $\mathbb{R}^n$.

We remember that a plane complex algebraic curve $\mathcal{C} \subset \mathbb{C}^2$ is in fact a 2-dimensional object in $\mathbb{R}^4$, since we know that $\mathbb{C}^2$ is isomorphic with $\mathbb{R}^4$. For our study, we use the stereographic projection from $\mathbb{R}^4$ to $\mathbb{R}^3$ to project objects from $\mathbb{R}^4$ to $\mathbb{R}^3$ by preserving their topological properties. In other words, the stereographic projection preserves the invariants of a plane complex algebraic curve. In this thesis, we study the following invariants of a plane complex algebraic curve and its singularities:

- the link of each singularity Q of the plane complex algebraic curve $\mathcal{C}$;
- the Alexander polynomial of the link of each singularity Q of $\mathcal{C}$, which we also call simply the Alexander polynomial of the singularity Q;
- the delta-invariant of each singularity Q of $\mathcal{C}$;

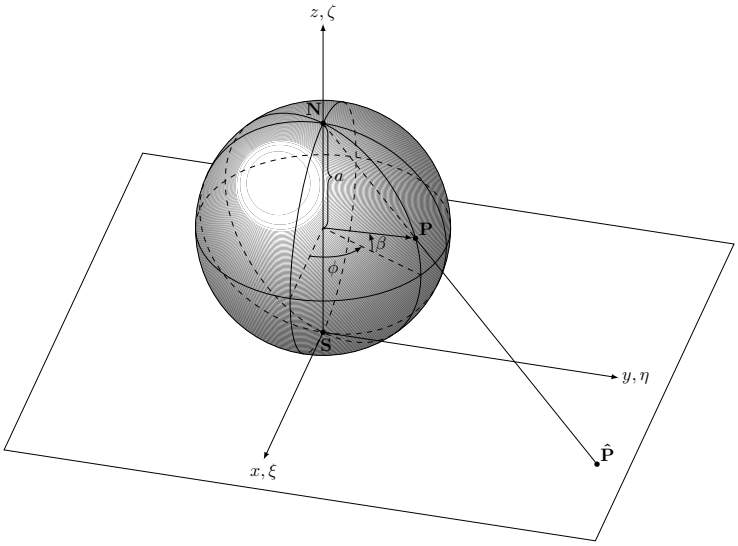


Figure 2.56: Stereographic projection from $\mathbb{R}^3$ to $\mathbb{R}^2$. Picture generated with PGF/TikZ by T. M. Trzeciak.

- the Milnor number of each singularity Q of $\mathcal{C}$;
- the genus of $\mathcal{C}$;
- the Euler characteristic of the Riemann surfaces attached to $\mathcal{C}$.

In the following subsections we define each of these invariants and we explain the main reasons for studying them.

2.4.2 Link of a Singularity

In this subsection, we define the link of a singularity of a plane complex algebraic curve and we formally explain how we can use the link of a singularity to study the local topology of plane complex algebraic curves. We add that some general remarks on the local topology of plane complex algebraic curves was made in Subsection 2.3.3. However, in this subsection we include a more formal study for the local topology of plane complex algebraic curves. We recall that in this thesis we consider $\mathbb{C}^2$ isomorphic to $\mathbb{R}^4$, which we denote with $\mathbb{C}^2 \simeq \mathbb{R}^4$. Therefore, we consider a plane complex algebraic curve $\mathcal{C} \subset \mathbb{C}^2$ as a real 2-dimensional object in $\mathbb{R}^4$. One of the main purposes of this thesis is to study and to understand the topology of 2-dimensional objects near their singularities, topology that can be determined by introducing the notion of a link of a singularity.

We introduce the notion of topological equivalence between two isolated singularities of two plane complex algebraic curves following [Tráng, 1973]. We consider that both singularities are centered in the origin $(0, 0)$.

Definition 48. Let $\mathcal{C} = \{(z, w) \in \mathbb{C}^2 \mid f(z, w) = 0\}$ be a plane complex algebraic curve defined by $f(z, w) \in \mathbb{C}[z, w]$, with an isolated singularity in the origin $(0, 0) \in \mathbb{C}^2$. Let $\mathcal{D} = \{(z, w) \in \mathbb{C}^2 \mid g(z, w) = 0\}$ be another plane complex algebraic curve defined by $g(z, w) \in \mathbb{C}[z, w]$, with an isolated singularity in the origin $(0, 0) \in \mathbb{C}^2$. We say that the isolated singularities $(0, 0)$ of $\mathcal{C}$ and $\mathcal{D}$ are topologically equivalent (or that $\mathcal{C}$ and $\mathcal{D}$ have

the same topological type at $(0,0)\big)$ if there exist two neighborhoods $U, V \subset \mathbb{C}^2$ of $(0,0)$ and a homeomorphism $\varphi : U \to V$ such that $\varphi(\mathcal{C} \cap U) = \mathcal{D} \cap V$ and $\varphi\big((0,0)\big) = (0,0)$.

For our study we use the following essential theorem:

Theorem 10. *([Milnor, 1968]) 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 f. 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 $(0,0)$; S_ϵ is the sphere centered in the origin $\vec{0}$ and of radius ϵ; and D_ϵ is the disk centered in $\vec{0}$ of radius ϵ. Then, for sufficiently small ϵ, $X_\epsilon = S_\epsilon \cap V$ is a $(2n-1)$-dimensional nonsingular set and the pair $(D_\epsilon, D_\epsilon \cap V)$ is homeomorphic to the pair consisting of the cone over S_ϵ and the cone over $X_\epsilon = S_\epsilon \cap V$.*

We now discuss the meaning of Milnor's theorem[1]. In the curve case of Theorem 10, we have that $n = 1$ and that all the singularities are isolated. Moreover, we alternatively use the following notation for denoting the plane complex algebraic curve: $V = \big\{(z,w) \in \mathbb{C}^2 \mid f(z,w) = 0\big\} := f^{-1}(0)$.

In addition, in the case $n = 1$, Milnor's theorem says that there exists $\epsilon_0 \in \mathbb{R}_{>0}$ such that for any $\epsilon_1, \epsilon_2 \in \mathbb{R}_{>0}$ with $\epsilon_1 < \epsilon_0$ and $\epsilon_2 < \epsilon_0$, the images of $X_{\epsilon_1} = S_{\epsilon_1} \cap V \subset S_{\epsilon_1}$ and $X_{\epsilon_2} = S_{\epsilon_2} \cap V \subset S_{\epsilon_2}$ through stereographic projection are links and they are equivalent, i.e. $D_{\epsilon_1} \cap \mathcal{C}$ and $D_{\epsilon_2} \cap \mathcal{C}$ are homeomorphic. In addition, for any $0 < \epsilon < \epsilon_0$ the image of X_ϵ through stereographic projection is called *the link L of the singularity* of f (or of $\mathcal{C}$) at $(0,0)$ and it is well-defined up to homeomorphism of pairs. In this case, the link L defined as the image of $X_\epsilon \subset S_\epsilon$ through stereographic projection determines the topological type of the singularity $(0,0)$ of $\mathcal{C}$. In the literature, a link is called algebraic if it is equivalent to the link of a plane curve singularity. Under these assumptions, we actually notice that Theorem 10 asserts that the equivalence class of the link of a singularity determines the homeomorphism type of the singularity itself.

Under the same hypotheses from Theorem 10 and considering S^1 the unit circle, Milnor fibration theorem states that the mapping $\phi : S_\epsilon \setminus L \to S^1, \phi(z,w) = f(z,w)/|f(z,w)|$ is a fibration, i.e. the complement $S_\epsilon \setminus L$ is a union of smooth surfaces, each being the preimage of one point.

Based on the previous observations, we notice that we can compute the link of a singularity Q of a plane complex algebraic curve $\mathcal{C}$ in the following main steps:

1. we translate the singularity Q of the curve $\mathcal{C}$ in the origin O by an affine change of coordinates;

2. we consider the curve $\mathcal{C}$ with an isolated singularity in the origin O, which is a 2-dimensional set in the 4-dimensional space $\mathbb{R}^4$, i.e. $\mathcal{C}$ is a surface in $\mathbb{R}^4$. We take the sphere centered in the origin and of a small radius ϵ;

3. we intersect the curve $\mathcal{C}$ with this sphere obtaining a 1-dimensional set X_ϵ in the 4-dimensional space, i.e. X_ϵ is a curve in $\mathbb{R}^4$. Based on Theorem 10 this 1-dimensional set is the link of the singularity O for sufficiently small values of the radius ϵ, i.e. the 1-dimensional set is an algebraic link for sufficiently small values of the radius ϵ. Next,

[1]We wish to emphasize that J. Milnor, the author of Theorem 10 and the founder of other important results used in this thesis, was awarded the Abel Prize in 2011 by the Norwegian Academy of Science and Letters "for pioneering discoveries in topology, geometry and algebra." The author would wish to congratulate J. Milnor on this special occasion and to express her gratitude and her admiration towards the innovative work and results conducted by J. Milnor. It is the inspiring work of such great minds that leads the young and brilliant students to aim for obtaining a PhD in mathematics or even to devote their talent towards a career in research!

we follow Brauner and Heergaard technique [Brauner, 1928] to move the algebraic link from the 4-dimensional space to the 3-dimensional space using the stereographic projection. As discussed earlier, the stereographic projection allows us not only to project the algebraic link from $\mathbb{R}^4$ into $\mathbb{R}^3$, but it actually preserves all the topological properties of the algebraic link from the the 4-dimensional Euclidean space into the 3-dimensional Euclidean space.

We notice that the last step from the previous computation depends on the input parameter ϵ, which represents the radius of the sphere that we intersect with the given plane complex algebraic curve. More exactly, from Milnor's theorem we know that the intersection of a plane complex algebraic curve $\mathcal{C}$ with a singularity in the origin with a small sphere S_ϵ centered in the origin and of radius ϵ is the link of the singularity for sufficiently small values of the parameter ϵ. However, we notice that Milnor's theorem does not provide a constructive method for determining a general formula for the parameter ϵ for which the intersection $\mathcal{C} \cap S_\epsilon$ is the link of the singularity. This is the key point for introducing the notion of ϵ-link of a singularity of a plane complex algebraic link in Section 2.5. We sometimes refer to the ϵ-link of a plane curve singularity as simply the approximate link of the plane curve singularity.

We recall that in Subsection 2.3.3 we have informally introduced the notion of an algebraic link. Thus we say that a link is algebraic if it arises as the intersection of an algebraic curve with a sufficiently small sphere. We notice that this process coincides with the computation of the link of a singularity, as described in this subsection. We remember that two links are equivalent if there exists an orientation preserving homeomorphism on $\mathbb{R}^3$ that maps one link onto the other. Moreover, this equivalence is called (ambient) isotopy. We can now introduce the formal definition of an algebraic link:

Definition 49. A link is called algebraic if it is equivalent to the link of a plane curve singularity.

Some examples of algebraic links are the trefoil knot, the Hopf link, and all the torus knots. The Borromean rings are not an algebraic link, i.e they do not arise from the intersection of a plane complex algebraic curve with a sufficiently small sphere. For more information on these types of knots and links, the reader can check Subsection 2.3.3. Following [Brieskorn and Knorrer, 1986], we recall the description of a torus knot as the intersection of a plane complex algebraic curve with a 3-dimensional sphere. We consider the plane complex algebraic curve $\mathcal{C} = \{(z, w) \in \mathbb{C}^2 \mid z^p + w^q = 0\}$ with an isolated singularity in the origin $(0, 0)$ and with p, q being coprime positive integers. In addition, we take the 3-dimensional sphere of radius 1 and centered in the origin $(0, 0)$ denoted with $S_1 = \{(z, w) \in \mathbb{C}^2 \mid |z|^2 + |w|^2 = 1\} \subset \mathbb{C}^2$. Then the intersection $\mathcal{C} \cap S_1$ is an (p, q)-torus knot, i.e. a knot that lies on the surface of a torus, with p and q counting the number of times that the knot winds around the two cycles of the torus.

We notice that the construction of the link of a singularity relates singularity theory to knot theory. In the following subsections we derive a polynomial invariant for algebraic links, which completely distinguishes among all the algebraic links. From this polynomial invariant we derive formulas for other topological invariants of a plane complex algebraic curve.

2.4.3 Alexander Polynomial of a Singularity

In this subsection we introduce the Alexander polynomial of a singularity of a plane complex algebraic curve and we mention the importance of computing the Alexander polynomial. On one side, in Subsection 2.3.3 we introduced basic notions from knot theory and we

saw that at present there exist several link invariants in knot theory for distinguishing links among one another, such as the tricolorability, the unknotting number, the Alexander polynomial, the Jones polynomial, etc. In addition, we discussed in more details one of these invariants, i.e. the Alexander polynomial of a link. Moreover, we presented a method for the computation of the Alexander polynomial of a knot based on the fundamental group of the complement of the knot in $\mathbb{R}^3$. However, we saw that there exists no complete invariant for links, i.e. there exists no invariant that distinguishes among all the links.

On the other side, in Subsection 2.4.2 we introduced the notion of a link of a singularity, we presented a method for computing the link of a singularity and we explained how the link of the singularity can be used to study the local topology of the singularity of a plane complex algebraic curve. Furthermore based on the existence of links of singularities, we introduced a special class of links called the algebraic links. In particular, we said that a link is algebraic if it is equivalent to the link of a plane curve singularity.

Since the link of the singularity of a plane complex algebraic curve offers essential information on the local topology of the curve itself as explained in Subsection 2.4.2, in this thesis, for our purpose, we are thus interested in studying invariants of links of singularities, i.e. we are motivated to study invariants of algebraic links. An important result in this direction of research was proved by Yamamoto in 1984, see [Yamamoto, 1984]. Yamamoto showed that the Alexander polynomial is a complete invariant for the algebraic links, that is the Alexander polynomial uniquely identifies all the algebraic links up to an (ambient) isotopy. This result has an important consequence for our study, which we explain in the following paragraph:

- We consider two singularities of two plane complex algebraic curves denoted with Q_1 and Q_2.
- We assume that we can compute the links of the two singularities Q_1 and Q_2 denoted with $L(Q_1)$ and respectively with $L(Q_2)$ using the method described in Subsection 2.4.2.
- We can use Yamamoto's result as follows: if the Alexander polynomials of $L(Q_1)$, and of $L(Q_2)$ are equal, then Q_1 and Q_2 have the same topology, else they have different topology.

In this way, we can use the Alexander polynomial of the link of a singularity to distinguish the topological type of the singularity itself. Therefore we require a computational method for determining the Alexander polynomial of the link of a singularity. In this subsection, we present a straightforward algorithm to compute the Alexander polynomial attached to the link of a singularity by using combinatorial objects from knot theory such as the oriented diagram of the link, i.e. its arcs and its crossings, and the Reidemeister moves. This method is in fact the Alexander's method used for computing the Alexander polynomial of a link, as shortly mentioned in Subsection 2.3.3. In this thesis, we use the following terminology: if Q is the singularity of a plane complex algebraic curve and $L(Q)$ is the link of the singularity Q, then we sometimes call the Alexander polynomial of the link $L(Q)$ of the singularity Q simply the Alexander polynomial of the singularity Q or the Alexander polynomial of the algebraic link $L(Q)$.

First of all, we recall several notions from knot theory as discussed in Subsection 2.3.3. These notions are essential for the computation of the Alexander polynomial of a link. In our study, we are interested in the following type of combinatorial information of an oriented diagram of a link:

- the type of each crossing in the oriented diagram. We recall that we distinguish between two types of crossings in an oriented diagram, i.e. lefthanded or righthanded

crossing. In particular, 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 $+1$ as indicated by the dotted round arrow in Figure 2.57. Moreover, for the computation of the Alexander polynomial of the link of a singularity we introduce the following essential terminology: whether lefthanded or righthanded, each crossing of a link diagram is determined by three arcs and we denote the overgoing arc with the label i, and the undergoing arcs with the labels j and k as seen in Figure 2.57.

- the arc of the oriented diagram, which is defined as the part of a diagram between two undercrossings, see Figure 2.58. Furthermore, we remember that the number of arcs in a diagram equals the number of crossings in the same diagram, see Figure 2.58.

Figure 2.57: Types of crossings: lefthanded crossing (-1) and righthanded crossing (+1), together with the labels for the 3 arcs of a crossing.

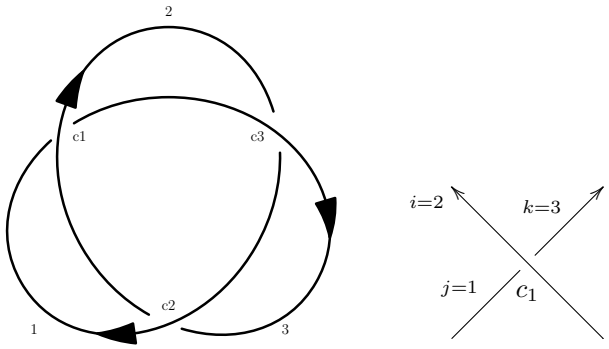


Figure 2.58: Oriented diagram of the trefoil knot with 3 arcs denoted with $\{1, 2, 3\}$ and 3 crossings denoted with $\{c_1, c_2, c_3\}$. Example of arcs labeling for the crossing denoted c_1, which is lefthanded.

For introducing the Alexander polynomial of a link, we introduce some preliminary definitions based mainly on the book of [Livingston, 1993]. Firstly, we introduce the notion of a labelling matrix for the oriented link diagram $D(L)$ of a link L :

Definition 50. Let $D(L)$ be an oriented link diagram of the link L with r components and n crossings $c_q : q \in \{1, ..., n\}$. We denote the arcs of $D(L)$ with the labels $\{1, ..., n\}$ and separately the crossings of $D(L)$ with $\{1, ..., n\}$. 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 c_q as follows:

- at b_{q1} store the type of the crossing c_q, i.e. $+1$ or -1;

- at b_{q2} store the label of the arc i of c_q in $D(L)$;
- at b_{q3} store the label of the arc j of c_q in $D(L)$;
- at b_{q4} store the label of the arc k of c_q in $D(L)$, see Figure 2.58 for an example.

Secondly, we define the prealexander matrix for the oriented diagram $D(L)$ of the link L in the following way:

Definition 51. Let $D(L)$ be an oriented link diagram of the link L with r components and n crossings $c_q : q \in \{1, ..., n\}$. We denote the arcs and the crossings of $D(L)$ as in Definition 50. We consider $LM(L)$ the labelling matrix of $D(L)$ as in Definition 50. We denote the prealexander matrix of L with $PM(L) \in \mathcal{M}(n, n, \mathbb{Z}[t_1, t_2, ..., t_r])$. If $D(L)$ has no crossings, then $PM(L) \in \mathcal{M}(0, 0, \emptyset)$, otherwise we define $PM(L)$ row by row for each crossing c_q depending on $LM(L)$. For each crossing c_q we consider the variable t_s, where $s \in \{1, ..., r\}$ is the s-th knot component of $D(L)$, which contains the overgoing arc that determines the crossing c_q. Then:

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

Before we define the Alexander polynomial attached to a link, we recall some essential notions from algebra:

- We say that a minor M_{ij} is the reduced determinant of a determinant expansion that is formed by omitting the ith row and the jth column of a matrix A. We recall that the determinant expansion is a method for computing the determinant of a given square matrix A. If we denote by $|A|$ the determinant of a square matrix A, then the determinant expansion of the matrix is given by $|A| = \sum_{i=1}^{k} (-1)^{i+j} a_{ij} M_{ij}$, where M_{ij} is the minor of A obtained by taking the determinant of A for which the row i and the column j were eliminated. We add that determinant expansions are efficient for matrices of small sizes, whereas for matrices of larger sizes the Gaussian elimination method is more efficient.
- In addition, a normalized polynomial is a polynomial in which the term of the lowest degree is a positive constant.

We now define the Alexander polynomial of the oriented diagram $D(L)$ of the link L depending on the number of knot components in L :

Definition 52. Let $D(L)$ be an oriented link diagram of the link L with r components and n crossings, let $LM(L)$ be its labelling matrix as in Definition 50 and let $PM(L)$ be its prealexander matrix as in Definition 51. Then:

1. The univariate Alexander polynomial [Livingston, 1993] denoted with

$$\Delta(t_1) \in \mathbb{Z}[t_1^{\pm 1}]$$

is the normalized polynomial computed as the determinant of any $(n-1)\times(n-1)$ minor of the prealexander matrix of $D(L)$.

2. The multivariate Alexander polynomial [Cimasoni, 2004] denoted with
$$\Delta(t_1,...,t_r)\in\mathbb{Z}[t_1^{\pm 1},...,t_r^{\pm 1}]$$
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 $D(L)$.

If $PM(L)\in\mathcal{M}(0,0,\emptyset)$, then we define $\Delta(t_1)=1$ and $\Delta(t_1,...,t_r)=0$.

In Definition 52, the univariate polynomial computed as the determinant of any $(n-1)\times(n-1)$ minor of the prealexander matrix of $D(L)$ depends on the choice of the original oriented diagram $D(L)$ of a knot and its labellings. Alexander's result [Alexander, 1928] is that although the choice of the original oriented diagram of a knot and its labellings may produce different polynomials, any of them will differ by a multiple of $\pm t_1^k$, for some integer k. Thus, if we normalize the polynomial to have a positive constant term, the resulting Alexander polynomial will be a knot invariant. A similar argument follows from [Cimasoni, 2004] for the multivariate Alexander polynomial attached to a link with at least 2 knot components. Therefore we introduce the univariate and the multivariate Alexander polynomial as a normalized polynomial in Definition 52.

We observe that the Definition 50, the Definition 51 and the Definition 52 are derived from Fox's method for the computation of the Alexander polynomial of a link based on the fundamental group of the complement of the link in $\mathbb{R}^3$, as described in Subsection 2.3.3.

From Definition 50, Definition 51 and Definition 52, we thus distinguish three steps for the computation of the Alexander polynomial Δ of an oriented link diagram $D(L)$ of an arbitrary link L, which are sketched in the following diagram:

$$\boxed{D(L)}\xrightarrow{(1)}\boxed{\text{Labelling matrix}(L)}\xrightarrow{(2)}\boxed{\text{Prealexander matrix}(L)}\xrightarrow{(3)}\boxed{\Delta},$$

and that means:

1. in the first step denoted (1), we compute the labelling matrix $LM(L)$ of the arbitrary link L from the oriented diagram $D(L)$ of the link;
2. in the second step denoted (2), we compute the prealexander matrix $PM(L)$ of the link L from the labelling matrix $LM(L)$;
3. in the third and last step denoted (3), we compute the Alexander polynomial Δ of the link L from the prealexander matrix $PM(L)$.

In the following paragraphs, we include several examples for the computation of the Alexander polynomial of an arbitrary link of a singularity to familiarize the reader with these types of computations.

Example 17. In this example we compute the Alexander polynomial of the oriented diagram of the trefoil knot L from Figure 2.58. We add that the trefoil knot is an algebraic knot. We first give the diagram an arbitrary orientation, we denote the crossings of the diagram with the labels $\{c_1,c_2,c_3\}$ and separately the arcs of the diagram with the labels $\{1,2,3\}$ as indicated in Figure 2.58. We then compute the labelling matrix of $LM(L)$ with Definition 50 and we obtain:

$$LM(L)=\left(\begin{array}{c|cccc} & \text{type} & \text{label}_i & \text{label}_j & \text{label}_k \\ \hline c_1 & -1 & 2 & 1 & 3 \\ c_2 & -1 & 1 & 3 & 2 \\ c_3 & -1 & 3 & 2 & 1 \end{array}\right).$$

From $LM(L)$ we compute the prealexander matrix of $D(L)$ with Definition 51. We notice that L has only one knot component so $s = 1$ in Definition 51. We obtain the following expression for the prealexander matrix $PM(L)$:

$$PM(L) = \left(\begin{array}{c|ccc} & \text{label}_i & \text{label}_j & \text{label}_k \\ \hline c_1 & 2 & 1 & 3 \\ -1 & 1-t_1 & t_1 & -1 \\ \hline c_2 & 1 & 3 & 2 \\ -1 & 1-t_1 & t_1 & -1 \\ \hline c_3 & 3 & 2 & 1 \\ -1 & 1-t_1 & t_1 & -1 \end{array}\right) = \left(\begin{array}{c|ccc} & 1 & 2 & 3 \\ \hline c_1 & t_1 & 1-t_1 & -1 \\ c_2 & 1-t_1 & -1 & t_1 \\ c_3 & -1 & t_1 & 1-t_1 \end{array}\right).$$

From $PM(L)$ we compute the univariate Alexander polynomial with Definition 52 and we obtain:

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

$$\Delta(t_1) = \text{Normalize}(-t_1^2 + t_1 - 1) = t_1^2 - t_1 + 1,$$

where we denote with Minor_{33} the minor of $PM(L)$ obtained by taking the determinant of $PM(L)$ for which the 3th row and respectively the 3th column of $PM(L)$ were eliminated.

Example 18. In this example we compute the Alexander polynomial of the cinquefoil link L, which is a link with one knot component. We consider the cinquefoil link from Figure 2.59, which is an algebraic link.

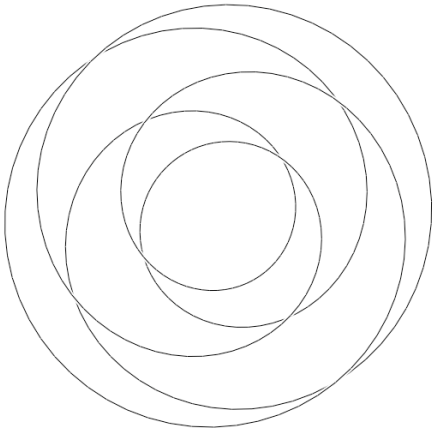


Figure 2.59: Oriented counterclockwise diagram of the cinquefoil algebraic knot with 8 arcs and 8 lefthanded crossings. Picture produced with 3D-XplorMath-J Applet. We denote the crossings from the upperleft to the lowerright corner with $\{c_1, c_2, c_3, c_4\}$ and the crossings from the lowerleft to the upperright corner with $\{c_5, c_6, c_7, c_8\}$.

We first consider the diagram to have a counterclockwise orientation and we denote the arcs of the oriented diagram with the labels $\{1, ..., 8\}$, and the crossings with the labels $\{c_1, ..., c_8\}$ as indicated in Figure 2.59. We then compute the labelling matrix of this algebraic link

denoted $LM(L)$ by using Definition 50 and we obtain:

$$LM(L) = \left(\begin{array}{c|cccc} & type & label_i & label_j & label_k \\ \hline c_1 & -1 & 1 & 8 & 2 \\ c_2 & -1 & 2 & 5 & 4 \\ c_3 & -1 & 1 & 6 & 7 \\ c_4 & -1 & 2 & 3 & 1 \\ c_5 & -1 & 1 & 4 & 3 \\ c_6 & -1 & 2 & 1 & 6 \\ c_7 & -1 & 1 & 2 & 5 \\ c_8 & -1 & 2 & 7 & 8 \end{array}\right).$$

Based on the computed labelling matrix $LM(L)$, we now compute the prealexander matrix of the cinquefoil algebraic link using Definition 51. Since L has only one knot component, it follows that $s = 1$ in Definition 51. We obtain the following expression for the prealexander matrix $PM(L)$:

$$PM(L) = \left(\begin{array}{c|ccc} & label_i & label_j & label_k \\ \hline c_1 & 1 & 8 & 2 \\ -1 & 1-t_1 & t_1 & -1 \\ \hline c_2 & 2 & 5 & 4 \\ -1 & 1-t_1 & t_1 & -1 \\ \hline c_3 & 1 & 6 & 7 \\ -1 & 1-t_1 & t_1 & -1 \\ \hline c_4 & 2 & 3 & 1 \\ -1 & 1-t_1 & t_1 & -1 \\ \hline c_5 & 1 & 4 & 3 \\ -1 & 1-t_1 & t_1 & -1 \\ \hline c_6 & 2 & 1 & 6 \\ -1 & 1-t_1 & t_1 & -1 \\ \hline c_7 & 1 & 2 & 5 \\ -1 & 1-t_1 & t_1 & -1 \\ \hline .c_8 & 2 & 7 & 8 \\ -1 & 1-t_1 & -1 & t_1 \end{array}\right) =$$

$$= \left(\begin{array}{c|cccccccc} & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ \hline c_1 & 1-t_1 & -1 & 0 & 0 & 0 & 0 & 0 & t_1 \\ c_2 & 0 & 1-t_1 & 0 & -1 & t_1 & 0 & 0 & 0 \\ c_3 & 1-t_1 & 0 & 0 & 0 & 0 & t_1 & -1 & 0 \\ c_4 & -1 & 1-t_1 & t_1 & 0 & 0 & 0 & 0 & 0 \\ c_5 & 1-t_1 & 0 & -1 & t_1 & 0 & 0 & 0 & 0 \\ c_6 & t_1 & 1-t_1 & 0 & 0 & 0 & -1 & 0 & 0 \\ c_7 & 1-t_1 & t_1 & 0 & 0 & -1 & 0 & 0 & 0 \\ c_8 & 0 & 1-t_1 & 0 & 0 & 0 & 0 & t_1 & -1 \end{array}\right).$$

Finally, by using Definition 52, we compute the Alexander polynomial of the cinquefoil algebraic link by using the computed labelling matrix $LM(L)$ and the computed prealexander matrix $PM(L)$. Since the cinquefoil algebraic link has one knot component (i.e. it is a knot), we denote its Alexander polynomial with $\Delta(t_1)$ and by Definition 52 for the univariate case of the Alexander polynomial we obtain:

$$\det\Big(\text{Minor}_{77}\big(PM(L)\big)\Big) = -t_1^5 + t_1^4 - t_1^3 + t_1^2 - t_1,$$

$$\Delta(t_1) = \text{Normalize}(-t_1^5 + t_1^4 - t_1^3 + t_1^2 - t_1) = t_1^4 - t_1^3 + t_1^2 - t_1 + 1,$$

where we denote with Minor_{77} the minor of $PM(L)$ obtained by taking the determinant of $PM(L)$ for which the 7th row and respectively the 7th column of the prealexander matrix $PM(L)$ were eliminated.

Example 19. In this example we compute the Alexander polynomial of the oriented diagram of the Hopf link L from Figure 2.60. We add that the Hopf link is an algebraic link and it has 2 knot components.

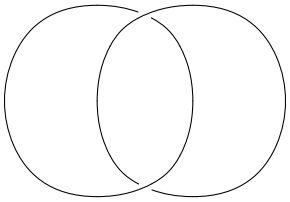


Figure 2.60: Oriented clockwise diagram of the Hopf link with 2 lefthanded crossings and 2 arcs. We denote the crossings from up to down with the labels $\{c_1, c_2\}$ and we denote the arcs from left to right with the labels $\{1, 2\}$.

We first give the diagram of the Hopf link a clockwise orientation (i.e. the two knot components of the Hopf link have both a clockwise orientation). In addition, we denote the crossings of the diagram with the labels $\{c_1, c_2\}$ and separately we denote the arcs of the same diagram with the labels $\{1, 2, \}$ as indicated in Figure 2.58. We then compute the labelling matrix of $LM(L)$ with Definition 50 and we obtain:

$$LM(L) = \left(\begin{array}{c|cccc} & \text{type} & \text{label}_i & \text{label}_j & \text{label}_k \\ \hline c_1 & -1 & 2 & 1 & 1 \\ c_2 & -1 & 1 & 2 & 2 \end{array}\right).$$

From $LM(L)$ we compute the prealexander matrix $PM(L)$ of the oriented diagram of the Hopf link with Definition 51. Since L has 2 knot components, it follows that $s = \{1, 2\}$ in Definition 51. In addition, we observe that for the crossing denoted with c_1 the overgoing arc is represented by the arc with the label 2, whereas for the crossing denoted with c_2 the overgoing arc is given by the arcs with the label 1. Therefore, we obtain the following expression for the prealexander matrix $PM(L)$:

$$PM(L) = \left(\begin{array}{c|ccc} & \text{label}_i & \text{label}_j & \text{label}_k \\ \hline c_1 & 2 & 1 & 1 \\ -1 & 1-t_2 & t_2 & -1 \\ \hline c_2 & 1 & 2 & 2 \\ -1 & 1-t_1 & t_1 & -1 \end{array}\right) = \left(\begin{array}{c|cc} & 1 & 2 \\ \hline c_1 & t_2-1 & 1-t_2 \\ c_2 & 1-t_1 & t_1-1 \end{array}\right).$$

From the prealexander matrix $PM(L)$ we compute the multivariate Alexander polynomial of the Hopf link with Definition 52. We first have to compute all the 2×2 minors of $PM(L)$. We denote by M_{ij} with $i, j \in \{1, 2\}$ the minor of $PM(L)$ obtained by taking the determinant of $PM(L)$ for which the ith row and respectively the jth column were eliminated. We obtain the following list of minors:

- $M_{11} = t_1 - 1$,
- $M_{12} = 1 - t_1$,

- $M_{21} = 1 - t_2$,
- $M_{22} = t_2 - 1$.

Definition 52 states that the Alexander polynomial is determined by the greatest common divisors of all the 2×2 minors of $PM(L)$. Since the Hopf Link L has 2 knot components we denote the Alexander polynomial of the Hopf link with $\Delta(t_1, t_2)$. It follows that the Alexander polynomial of the Hopf link is given by the following formula:

$$\Delta(t_1, t_2) = \gcd(t_1 - 1, 1 - t_1, 1 - t_2, t_2 - 1) = 1,$$

where by gcd we denote the greatest common divisor of polynomials.

For our study the Alexander polynomial of the link of a singularity of a plane complex algebraic curve (also called the Alexander polynomial of the plane curve singularity or simply the Alexander polynomial of the singularity) has significant importance:

- The Alexander polynomial distinguishes among all the algebraic links, and thus it is a complete invariant for links of singularities. We can use the Alexander polynomial of a link to extract essential information on the local topology of a plane curve singularity.
- From the Alexander polynomial of the singularity Q we can compute other invariants of the plane complex algebraic curve and its singularities as described in the next subsections, e.g. delta-invariant, Milnor number, genus.

As a remark we mention that the computation of the Alexander polynomial of the singularity Q of a plane complex algebraic curve $\mathcal{C}$ is derived from the link L of the singularity Q. This link L arises from the intersection of the curve $\mathcal{C}$ with a sphere of small radius ϵ centered in Q. Thus the computation of the Alexander polynomial of a singularity depends on the computation of the link of the singularity, which is conditioned by the parameter ϵ. This is in fact the main reason for introducing the ϵ-Alexander polynomial of the singularity in Section 2.5.

2.4.4 Delta-Invariant of a Singularity

In this subsection we give a formula for the delta-invariant of each singularity of a plane complex algebraic curve. This formula is derived from the Alexander polynomial of the plane curve singularity. We use Milnor results [Milnor, 1968] for computing the delta-invariant of the isolated singularity $(0, 0)$ of a plane complex algebraic curve. We proceed in the following way:

- we consider $\mu(O)$ a positive integer that measures the amount of degeneracy at the critical point $O(0, 0)$ of the complex polynomial $f(z, w)$. In fact, $\mu(O)$ is the Milnor number. We use the same notations from Subsection 2.4.2. We recall that we use the following notation for the plane complex algebraic curve defined by the polynomial $f(z, w)$, i.e. $V = \{(z, w) \in \mathbb{C}^2 \mid f(z, w) = 0\} := f^{-1}(0)$. In addition, we consider the sphere centered in the singularity $(0, 0)$ and of a small radius ϵ denoted with $S_\epsilon = \{(z, w) \in \mathbb{C}^2 \mid |z|^2 + |w|^2 = \epsilon^2\}$. It is shown that $\mu(O)$ is the degree of the characteristic polynomial Λ of the link $L = V \cap S_\epsilon$ determined by $V := f^{-1}(0)$. The characteristic polynomial denoted with Λ is defined in the following way:
  - Λ coincides with the Alexander polynomial $\Delta(t)$, if L has one knot component,
  - and $\Lambda = \dfrac{(t-1)}{\pm t^i}\Delta(t, ..., t)$, if L has more than one knot components.

It is proven thus that $\mu(O)$ is the degree of the characteristic polynomial Λ. Based on this observation we deduce the following formula for the Milnor number $\mu(O)$, formula that is derived from the Alexander polynomial:

- $\mu(O)$ is the degree of the Alexander polynomial, if L has one knot component,
- and $\mu(O)$ 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 the origin $(0, 0)$. 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 method for the computation of the delta-invariant of a plane curve singularity on the following theorem proved by Milnor:

Theorem 11. *([Milnor, 1968]) Suppose that r branches of the curve $V := f^{-1}(0)$ pass through the origin $O(0, 0)$, which is an isolated singularity for V. Then the delta-invariant of the isolated singularity $O(0, 0)$ denoted with $\delta(O) > 0$ is related to the Milnor number $\mu(O)$ by the equation $2\delta(O) = \mu(O) + r - 1$ and it is always a positive integer.*

From Theorem 11 we derive a formula for the delta-invariant of the isolated singularity $O(0, 0)$ of the plane complex algebraic curve $V = f^{-1}(0)$. We notice that the delta-invariant of the singularity O denoted with $\delta(O)$ is given by the formula:

$$2\delta(O) = \mu(O) + r - 1, \tag{2.48}$$

where μ represents the Milnor number of the singularity O and it is defined depending on the number r of knot components in the link L of the singularity O in the following way:

$$\mu(O) = \begin{cases} n, \text{ if } r = 1 \\ \\ n + 1, \text{ if } r \geq 2, \end{cases} \tag{2.49}$$

with n denoting the degree of the Alexander polynomial of the singularity O.

From Equation (2.48) and Equation (2.49) it follows that the formula for the delta-invariant of the singularity O depends on the degree of the Alexander polynomial and on the number r of variables in the Alexander polynomial, which coincides with the number of knot components in the link L of the singularity O. We obtain the following expression for the delta-invariant of the singularity O :

$$2\delta(O) = \begin{cases} n + r - 1, \text{ if } r = 1 \\ \\ n + 1 + r - 1, \text{ if } r \geq 2, \end{cases} \tag{2.50}$$

and thus

$$2\delta(O) = \begin{cases} n, \text{ if } r = 1 \\ \\ n + r, \text{ if } r \geq 2. \end{cases} \tag{2.51}$$

In the following, we assume that we have computed the Alexander polynomial Δ of the link L of each singularity of a plane complex algebraic curve as described in Subsection 2.4.3. From the Alexander polynomial Δ, we derive a formula for the delta-invariant of each singularity of a plane complex algebraic curve in the following way:

Definition 53. Let $\Delta(t_1, \ldots, t_r)$ be the Alexander polynomial of the link L of the isolated singularity Q of a plane complex algebraic curve. Let r be the number of variables in Δ and let n be the degree of Δ. We denote by $\delta(Q) > 0$ the delta-invariant of the singularity Q. If $r = 1$, then the delta-invariant of Q is computed as $\delta(Q) = n/2$, otherwise $\delta(Q) = (n+r)/2$.

As an important observation we add that the delta-invariant of the singularity Q of a plane complex algebraic curve $\mathcal{C}$ intuitively measures the number of double points of $\mathcal{C}$ concentrated at Q. For instance, a singular point Q of a plane complex algebraic curve with delta-invariant $\delta(Q)$ concentrates $\delta(Q)$ double points at Q. In addition, the delta-invariant of a plane curve singularity is a local invariant of a plane complex algebraic curve.

2.4.5 Genus of a Plane Complex Algebraic Curve

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. Thus another goal of this thesis is to compute the genus of plane complex algebraic curves.

For algebraic curves with only ordinary singularities we have a method for computing the genus, based on the multiplicities of the ordinary singularities. We recall that the ordinary singularities and the multiplicities of ordinary singularities were introduced in details in Subsection 2.2.1. We mention the following definition for computing the genus of plane complex algebraic curves, assuming that the curve has only ordinary singularities:

Definition 54. ([Sendra et al., 2008]) Let $\mathcal{C}$ be a plane complex algebraic curve given by its defining polynomial $f(z, w)$ of degree m. We denote by $OrdSing(\mathcal{C})$ the set of ordinary singular points of the curve $\mathcal{C}$. For a point $Q \in OrdSing(C)$ we denoted by m_Q the multiplicity of C at the point Q. Then the genus of the plane complex algebraic curve $\mathcal{C}$, denoted with $genus(C)$, is computed using the following formula:

$$genus(C) = \frac{1}{2}\left((m-1)(m-2) - \sum_{Q \in OrdSing(C)} m_Q(m_Q - 1)\right),$$

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

If the singularities of the plane complex algebraic curves are nonordinary, then the main idea is to construct a birational map and to transform the given curve into another birational curve with only ordinary singularities. We recall that a birational map is a rational map that admits a rational inverse, whereas a rational map is any map that can be written as the ratio of two polynomial functions. An important result says that if two curves are birational, then they have the same genus. So if we can transform the given curve to a birational curve with only ordinary singular points, then we can compute the genus of the original curve by computing the genus of the other one. We will not focus on the details of this method for computing the genus of plane complex algebraic curves, as this method is not among the purposes of this thesis. We advise the reader to consult [Brieskorn and Knorrer, 1986], [Fulton, 1989], [Sendra et al., 2008], and [Walker, 1978] for more information on this method.

We concentrate our attention on another formula for computing the genus of plane complex algebraic curves, a formula which does not depend on the type of singularities of the plane complex algebraic curves. Thus we derive a formula for the genus of a plane complex algebraic curve as described in [Milnor, 1968] based on the delta-invariants of all the singularities of the curve:

Definition 55. Let $\tilde{\mathcal{C}}$ be a plane complex algebraic curve in the projective plane defined by the squarefree homogeneous polynomial $f(z,w,u) \in \mathbb{C}[z,w,u]$ of degree m, as introduced in Subsection 2.1.2. We denote by $Sing(\tilde{\mathcal{C}})$ the singularities of $\tilde{\mathcal{C}}$ as introduced in Subsection 2.2.1, and by $\delta(Q) \in \mathbb{Z}_{>0}$ the delta-invariant of the singularity Q as defined in Subsection 2.4.4. The genus of $\tilde{\mathcal{C}}$, denoted with $genus(\tilde{\mathcal{C}}) \in \mathbb{Z}$, is defined as:

$$genus(\tilde{\mathcal{C}}) = \frac{(m-1)(m-2)}{2} - \sum_{Q \in Sing(\tilde{\mathcal{C}})} \delta(Q).$$

We mention that the link of a plane curve singularity, the Alexander polynomial and the delta-invariant of the singularity are local invariants of a plane complex algebraic curve, whereas the genus of the plane complex algebraic curve is a global topological invariant.

2.4.6 More Invariants: Milnor Number, Euler Characteristic

In this subsection we introduce other topological invariants of a plane complex algebraic curve. Based on Subsection 2.4.4 we first give a formal definition for the Milnor number of a plane curve singularity, which is a local topological invariant of a plane complex algebraic curve. We then introduce the Euler characteristic of the compact Riemann surface attached to the resolution of singularities of a plane complex algebraic curve.

We introduce the Milnor number of a plane curve singularity following [Milnor, 1968]. Informally, the Milnor number $\mu(Q)$ of the singularity $Q(a,b)$ of a plane complex algebraic curve $\mathcal{C}$ defined by the squarefree polynomial $f(z,w) \in \mathbb{C}[z,w]$ is defined as a positive integer, which measures the amount of degeneracy at the critical point $Q(a,b)$ of the complex polynomial $f(z,w)$. The Milnor number $\mu(Q)$ is defined as the multiplicity of $Q(a,b)$ as solution to the collection of the polynomial equations $\frac{\partial f}{\partial z}(z,w) = \frac{\partial f}{\partial w}(z,w) = 0$ or equivalently

$$\mu(Q) = dim\left(\frac{\mathbb{C}[z,w]}{\left\langle \frac{\partial f}{\partial z}, \frac{\partial f}{\partial w} \right\rangle}\right),$$

where we denote by $\left\langle \frac{\partial f}{\partial z}, \frac{\partial f}{\partial w} \right\rangle$ the ideal generated by the partial derivatives of $f(z,w)$ with respect to z and w denoted with $\frac{\partial f}{\partial z}$ and respectively with $\frac{\partial f}{\partial w}$. From Subsection 2.4.4 we derive a formula for the Milnor number of a plane curve singularity based on the Alexander polynomial of the singularity itself. Thus we introduce the following definition for the Milnor number of a plane curve singularity:

Definition 56. Let $\Delta(t_1, \ldots, t_r)$ be the Alexander polynomial of the link L of the isolated singularity Q of a plane complex algebraic curve. Let r be the number of variables in the Alexander polynomial Δ, which coincides with the number of knot components in the link L of Q. Let n be the degree of Δ. We denote with $\mu(Q)$ the Milnor number of the singularity Q. If $r = 1$, then $\mu(Q) = n$, otherwise $\mu(Q) = n + 1$. The Milnor number $\mu(Q)$ is always a positive integer.

We give the intuitive significance for the Milnor number of a plane curve singularity, and that is the larger the Milnor number of the singularity is, the more complicated the structure of the singular point is. In this thesis, the Milnor number is also important as it provides a formula for the delta-invariant of the singularity, which is then used to compute the genus of a plane complex algebraic curve.

We now introduce the Euler characteristic of the compact Riemann surface attached to the resolution of singularities of a plane complex algebraic curve. We consider a projective plane complex algebraic curve $\tilde{\mathcal{C}}$. We assume $\mathcal{S}$ to be the compact Riemann surface attached to the resolution of singularities of $\tilde{\mathcal{C}}$, as discussed in Subsubsection 2.3.2. For introducing the notion of Euler characteristic, we need to recall some basic notions concerning compact surfaces and their triangulations. We define a triangulation (also called a polygonal representation) of a surface (or of a plane polygon) as the division of the surface (or the plane polygon) into a set of triangles (or a set of polygonal regions) called faces, which are formed of smooth non-selfintersecting edges joined at vertices. In addition, the triangulation of a surface requires the following restrictions:

- any two faces have only one edge in common if any. Each edge belongs to the boundaries of two faces;
- two edges meet in one common vertex if any;
- at least 3 edges meet at each vertex. In the literature, there exist different formulations concerning the restrictions imposed by the triangulation of a surface. Still, all these different formulations of the restrictions of a triangulation of a surface produce the same formula for the Euler characteristic.

From the literature [Lee, 2000] we know that every surface (or a 2-dimensional real manifold) has a triangulation, which might require an infinite number of triangles. However every compact surface (as in the case of our problem) admits a triangulation with a finite number of triangles. We introduce the definition of the Euler characteristic of a compact surface as follows:

Definition 57. Let $V(T)$, $E(T)$, $F(T)$ be the number of vertices, the number of edges and respectively the number of faces of the triangulation T of a compact surface $\mathcal{S}$. We define the Euler characteristic of $\mathcal{S}$ denoted with $\chi(\mathcal{S}, T)$ as the quantity

$$\chi(\mathcal{S}, T) = V(T) - E(T) + F(T).$$

Since $\chi(\mathcal{S}, T)$ is independent on the choice of a triangulation of $\mathcal{S}$ we denote the Euler characteristic of the compact surface $\mathcal{S}$ from Definition 57 with $\chi(\mathcal{S})$. In addition, the Euler characteristic is a topological invariant, see [Lee, 2000] for details. The Euler characteristic of a compact surface $\mathcal{S}$ is related to the genus of $\mathcal{S}$ by the following proposition:

Proposition 2. *([Lee, 2000, p. 143]) The Euler characteristic of a compact surface $\mathcal{S}$ with genus g is equal to:*

(i) 2, *if $\mathcal{S}$ is homeomorphic to a sphere;*

(ii) $2 - 2g$, *if $\mathcal{S}$ is homeomorphic to a connected sum of g tori.*

Thus to compute the Euler characteristic $\chi(S)$ of the compact Riemann surface $\mathcal{S}$ attached to the resolution of singularities of a projective plane complex algebraic curve $\tilde{\mathcal{C}}$ of genus g, we use the formula $\chi(\mathcal{S}) = 2 - 2g$ from Proposition 2. The computation of the Euler characteristic shows that the topological invariants computed in the previous subsections of a plane complex algebraic curve can be used to derive other information on a plane complex algebraic curve.

2.5 Approximate Invariants of Plane Complex Algebraic Curves

In Section 2.4 we introduced several invariants for a plane complex algebraic curve $\mathcal{C}$ with an isolated singularity, i.e. the link of the isolated singularity, the Alexander polynomial attached to the link of the singularity, the Milnor number of the singularity, the delta-invariant of the singularity, the genus of the curve and the Euler characteristic of the Riemann surface attached to the resolution of singularities of the curve $\mathcal{C}$. We notice that the computation of these invariants is conditioned by the computation of the image of X_ϵ through stereographic projection, which is the *link L of the singularity* and which depends on the parameter $\epsilon \in \mathbb{R}_{>0}$.
Hence we are motivated to define the ϵ-*invariants* of a plane complex algebraic curve with an isolated singularity, which depend on a parameter $\epsilon \in \mathbb{R}_{>0}$:

Definition 58. Let $\mathcal{C}$ be a plane complex algebraic curve defined by the squarefree polynomial $p(z,w) \in \mathbb{C}[z,w]$. Let $Q(z_0,w_0) \in \mathbb{C}^2$ be an isolated singularity of $\mathcal{C}$ and let $S_\epsilon(Q) = \{(z,w) \in \mathbb{C}^2 : |z-z_0|^2 + |w-w_0|^2 = \epsilon^2\}$ be the sphere centered in Q of radius $\epsilon \in \mathbb{R}_{>0}$. We take $Y_\epsilon = \mathcal{C} \cap S_\epsilon(Q)$. We consider $\pi_{(\epsilon,N)}$ the stereographic projection of the sphere $S_\epsilon(Q)$ from its north pole N, which does not belong to $\mathcal{C}$ and which is defined as:

$$\begin{array}{c} \pi_{(\epsilon,N)} : S_\epsilon \setminus \{N\} \subset \mathbb{R}^4 \to \mathbb{R}^3 \\ (a,b,c,d) \to (x,y,z) = (\frac{a}{\epsilon-d}, \frac{b}{\epsilon-d}, \frac{c}{\epsilon-d}) \end{array} , \tag{2.52}$$

with its inverse given by:

$$\begin{array}{c} \pi^{-1}_{(\epsilon,N)} : \mathbb{R}^3 \to S_\epsilon \setminus \{N\} \subset \mathbb{R}^4 \\ (x,y,z) \mapsto (a,b,c,d) = (\frac{2x\epsilon}{\kappa}, \frac{2y\epsilon}{\kappa}, \frac{2z\epsilon}{\kappa}, \frac{-\epsilon+x^2\epsilon+y^2\epsilon+z^2\epsilon}{\kappa}), \end{array} \tag{2.53}$$

where $\kappa = 1 + x^2 + y^2 + z^2$.
If $\pi_{(\epsilon,N)}(Y_\epsilon)$ has no singularities, then:

- we call $L_\epsilon(Q) := \pi_{(\epsilon,N)}(Y_\epsilon)$ the ϵ-*link of the singularity* of $p(z,w)$ (or of $\mathcal{C}$) at Q. We call L_ϵ an ϵ-algebraic link.
- we define the ϵ-*Alexander polynomial* of $\mathcal{C}$ at Q as the Alexander polynomial of L_ϵ. We denote the ϵ-Alexander polynomial with $\Delta_\epsilon(Q)$.
- we define the ϵ-*Milnor number* of Q as the Milnor number of the ϵ-Alexander polynomial of $\mathcal{C}$ at Q. If we assume that the ϵ-Alexander polynomial denoted Δ_ϵ has degree n, then we derive the following formula for the ϵ-Milnor number of Q depending on the number of variables r in Δ_ϵ :
  - if $r = 1$, then $\mu_\epsilon(Q) = n$,
  - otherwise $\mu_\epsilon(Q) = n + 1$.
- we define the ϵ-*delta-invariant* of Q as the delta-invariant of the ϵ-Alexander polynomial of $\mathcal{C}$ at Q. If we assume that the ϵ-Alexander polynomial denoted $\Delta_\epsilon(Q)$ has degree n and r variables, then we derive the following formula for the ϵ-delta-invariant of Q depending on the number of variables r in $\Delta_\epsilon(Q)$:
  - if $r = 1$, then $\delta_\epsilon(Q) = n/2$,
  - otherwise $\delta_\epsilon(Q) = (n+r)/2$.

- the ϵ-genus of the plane complex algebraic curve $\mathcal{C}$ is computed based on the ϵ-delta-invariants of all the singularities of $\mathcal{C}$ in the following way: ϵ-genus:

$$genus_\epsilon(\mathcal{C}) = \frac{(m-1)(m-2)}{2} - \sum_{Q \in Sing(\mathcal{C})} \delta_\epsilon(Q),$$

  where we denote with $Sing(\mathcal{C})$ the set of singularities of the plane complex algebraic curve $\mathcal{C}$.

- the ϵ-Euler characteristic χ_ϵ of the compact Riemann surface attached to the resolution of singularities of $\mathcal{C}$ is computed using the following formula $\chi_\epsilon = 2 - 2genus_\epsilon(\mathcal{C})$.

- moreover, we define the ϵ-*local topological type* of Q as the pair $\big(L_\epsilon(Q), \Delta_\epsilon(Q), \delta_\epsilon(Q)\big)$, where $L_\epsilon(Q)$ denotes the ϵ-link of the singularity Q, where $\Delta_\epsilon(Q)$ denotes the ϵ-Alexander polynomial of $L_\epsilon(Q)$, and $\delta_\epsilon(Q)$ denotes the ϵ-delta-invariant of Q.

Thus in our study computing the approximate invariants of a plane complex algebraic curve and its singularities means computing the invariants of the plane complex algebraic curve and its singularities depending on an input parameter $\epsilon \in \mathbb{R}_{>0}$. More specifically, for a plane complex algebraic curve $\mathcal{C}$ with a singularity Q, the approximate invariants of the singularity Q are determined by the intersection of the curve $\mathcal{C}$ with a sphere centered in Q and of a radius ϵ as specified in Definition 58. If the intersection $\mathcal{C} \cap S_\epsilon$ has no singularities, then we compute the ϵ-algebraic link and its ϵ-Alexander polynomial. The ϵ-Alexander polynomial can only change if the intersection $\mathcal{C} \cap S_\epsilon$ has singularities, otherwise we have an isotopy which leaves the ϵ-Alexander polynomial fixed. From the ϵ-Alexander polynomial we derive formulas for other approximate topological invariants of plane complex algebraic curves, including a formula for the ϵ-genus of the curve. In Chapter 3 we give straightforward algorithms to compute all the approximate invariants of a plane complex algebraic curve introduced in Definition 58. The algorithms depend on an input parameter ϵ, which represents the radius of a small sphere that we intersect with the given plane complex algebraic curve.

Chapter 3

Symbolic-Numeric Algorithms for Plane Complex Algebraic Curves

In this chapter, we state the main problem that we solve, i.e. the algebraic problem of computing approximate topological invariants (i.e. approximate delta-invariant, approximate genus) of a plane complex algebraic curve defined by a squarefree polynomial with exactly and inexactly-known coefficients. Moreover, based on Chapter 2, we design and we present straightforward symbolic-numeric algorithms for computing approximate invariants of a plane complex algebraic curve.

Problem 1. *Given the following:*

(i) a squarefree complex bivariate polynomial $p(z,w) \in \mathbb{C}[z,w]$ with exact and inexact coefficients that defines a plane complex algebraic curve $\mathcal{C} \subset \mathbb{C}^2$, i.e. $\mathcal{C} = \{(z,w) \in \mathbb{C}^2 \mid p(z,w) = 0\} \subset \mathbb{C}^2$. For the polynomial $p(z,w)$ we associate a positive real number $\delta \in \mathbb{R}_{>0}$, which measures the error level (also called the tolerance level or the noise level) in the coefficients of $p(z,w)$.

(ii) a parameter $\epsilon \in \mathbb{R}_{>0}$ that determines the sphere S_ϵ centered in the origin $O(0,0) \in \mathbb{C}^2$ and of radius ϵ denoted with $S_\epsilon = \{(z,w) \in \mathbb{C}^2 \mid |z|^2 + |w|^2 = \epsilon^2\}$.

our goal is: to compute a set of ϵ-invariants (also called approximate invariants) of $\mathcal{C}$ and its singularities as introduced in Chapter 2, Section 2.5, Definition 58. We compute: (1) the ϵ-algebraic link, ϵ-Alexander polynomial, the ϵ-Milnor number, the ϵ-delta-invariant of each singularity Q of $\mathcal{C}$, (2) the ϵ-genus of $\mathcal{C}$, (3) the ϵ-Euler characteristic of the Riemann surface attached to the resolution of singularities of $\mathcal{C}$, (4) the ϵ-topological type of each singularity Q of $\mathcal{C}$.

3.1 Algorithm for Computing the Approximate Singularities

3.1.1 Description of the Algorithm

In this subsection, we describe an algorithm for computing the set of singularities $Sing(\mathcal{C})$ of the plane complex algebraic curve $\mathcal{C}$ of degree m defined by a squarefree polynomial $p(z,w) \in$

$\mathbb{C}[z,w]$ with both exact and inexact coefficients. We assume that for the polynomial $p(z,w)$ we are also provided with a positive real number $\delta \in \mathbb{R}_{>0}$, which measures the error level in the coefficients of $p(z,w)$. Consequently, we need effective methods for computing the singularities of a plane complex algebraic curve defined by a polynomial with both exact and inexact data. For this purpose, we define a numerical singularity of $\mathcal{C}$ as a point $Q(z_0,w_0) \in \mathbb{C}^2$ such that $p(Q)$, $\frac{\partial p}{\partial z}(Q)$ and $\frac{\partial p}{\partial w}(Q)$ are small compared to the coefficients of the polynomial $p(z,w)$.

From Chapter 2, Section 2.5, Definition 6, it follows that to compute the set of singularities, we need to determine the following set of points:

$$Sing(\mathcal{C}) = \{(z_0,w_0) \in \mathbb{C}^2 \mid p(z_0,w_0) = \partial_z p(z_0,w_0) = \partial_w p(z_0,w_0) = 0\}.$$

We thus need to solve the following overdeterminate system of 3 polynomial equations in 2 unknowns z_0 and w_0 in $\mathbb{C}^2$:

$$p(z_0,w_0) = \partial_z p(z_0,w_0) = \partial_w p(z_0,w_0) = 0. \tag{3.1}$$

We first compute the roots of System (3.1) in $\mathbb{R}^2$ with subdivision methods introduced in [Mourrain and Pavone, 2009] and implemented in the free system Axel developed by [Wintz et al., 2006] and in the free computer algebra system Mathemagix developed by [van der Hoeven et al., 2002]. The advantage of these methods is that they numerically compute the roots of System (3.1), and that they can be used for both exact and inexact coefficients of the defining polynomial $p(z,w)$ of the curve $\mathcal{C}$. These subdivision methods take as input the polynomials defining the System (3.1), a box $B = [-a,a] \times [-b,b] \subset \mathbb{R}^2$, and a positive real number $\delta \in \mathbb{R}_{>0}$, which measures the error level in the coefficients of $p(z,w)$. The box B has to be big enough to contain all the roots of System (3.1). The output of the subdivision methods is a set of boxes S in $B \subset \mathbb{R}^2$ smaller than δ, which contains all the roots of System (3.1), and a set M containing the middle points of all the boxes from S.

We compute the real singularities of the plane algebraic curve defined by the squarefree polynomial $p(z,w) \in \mathbb{C}[z,w]$ with exact and inexact coefficients in the projective real plane $\mathbb{P}^2(\mathbb{R})$ by homogenizing and dehomogenizing the polynomial $p(z,w)$ with respect to different variables and making sure not to return solutions in the overlaps twice. We consider the projective plane over the real numbers $\mathbb{P}^2(\mathbb{R}) = U_z \cup U_w \cup U_u$, where U_z, U_w, U_u are homeomorphic to $\mathbb{R}^2$ and $U_z = \{(1 : \frac{w}{z} : \frac{u}{z})\}, U_w = \{(\frac{z}{w} : 1 : \frac{u}{w})\}, U_u = \{(\frac{z}{u} : \frac{w}{u} : 1)\}$ represent the dehomogenizations with respect to the variables z, w and u. Without loss of generality we assume that $|u| \geq |z|, |w|$. We notice that any point from $\mathbb{P}^2(\mathbb{R})$ is in one of the boxes B_z, B_w, B_u, where $B_z \subseteq U_z, B_w \subseteq U_w, B_u \subseteq U_u$ and $B_z = B_w = B_u = [-1,1] \times [-1,1]$. Thus by using subdivision methods, we compute a list of δ-boxes S in $B = [-1,1] \times [-1,1] \subset \mathbb{R}^2$ smaller than a given tolerance δ, and a list M of middle points for all the boxes in S with two properties: (1) each real singularity of the plane complex algebraic curve is contained in one of the δ-boxes from S, (2) and the value of p and its first derivatives in each point from M are small. It follows that by using the subdivision methods we compute a list of δ-boxes in B smaller than δ, which contain all the singularities of $\mathcal{C}$. Still, the existence and the uniqueness of a singularity in each δ-box is not guaranteed [Mourrain et al., 2008], but this is not required for our purpose.

In the same way, we can use subdivision methods to find the complex singularities of the plane complex algebraic curve $\mathcal{C}$ in the projective complex plane $\mathbb{P}^2(\mathbb{C})$. We consider the complex variables $z = z_1 + iz_2, w = w_1 + iw_2, u = u_1 + iu_2$ and the projective plane over the complex numbers $\mathbb{P}^2(\mathbb{C}) = U_z \cup U_w \cup U_u$, where U_z, U_w, U_u are homeomorphic to $\mathbb{C}^2$ and $U_z = \{(1 : \frac{w}{z} : \frac{u}{z})\}, U_w = \{(\frac{z}{w} : 1 : \frac{u}{w})\}, U_u = \{(\frac{z}{u} : \frac{w}{u} : 1)\}$ represent the dehomogenizations

with respect to the variables z, w and u. We assume $|z_1| \geq |z_2|, |w_1|, |w_2|, |u_1|, |u_2|$ and we show that any point from $\mathbb{P}^2(\mathbb{C})$ is in one of the boxes B_z, B_w, B_u, where $B_z \subseteq U_z$ and $B_z = \{(w,u) \in \mathbb{C}^2 | w_1, w_2, u_1, u_2 \in [-1,1]\}$ (we obtain equivalent formula for B_w, B_u). For instance, we consider $U_z = \{(1 : \frac{w}{z} : \frac{u}{z})\}$ and $Re(\frac{w}{z})$ the real part of the complex number $\frac{w}{z}$. We rewrite

$$Re(\frac{w}{z}) = Re\left(\frac{w_1 + iw_2}{z_1 + iz_2}\right) = Re\left(\frac{(z_1 - iz_2)(w_1 + iw_2)}{z_1^2 + z_2^2}\right) = \frac{z_1w_1 + z_2w_2}{z_1^2 + z_2^2}.$$

We get:

$$|Re(\frac{w}{z})| = \left|\frac{z_1w_1 + z_2w_2}{z_1^2 + z_2^2}\right| \leq \frac{|z_1||z_1| + |z_1||z_1|}{2|z_1|^2} = \frac{2|z_1|^2}{2|z_1|^2} = 1.$$

In our implementation, we apply the subdivision methods for the real case. As discussed before, we can apply the subdivision methods to the complex case, but this is not available yet in our current implementation.

We describe the algorithm `ApproxRealSing`$(\mathcal{C}, p, \delta)$ for computing the approximate real singularities (also called the numerical real singularities) of the plane complex algebraic curve $\mathcal{C}$ defined by the squarefree complex polynomial $p(z,w)$ with exact and inexact coefficients. For the inexact coefficients of $p(z,w)$ we are given a positive real number $\delta \in \mathbb{R}_{>0}$, which determines the error level. We recall that as input parameter to the subdivision methods we need a subset $B \subset \mathbb{R}^2$ called a box.

Algorithm 1 Approximate real singularities of the plane complex algebraic curve $\mathcal{C}$ defined by $p(z,w)$ of degree m and for which the tolerance in its coefficients is measured by δ. The algorithm takes as input also the box $B = [-a,a] \times [-b,b]$: `ApproxRealSing`$(\mathcal{C}, p, \delta, B)$

Input: $p(z,w) \in \mathbb{C}[z,w]$ a squarefree polynomial, m the degree of $p(z,w)$,
$\mathcal{C} = \{(z,w) \in \mathbb{C}^2 \mid p(z,w) = 0)\}$ a plane complex algebraic curve of degree m,
$\delta \in \mathbb{R}_{>0}$ a positive real number,
$B = [-a,a] \times [-b,b] \subset \mathbb{R}^2$ a subset of $\mathbb{R}^2$ called a box.
Output: a list of points $M \subset B$ such that for every $Q \in \mathit{Sing}(\mathcal{C})$ there exists a unique $R \in M$ such that $d(Q,R) \leq \delta$,
where $\mathit{Sing}(\mathcal{C})$ is the set of real singularities of $\mathcal{C}$ in the projective real plane.

1: Homogenize $p(z,w)$ w.r.t. u to get $p(z,w,u)$.

(a) Dehomogenize $p_1(z,w) := p(z,w,1)$.
(b) Get S_1 by solving $p_1 = \partial_z p_1 = \partial_w p_1 = 0$ with subdivision methods.
(c) Homogenize $S_1 = \{(z_0, w_0) \in \mathbb{R}^2\}$ to get $S_1' = \{(z_0 : w_0 : 1) \in \mathbb{P}^2(\mathbb{R})\}$.
(d) Dehomogenize $p_2(w,u) := p(1,w,u)$.
(e) Get S_2 by solving $p_2 = \partial_w p_2 = \partial_u p_2 = u = 0$ with subdivision methods.
(f) Homogenize $S_2 = \{(w_0, u_0) \in \mathbb{R}^2\}$ to get $S_2' = \{(1 : w_0 : u_0) \in \mathbb{P}^2(\mathbb{R})\}$.
(g) Dehomogenize $p_3(z,u) := p(z,1,u)$.
(h) Get S_3 by solving $p_3 = \partial_z p_3 = \partial_u p_3 = z = u = 0$ with subdivision methods.
(i) Homogenize $S_3 = \{(z_0, u_0) \in \mathbb{R}^2\}$ to get $S_3' = \{(z_0 : 1 : u_0) \in \mathbb{P}^2(\mathbb{R})\}$.

2: Return $\mathit{Sing}(\mathcal{C}) = S_1' \cup S_2' \cup S_3'$.

In the rest of this thesis we call the list M returned by the algorithm `ApproxRealSing` the list of numerical singularities of the plane complex algebraic curve $\mathcal{C}$. We sometimes denote the list M with $NumSing(\mathcal{C})$. In Example 20 we compute the set of numerical real singularities $Sing(\mathcal{C})$ of a plane complex algebraic curve $\mathcal{C}$ defined by a squarefree polynomial $p(z,w) \in \mathbb{C}[z,w]$ with exact coefficients using the **Algorithm** 1-`ApproxRealSing`$(\mathcal{C}, p, \delta)$.

Example 20. We consider $\mathcal{C}$ the plane complex algebraic curve defined by the squarefree polynomial $p(z,w) = z^2w + w^4 \in \mathbb{C}[x,y]$ of degree 4. We mention that the values of the box $B = [-a,a] \times [-b,b] \subset \mathbb{R}^2$ are set directly in the subdivision methods and we do not have to introduce them explicitly as input to the algorithm. We compute the set of singularities $Sing(\mathcal{C})$ of the plane complex algebraic curve $\mathcal{C}$ in the following steps:

1. We homogenize $p(z,w)$ w.r.t. the variable u to obtain $p(z,w,u) = z^2wu + w^4$.
   (a) We replace $u = 1$ in $p(z,w,u)$ and we obtain $p_1(z,w) = z^2w + w^4$.
   (b) We solve the overdeterminate system $z_0^2w_0 + w_0^4 = 2z_0w_0 = z_0^2 + 4w_0^3 = 0$ and we get $S_1 = \{(0,0)\}$.
   (c) We homogenize S_1 and we get $S_1^{'} = \{(0:0:1)\}$.
   (d) We replace $z = 1$ in $p(z,w,u)$ and we obtain $p_2(w,u) = wu + w^4$.
   (e) Similarly as for the System (1b), we solve the overdeterminate system $w_0u_0 + w_0^4 = u_0 + 4w_0^3 = w_0 = u_0 = 0$ and we obtain $S_2 = \{(0,0)\}$.
   (f) We homogenize S_2 and we get $S_2^{'} = \{(1:0:0)\}$.
   (g) We replace $w = 1$ in $p(z,w,u)$ and we obtain $p_3(z,u) = z^2u + 1$. We notice that $S_3 = \emptyset$.
2. We return $Sing(\mathcal{C}) = S_1^{'} \cup S_2^{'} \cup S_3^{'} = \{(0:0:1), (1:0:0)\}$.

3.1.2 Applications of the Algorithm

We use the **Algorithm** 1-`ApproxRealSing` to compute the set of approximate real singularities of a plane complex algebraic curve defined by a squarefree polynomial with both exact and inexact coefficients. We mention that there exists several symbolic methods that can be used to compute the singularities of a plane complex algebraic curve defined by a squarefree polynomial with exact coefficients. Such methods include the Gröbner bases method [Buchberger and Winkler, 1998], or the resultants method [Lang, 2002]. However these methods are not usable for problems with inexact data, as in the case of our problem.

3.2 Algorithm for Computing the Approximate Link of a Singularity

3.2.1 Description of the Algorithm

In this subsection, we construct an algorithm for computing the ϵ-link (or the approximate link) L_ϵ of an isolated singularity of a plane complex algebraic curve using the notions defined in Chapter 2, Subsection 2.3.3 and Subsection 2.4.2. We consider a plane complex algebraic curve $\mathcal{C} = \{(z,w) \in \mathbb{C}^2 \mid p(z,w) = 0\}$ defined by the squarefree polynomial $p(z,w) \in \mathbb{C}[z,w]$ with an isolated singularity in the point $Q(z_0,w_0) \subset \mathbb{C}^2$. For our purpose, we firstly translate the singularity $Q(z_0,w_0)$ of the plane complex algebraic curve $\mathcal{C}$ in the origin $O(0,0) \in \mathbb{C}^2$ by making an affine change of coordinates, i.e. $\mathcal{C} = \{(z,w) \in \mathbb{C}^2 \mid p(z+$

$z_0, w+w_0) = 0\}$. Hence, we obtain that the plane complex algebraic curve $\mathcal{C}$ has a singularity in the origin $O(0,0) \in \mathbb{C}^2$. We secondly substitute the complex variables $z = a+ib, w = c+id$ in the defining polynomial of $\mathcal{C}$ and we obtain $p(a,b,c,d) = R(a,b,c,d) + iI(a,b,c,d)$, with $R(a,b,c,d), I(a,b,c,d) \in \mathbb{R}[a,b,c,d]$. The equations $R(a,b,c,d) = I(a,b,c,d) = 0$ define a surface in $\mathbb{R}^4$, which we denote with $\mathcal{S}$:

$$\mathcal{S} = \{(a,b,c,d) \in \mathbb{R}^4 \mid R(a,b,c,d) = I(a,b,c,d) = 0\}. \tag{3.2}$$

We next take the sphere centered in the origin $(0,0,0,0) \in \mathbb{R}^4$ and of a small radius $\epsilon \in \mathbb{R}_{>0}$ denoted with:

$$S_\epsilon = \{(a,b,c,d) \in \mathbb{R}^4 \mid a^2 + b^2 + c^2 + d^2 = \epsilon^2\}.$$

We intersect $\mathcal{S}$ with this small sphere S_ϵ to obtain a curve $Y_\epsilon = \mathcal{S} \cap S_\epsilon$ in $\mathbb{R}^4$.
We now consider $N(0,0,0,\epsilon)$ the north pole of the sphere S_ϵ, which does not belong to $\mathcal{S}$. We consider the stereographic projection from $\mathbb{R}^4$ to $\mathbb{R}^3$, defined by the following homeomorphism:

$$\begin{gathered} \pi_{(\epsilon,N)} : S_\epsilon \setminus \{N\} \subset \mathbb{R}^4 \to \mathbb{R}^3, \\ (a,b,c,d) \mapsto (x,y,z) = (\frac{a}{\epsilon - d}, \frac{b}{\epsilon - d}, \frac{c}{\epsilon - d}). \end{gathered} \tag{3.3}$$

with its inverse $\pi_{(\epsilon,N)}$ given by:

$$\begin{gathered} \pi^{-1}_{(\epsilon,N)} : \mathbb{R}^3 \to S_\epsilon \setminus \{N\} \subset \mathbb{R}^4, \\ (x,y,z) \mapsto (a,b,c,d) = (\frac{2x\epsilon}{\kappa}, \frac{2y\epsilon}{\kappa}, \frac{2z\epsilon}{\kappa}, \frac{-\epsilon + x^2\epsilon + y^2\epsilon + z^2\epsilon}{\kappa}), \end{gathered} \tag{3.4}$$

where $\kappa = 1 + x^2 + y^2 + z^2$. We notice that $\pi_{(\epsilon,N)}$ allows us to project the set $Y_\epsilon = \mathcal{S} \cap S_\epsilon \subset S_\epsilon \setminus \{N\}$ from $\mathbb{R}^4$ into $\mathbb{R}^3$, and the inverse $\pi^{-1}_{(\epsilon,N)}$ allows us to compute the set $\pi_{(\epsilon,N)}(Y_\epsilon)$ in the following way:

$$\pi_{(\epsilon,N)}(Y_\epsilon) = \{(x,y,z) \in \mathbb{R}^3 \mid \exists (a,b,c,d) \in Y_\epsilon : \pi_{(\epsilon,N)}(a,b,c,d) = (x,y,z)\} \Leftrightarrow$$

$$\pi_{(\epsilon,N)}(Y_\epsilon) = \{(x,y,z) \in \mathbb{R}^3 \mid \exists (a,b,c,d) = \pi^{-1}_{(\epsilon,N)}(x,y,z) \in Y_\epsilon = \mathcal{S} \cap S_\epsilon\}. \tag{3.5}$$

In Equation (3.5) we replace the formula for (a,b,c,d) computed in Equation (3.4) and we obtain:

$$\pi_{(\epsilon,N)}(Y_\epsilon) = \{(x,y,z) \in \mathbb{R}^3 \mid (\frac{2x\epsilon}{\kappa}, \frac{2y\epsilon}{\kappa}, \frac{2z\epsilon}{\kappa}, \frac{-\epsilon + x^2\epsilon + y^2\epsilon + z^2\epsilon}{\kappa}) \in Y_\epsilon\}. \tag{3.6}$$

We denote $\alpha := (\frac{2x\epsilon}{\kappa}, \frac{2y\epsilon}{\kappa}, \frac{2z\epsilon}{\kappa}, \frac{-\epsilon + x^2\epsilon + y^2\epsilon + z^2\epsilon}{\kappa})$ and we rewrite Equation (3.6) as follows:

$$\pi_{(\epsilon,N)}(Y_\epsilon) = \{(x,y,z) \in \mathbb{R}^3 \mid \alpha \in \mathcal{S} \cap S_\epsilon\}.$$

Based on Equation (3.2) we get:

$$\pi_{(\epsilon,N)}(Y_\epsilon) = \{(x,y,z) \in \mathbb{R}^3 \mid R(\alpha) = I(\alpha) = 0\}.$$

We clear out the denominators in $R(\alpha) = I(\alpha) = 0$ and we obtain two equations $g_\epsilon(x,y,z) = 0$ and $h_\epsilon(x,y,z) = 0$ defining the stereographic projection of Y_ϵ :

$$\pi_{(\epsilon,N)}(Y_\epsilon) = \{(x,y,z) \in \mathbb{R}^3 \mid g_\epsilon(x,y,z) = h_\epsilon(x,y,z) = 0\},$$

where $g_\epsilon(x,y,z), h_\epsilon(x,y,z) \in \mathbb{R}[x,y,z]$. Based on Chapter 2, Definition 58, if $\pi_{(\epsilon,N)}(Y_\epsilon)$ has no singularities (i.e. it is a smooth and closed implicit algebraic curve in $\mathbb{R}^3$ given as the intersection of two implicit algebraic surfaces with the defining polynomials $g_\epsilon(x,y,z)$ and respectively $h_\epsilon(x,y,z)$), then we call $L_\epsilon := \pi_{(\epsilon,N)}(Y)$ the ϵ-link (also called the approximate link) of the singularity Q of the curve $\mathcal{C}$. In addition, from Chapter 2, Theorem 10 of Milnor we know that for sufficiently small ϵ, L_ϵ is equal to the link of the singularity Q, which we denote with L. In fact, L is a smooth and closed implicit algebraic curve in $\mathbb{R}^3$ given as the intersection of two implicit algebraic surfaces S_1, S_2 in $\mathbb{R}^3$ with defining polynomials $g_\epsilon(x,y,z)$, $h_\epsilon(x,y,z) \in \mathbb{R}[x,y,z]$. We add that the two surfaces S_1, S_2, which define as their intersection the link L, are part of the Milnor fibration as discussed in Subsection 2.4.2.

Remark 10. We make an important observation concerning the existence of vertical and of horizontal tangents of the plane complex algebraic curve $\mathcal{C}$ at the north pole $N(0,\epsilon)$ of the sphere S_ϵ, north pole that is used for defining the stereographic projection $\pi_{(\epsilon,N)}$ from Equation (3.3). To ensure the correct stereographic projection of the intersection $\mathcal{S} \cap S_\epsilon$, where $\mathcal{S}$ denotes the 2-dimensional object in $\mathbb{R}^4$ defined by $\mathcal{C}$, and S_ϵ denotes the sphere of radius ϵ centered in the origin, we have to eliminate the vertical and the horizontal tangents of $\mathcal{C}$ at N, in case these vertical and horizontal tangents do exist. One solution of eliminating these tangents would be to choose another coordinates for the north pole N instead of the chosen coordinates $(0,\epsilon)$. In this case, we would have to adapt also the formula for defining the stereographic projection from Equation (3.3). Alternatively, we choose to rotate the plane complex algebraic curve before we project the intersection $\mathcal{S} \cap S_\epsilon$ such that we eliminate the vertical and the horizontal tangents at $N(0,\epsilon)$. We distinguish the following cases for rotating the plane complex algebraic curve $\mathcal{C}$:

- We consider the lowest degree part $l(x,y)$ of the defining polynomial $p(z,w) \in \mathbb{C}[z,w]$ of $\mathcal{C}$. If the variable z factors out in $l(z,w)$ (i.e. the curve $\mathcal{C}$ has a vertical tangent at $N(0,\epsilon)$), then we make the substitution $\{z \to -w, w \to z\}$ in $p(z,w)$ (i.e. we rotate the curve $\mathcal{C}$ counterclockwise by $90°$). We consider the rotation matrix $\begin{pmatrix} \cos\theta & -sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$ with $\theta = 90°$, and we obtain the substitution:
$$\begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} z \\ w \end{pmatrix} \to \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} z \\ w \end{pmatrix} \to \begin{pmatrix} -w \\ z \end{pmatrix}.$$
- We consider the lowest degree part $l(z,w)$ of the defining polynomial $p(z,w) \in \mathbb{C}[z,w]$ of $\mathcal{C}$. If the variable w factors out in $l(z,w)$ (i.e. the curve has an horizontal tangent at $N(0,\epsilon)$), then we make the substitution $\{z \to -w, w \to z\}$ in $p(z,w)$ (i.e. we rotate the curve $\mathcal{C}$ counterclockwise by $90°$). The substitution is basically obtained as in the previous case.
- We consider the lowest degree part $l(z,w)$ of the defining polynomial $p(z,w) \in \mathbb{C}[z,w]$ of $\mathcal{C}$. If both variables z and w factor out in $l(z,w)$ (i.e. the curve $\mathcal{C}$ has both a vertical and an horizontal tangent at $N(0,\epsilon)$), then we make the substitution $\{z \to z-w, w \to z+w\}$ in $p(z,w)$ (i.e. we rotate the curve $\mathcal{C}$ counterclockwise by $45°$). We consider the rotation matrix $\begin{pmatrix} \cos\theta & -sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$ with $\theta = 45°$, and we obtain the substitution:
$\begin{pmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{pmatrix} \begin{pmatrix} z \\ w \end{pmatrix} \to \begin{pmatrix} z-w \\ z+w \end{pmatrix} \frac{1}{\sqrt{2}}$, where we can omit the $\frac{1}{\sqrt{2}}$ term.

We describe the algorithm `ApproxLink`$(\epsilon, Q, \mathcal{C}, p)$ for computing the ϵ-algebraic link L_ϵ of the singularity Q of the plane complex algebraic curve $\mathcal{C}$ defined by the squarefree polynomial $p(z,w) \in \mathbb{C}[z,w]$ with exact and inexact coefficients. The parameter ϵ denotes the radius of the sphere $S_\epsilon \subset \mathbb{C}^2$ that we intersect with the zero set of $p(z,w)$, as described in Chapter 2, Section 2.5, Definition 58.

Algorithm 2 ϵ-link of the singularity Q of the plane algebraic curve $\mathcal{C}$ defined by $p(z,w)$: `ApproxLink`$(\epsilon, Q, \mathcal{C}, p)$

Input: $p(z,w) \in \mathbb{C}[z,w]$ a squarefree complex polynomial, m the degree of $p(z,w)$,
$\mathcal{C} = \{(z,w) \in \mathbb{C}^2 \mid p(z,w) = 0\}$ a plane complex algebraic curve of degree m,
$Q(z_0, w_0)$ a numerical singularity of $\mathcal{C}$,
$\epsilon \in \mathbb{R}_{>0}$ a positive real number, which defines the sphere $S_\epsilon(O)$ of radius ϵ centered in O.
Output: $G,\ H \in \mathbb{R}[x,y,z]$,
where the common zero set of G, H equals L_ϵ.

1: Translate the singularity $Q(z_0, w_0)$ in the origin $O(0,0)$ by substituting $z \leftarrow z + z_0, w \leftarrow w + w_0$ in $p(z,w)$. In $p(z,w)$ set the terms of degree 0 and of degree 1 to zero.

2: Consider $l(z,w)$ the lowest degree part of $p(z,w)$.

3: If z or w factors out in $l(z,w)$, substitute $\{z \to -w, w \to z\}$ in $p(z,w)$.

4: If z and w factors out in $l(z,w)$, substitute $\{z \to z - w, w \to z + w\}$ in $p(z,w)$.

5: Substitute $z \leftarrow a + ib, w \leftarrow c + id$ in $p(z,w)$ and obtain

$$p(a,b,c,d) = R(a,b,c,d) + iI(a,b,c,d), \text{ with } R, I \in \mathbb{R}[a,b,c,d].$$

6: Extract $R(a,b,c,d) = I(a,b,c,d) = 0$, which defines

$$\mathcal{S} = \{(a,b,c,d) \in \mathbb{R}^4 \mid R(a,b,c,d) = I(a,b,c,d) = 0\}.$$

7: Define $\alpha =: (\frac{2x\epsilon}{\kappa}, \frac{2y\epsilon}{\kappa}, \frac{2z\epsilon}{\kappa}, \frac{-\epsilon + x^2\epsilon + y^2\epsilon + z^2\epsilon}{\kappa})$, where $\kappa = 1 + x^2 + y^2 + z^2$.

8: Substitute $(a,b,c,d) \leftarrow \alpha$ in $\mathcal{S}$ to get $R(\alpha) = I(\alpha) = 0$.

9: Eliminate the denominators in $R(\alpha) = I(\alpha) = 0$ to get $g_\epsilon(x,y,z) = h_\epsilon(x,y,z) = 0$, with $g_\epsilon, h_\epsilon \in \mathbb{R}[x,y,z]$. For $Y_\epsilon = \mathcal{S} \cap S_\epsilon(O)$ these equations define:

$$\pi_{(\epsilon,N)}(Y_\epsilon) = \{(x,y,z) \in \mathbb{R}^3 : g_\epsilon(x,y,z) = h_\epsilon(x,y,z) = 0\},$$

where $\pi_{(\epsilon,N)}$ denotes the stereographic projection defined by Equation (3.3).

10: If $\pi_{(\epsilon,N)}(Y_\epsilon)$ has no singularities, then

- return $G =: g_\epsilon(x,y,z)$ and $H =: h_\epsilon(x,y,z)$.
- else return "`failure`".

Example 21. We consider $\mathcal{C}$ the plane complex algebraic curve defined by the squarefree polynomial $p(z,w) = z^2w + w^4 \in \mathbb{C}[x,y]$, i.e.

$$\mathcal{C} = \{(z,w) \in \mathbb{C}^2 \mid z^2w + w^4 = 0\}.$$

In Example 20, we computed the set of real singularities of $\mathcal{C}$ defined by the homogeneous polynomial $p(z,w,u) = z^2wu + w^4$ in the projective real plane, and we obtained $Sing(\mathcal{C}) = \{(0:0:1),(1:0:0)\}$ with the **Algorithm** 1-`ApproxRealSing`. We now want to compute the ϵ-links of both singularities from $Sing(\mathcal{C})$. We notice that the affine parts of the considered projective algebraic curve have the same singularity $(0,0)$. We thus need to compute the ϵ-link of the singularity $(0,0)$ two times obtaining the same duplicated result. We now compute the ϵ-link of the singularity $Q(0,0)$ using the **Algorithm** 2−`ApproxLink`$(\epsilon, Q, \mathcal{C}, p)$. We consider the input parameter to be $\epsilon = 0.5$.

1. We substitute $z \leftarrow a + ib, w \leftarrow c + id$ and we obtain

$$p(a,b,c,d) = a^2c - b^2c + c^4 - 2abd - 6c^2d^2 + d^4 + \\ +i(2abc + a^2d - b^2d + 4c^3d - 4cd^3).$$

2. We extract the real and the imaginary part from the equation $p(a,b,c,d) = 0$ and we obtain the following two equations:

$$a^2c - b^2c + c^4 - 2abd - 6c^2d^2 + d^4 = 2abc + a^2d - b^2d + 4c^3d - 4cd^3 = 0. \quad (3.7)$$

3. We define $\alpha =: (\frac{2x\epsilon}{\kappa}, \frac{2y\epsilon}{\kappa}, \frac{2z\epsilon}{\kappa}, \frac{-\epsilon+x^2\epsilon+y^2\epsilon+z^2\epsilon}{\kappa})$, where $\kappa = 1 + x^2 + y^2 + z^2$.

4. We substitute $(a,b,c,d) \leftarrow \alpha$ in the Equations (3.7).

5. We clear out the denominators in the two previous equations and we obtain the following two polynomials:

$$g_\epsilon(x,y,z) = 8xy\epsilon^3 - 8x^5y\epsilon^3 - 16x^3y^3\epsilon^3 - 8xy^5\epsilon^3 + 8x^2z\epsilon^3 + 8x^4z\epsilon^3 - 8y^2z\epsilon^3 - \\ 8y^4z\epsilon^3 - 16x^3yz^2\epsilon^3 - 16xy^3z^2\epsilon^3 + 8x^2z^3\epsilon^3 - 8y^2z^3\epsilon^3 - 8xyz^4\epsilon^3 + \epsilon^4 - 4x^2\epsilon^4 + \\ 6x^4\epsilon^4 - 4x^6\epsilon^4 + x^8\epsilon^4 - 4y^2\epsilon^4 + 12x^2y^2\epsilon^4 - 12x^4y^2\epsilon^4 + 4x^6y^2\epsilon^4 + 6y^4\epsilon^4 - \\ 12x^2y^4\epsilon^4 + 6x^4y^4\epsilon^4 - 4y^6\epsilon^4 + 4x^2y^6\epsilon^4 + y^8\epsilon^4 - 28z^2\epsilon^4 + 60x^2z^2\epsilon^4 - \\ 36x^4z^2\epsilon^4 + 4x^6z^2\epsilon^4 + 60y^2z^2\epsilon^4 - 72x^2y^2z^2\epsilon^4 + 12x^4y^2z^2\epsilon^4 - \\ 36y^4z^2\epsilon^4 + 12x^2y^4z^2\epsilon^4 + 4y^6z^2\epsilon^4 + 70z^4\epsilon^4 - 60x^2z^4\epsilon^4 + \\ 6x^4z^4\epsilon^4 - 60y^2z^4\epsilon^4 + 12x^2y^2z^4\epsilon^4 + 6y^4z^4\epsilon^4 - \\ 28z^6\epsilon^4 + 4x^2z^6\epsilon^4 + 4y^2z^6\epsilon^4 + z^8\epsilon^4 \text{ and}$$

$$h_\epsilon(x,y,z) = -4x^2\epsilon^3 + 4x^6\epsilon^3 + 4y^2\epsilon^3 + 4x^4y^2\epsilon^3 - 4x^2y^4\epsilon^3 - 4y^6\epsilon^3 + \\ 16xyz\epsilon^3 + 16x^3yz\epsilon^3 + 16xy^3z\epsilon^3 + 8x^4z^2\epsilon^3 - 8y^4z^2\epsilon^3 + 16xyz^3\epsilon^3 + \\ 4x^2z^4\epsilon^3 - 4y^2z^4\epsilon^3 + 8z\epsilon^4 - 24x^2z\epsilon^4 + 24x^4z\epsilon^4 - 8x^6z\epsilon^4 - \\ 24y^2z\epsilon^4 + 48x^2y^2z\epsilon^4 - 24x^4y^2z\epsilon^4 + 24y^4z\epsilon^4 - 24x^2y^4z\epsilon^4 - \\ 8y^6z\epsilon^4 - 56z^3\epsilon^4 + 80x^2z^3\epsilon^4 - 24x^4z^3\epsilon^4 + 80y^2z^3\epsilon^4 - \\ 48x^2y^2z^3\epsilon^4 - 24y^4z^3\epsilon^4 + 56z^5\epsilon^4 - 24x^2z^5\epsilon^4 - \\ 24y^2z^5\epsilon^4 - 8z^7\epsilon^4.$$

6. For $\epsilon = 0.5$, the two polynomials $g_{0.5}(x,y,z), h_{0.5}(x,y,z)$ define two space algebraic surfaces, which determine as their intersection a space implicitly defined algebraic

curve denoted with $\mathcal{M}$:

$$\mathcal{M} = \{(x,y,z) \in \mathbb{R}^3 \mid g_{0.5}(x,y,z) = h_{0.5}(x,y,z) = 0\} =$$
$$= \{(x,y,z) \in \mathbb{R}^3 | g_{0.5}(x,y,z) = 0.0625 - 0.25x^2 + 0.375x^4 - 0.25x^6 + 0.0625x^8 +$$
$$1.xy - 1.x^5y - 0.25y^2 + 0.75x^2y^2 - 0.75x^4y^2 + 0.25x^6y^2 - 2.x^3y^3 + 0.375y^4 -$$
$$0.75x^2y^4 + 0.375x^4y^4 - 1.xy^5 - 0.25y^6 + 0.25x^2y^6 + 0.0625y^8 + 1.x^2z +$$
$$1.x^4z - 1.y^2z - 1.y^4z - 1.75z^2 + 3.75x^2z^2 - 2.25x^4z^2 + 0.25x^6z^2 -$$
$$2.x^3yz^2 + 3.75y^2z^2 - 4.5x^2y^2z^2 + 0.75x^4y^2z^2 - 2.xy^3z^2 -$$
$$2.25y^4z^2 + 0.75x^2y^4z^2 + 0.25y^6z^2 + 1.x^2z^3 - 1.y^2z^3 +$$
$$4.375z^4 - 3.75x^2z^4 + 0.375x^4z^4 - 1.xyz^4 - 3.75y^2z^4 +$$
$$0.75x^2y^2z^4 + 0.375y^4z^4 - 1.75z^6 + 0.25x^2z^6 +$$
$$0.25y^2z^6 + 0.0625z^8 = 0,$$
$$h_{0.5}(x,y,z) = -0.5x^2 + 0.5x^6 + 0.5y^2 + 0.5x^4y^2 - 0.5x^2y^4 - 0.5y^6 + 0.5z - 1.5x^2z +$$
$$1.5x^4z - 0.5x^6z + 2.xyz + 2.x^3yz - 1.5y^2z + 3.x^2y^2z - 1.5x^4y^2z + 2.xy^3z +$$
$$1.5y^4z - 1.5x^2y^4z - 0.5y^6z + 1.x^4z^2 - 1.y^4z^2 - 3.5z^3 + 5.x^2z^3 -$$
$$1.5x^4z^3 + 2.xyz^3 + 5.y^2z^3 - 3.x^2y^2z^3 - 1.5y^4z^3 + 0.5x^2z^4 -$$
$$0.5y^2z^4 + 3.5z^5 - 1.5x^2z^5 - 1.5y^2z^5 - 0.5z^7 = 0\}.$$

We notice that the polynomials $g_{0.5}(x,y,z), h_{0.5}(x,y,z)$ define two implicit algebraic surfaces S_1, S_2 in $\mathbb{R}^3$ whose intersection is a space implicit algebraic curve $\mathcal{M}$. In order to visualize this curve and the surfaces S_1, S_2, we need specific algebraic and geometric algorithms. For this purpose we use the Axel free system developed by [Alberti and Mourrain, 2007, Wintz, 2008]. We implement the **Algorithm** 2-`ApproxLink` in Axel, and thus we compute and we visualize the space implicit algebraic curve $\mathcal{M}$ and the surfaces S_1, S_2. For the space implicit algebraic curve $\mathcal{M}$, Axel uses subdivision methods from [Liang et al., 2008] and it outputs a piecewise linear approximation as a 3-dimensional graph data structure, see Figure 3.1. This 3-dimensional graph data structure is a set of edges in $\mathbb{R}^3$ together with their Euclidean space coordinates and a set of edges connecting them. In addition, this 3-dimensional graph data structure is called the topology of the space algebraic curve. From this figure, we also notice that for $\epsilon = 0.5$, the space algebraic curve $\mathcal{M}$ has no singularities (i.e. it is a smooth and closed space algebraic curve), and thus it represents the ϵ-link L_ϵ of the singularity $(0,0)$ of $\mathcal{C}$ for $\epsilon = 0.5$, i.e. $\mathcal{M}$ coincides with $\pi_{(\epsilon,N)}(Y_\epsilon)$. We say that the ϵ-link L_ϵ is an ϵ-algebraic link. We observe that L_ϵ is an ϵ-differentiable algebraic link. Moreover, in this case the ϵ-link L_ϵ is in fact the link of the singularity $(0,0)$ of the curve $\mathcal{C}$ defined by the squarefree complex polynomial $z^2w + w^4 \in \mathbb{C}[z,w]$, and the surfaces S_1, S_2 are part of the Milnor fibration as explained in Subsection 2.4.2. In this case we say that L is an algebraic link. In addition, we notice that L is in fact a differentiable algebraic link.

3.2.2 Applications of the Algorithm

In our approach, we approximate an ϵ-differentiable algebraic link, namely the intersection of G and H computed by the **Algorithm** 2-`ApproxLink` by a piecewise linear algebraic link. From now on, we only consider in our study piecewise linear algebraic links. We recall that with the **Algorithm** 2-`ApproxLink` we compute the ϵ-link L_ϵ of the singularity Q of the plane complex algebraic curve $\mathcal{C}$ defined by the squarefree polynomial $p(z,w) \in \mathbb{C}[z,w]$. In our approach, we basically computed two polynomials $G, H \in \mathbb{R}[x,y,z]$ by using the stereographic projection and we showed that the ϵ-link L_ϵ of the singularity Q is the zero common set of G, H. Moreover, L_ϵ is a smooth and closed implicit algebraic curve in $\mathbb{R}^3$ given as the intersection of two implicit space algebraic surfaces in $\mathbb{R}^3$ with defining polynomials G, H. If the zero common set of G and H has singularities, then L_ϵ is not a link (i.e. it is not

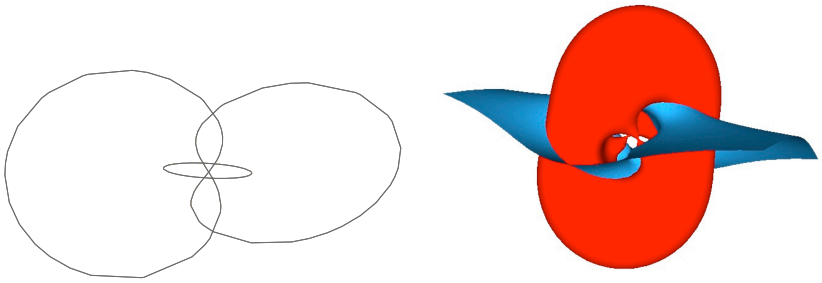


Figure 3.1: Link of the singularity $(0,0)$ of the plane complex algebraic curve $\mathcal{C}$ defined by $z^2w+w^4=0$. From left to right: (1) the link L of the singularity $(0,0)$ of $\mathcal{C}$ represented by a link with 2 components. The link L is computed as a 3-dimensional graph data structure; (2) the two algebraic surfaces that define as their intersection the link L. Pictures produced with GENOM3CK in Axel, see Chapter 5 for more information.

a smooth curve) and we return an error message. We call the ϵ-link L_ϵ an ϵ-differentiable algebraic link.

For the implementation of the **Algorithm** 2-`ApproxLink`, we use the Axel free algebraic geometric modeler. Axel uses certified algorithms [Liang et al., 2008] to compute a piecewise linear approximation $L_\epsilon^{'}$ for the ϵ-differentiable algebraic link L_ϵ, which is isotopic to L_ϵ, i.e. $L_\epsilon^{'}$ can be continuously deformed into L_ϵ. We recall that $L_\epsilon^{'}$ is computed as a 3-dimensional graph data structure $Graph=\langle V,E\rangle$, where V is a set of points (or vertices) together with their Euclidean coordinates in $\mathbb{R}^3$ and E is a set of edges connecting them. We denote $L_\epsilon^{'}:=Graph(L_\epsilon)$.

Next, from the output $Graph(L_\epsilon)$ returned by Axel for each ϵ-differentiable algebraic link L_ϵ, we compute the diagram of $Graph(L_\epsilon)$ denoted by $D\big(Graph(L_\epsilon)\big)$, as introduced in Chapter 2, Subsection 2.3.3, Definition 38. We basically compute the elements of $D\big(Graph(L_\epsilon)\big)$, i.e. the crossing points, the arcs and the number of (knot) components of each diagram. We also compute the type of each crossing point, i.e. righthanded or lefthanded crossing. These combinatorial information computed for each ϵ-differentiable algebraic link allows us to compute the ϵ-Alexander polynomial of each singularity of the plane complex algebraic curve as explained in Chapter 2, Subsection 2.4.3.

3.3 Algorithm for Computing the Approximate Alexander Polynomial

3.3.1 Sweep-Line Algorithms from Computational Geometry

This subsection contains the results from the paper [Hodorog et al., 2011] and from the technical report [Hodorog and Schicho, 2010a]. We present an adapted version of the Bentley-Ottmann algorithm [Berg et al., 2008] for computing all the intersection points among the edges of the projection of a 3-dimensional graph. In addition, the adapted algorithm computes some extra information on each intersection point and on the pair of edges that contains it. For our purpose, the adapted Bentley-Ottmann algorithm operates on a 3-dimensional graph data structure, which represents the piecewise linear approximation of a closed and smooth space algebraic curve, implicitly defined as the intersection of two algebraic surfaces and computed by the **Algorithm** 2-`ApproxLink`. We compute this space algebraic curve as the ϵ-link L_ϵ of the singularity Q of a plane complex algebraic curve $\mathcal{C}$

defined by the squarefree polynomial $p(z, w)$, as described in Section 3.2.

We manage the adapted version of the Bentley-Ottmann algorithm in a simpler way than in the original version because the 3-dimensional graph has some special properties as described in [Diestel, 2005]: (i) it consists of several cycles; (ii) it is a regular graph, i.e. it contains no loops or multiple edges; (iii) and its projection contains at most one crossing point. The first two properties are always guaranteed since the 3-dimensional graph represents the piecewise linear approximation of an implicitly defined space algebraic curve, which is closed and smooth (i.e. it does not intersects itself). We perform a test to check whether the third property holds for the given 3-dimensional graph and if the test fails, then we report a failure message. Using the free algebraic geometric modeler Axel [Wintz et al., 2006] we compute efficient and robust results.

For our purpose, the adapted Bentley-Ottmann algorithm offers essential benefits: it allows us to compute the approximate Alexander polynomial of the singularity of a plane complex algebraic curve. From the approximate Alexander polynomial we compute other approximate invariants of the singularity, e.g. the approximate Milnor number and the approximate delta-invariant. In this way, we recover topological local information on each singularity of a plane complex algebraic curve. Thus we can use the adapted algorithm to solve a specific problem from algebraic geometry, i.e. the problem of computing several topological invariants for each singularity of a plane complex algebraic curve. These topological invariants play an important role in the classification and the analysis of the singularities of a plane complex algebraic curve as discussed in [Arnold et al., 1998].

We recall that the Bentley-Ottmann algorithm for reporting the pairwise intersections among a set of objects in the plane, proved itself useful in many applications from combinatorial geometry and computer graphics. A generalized version of the Bentley-Ottmann algorithm [Bieri and Schmidt, 1991] computes the pairwise intersections among geometric objects in $\mathbb{R}^d$. The Bentley-Ottmann algorithm uses a *sweep* technique, i.e. a sweep plane (or a sweep line in $\mathbb{R}^2$) sweeps the space $\mathbb{R}^d$ (or $\mathbb{R}^2$) that contains a set of geometric objects. At certain positions called event points, the sweep is interrupted and the problem is locally solved. The sweep is greedy, without any backtracking.

Data Structures

For our study, we define a 3-dimensional graph data structure as follows:

Definition 59. A (3-dimensional) graph is defined as a pair $\mathcal{G} = \langle V, E \rangle$, where V is a list of points (vertices) in the 3-dimensional space together with their Euclidean coordinates, and E is a list of edges connecting them, i.e. $V = \{p(x, y, z) \in \mathbb{R}^3\}$ and $E = \{e(i, j) \mid i, j \in V\}$.

We are interested in the following elements of a 3-dimensional graph:

Definition 60. A point (or a vertex) in the 3-dimensional graph is a 4-tuple of the form $p(index, x, y, z)$, where $index \in \mathbb{Z}$ uniquely identifies each point in the graph, and $(x, y, z) \in \mathbb{R}^3$ are the Euclidean coordinates of the point. An edge in the 3-dimensional graph is defined as a 2-tuple $e(s, d)$, where s is the index of the source point of e and d is the index of the destination point of e, see Figure 3.2.

We now present several notations, which we use in the rest of this thesis.

Remark 11. We introduce the following notations:

- A 3-dimensional graph is denoted as a pair $G = \langle V, E \rangle$, where V is a set of points in the 3-dimensional space together with their Euclidean coordinates, and E is a set of edges connecting them, i.e. $V = \{p(x, y, z) \in \mathbb{R}^3\}$ and $E = \{e(i, j) \mid i, j \in P\}$.

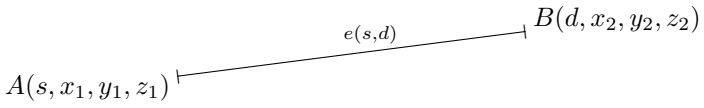


Figure 3.2: An edge $e(s, d)$ in a 3-dimensional graph. The edge e is determined by its source point $A(s, x_1, y_1, z_1)$ and by its destination point $B(d, x_2, y_2, z_2)$, where $s, d \in \mathbb{Z}$ uniquely identify the points A, B and $(x_1, y_1, z_1), (x_2, y_2, z_2) \in \mathbb{R}^3$ are the Euclidean coordinates of A, B.

- We denote a point in the graph as a 4-tuple $p(index, x, y, z)$, where $index \in \mathbb{Z}$ uniquely identifies each point in the graph, and $(x, y, z) \in \mathbb{R}^3$ are the Euclidean coordinates of the point.
- We denote an edge in the graph as a 2-tuple $e(s, d)$, where s is the index of the source point of e and d is the index of the destination point of e.
- We use the dot notation for accessing the elements of a tuple, i.e. $p.index$, $p.x$, $p.y$, $p.z$, $e.s$, $e.d$.
- For a random point $p(index, x, y, z)$ of a 3-dimensional graph, we introduce the following notations:
  - $point3D(index) = (x, y, z) \in \mathbb{R}^3$ for denoting the x, y, z coordinates of $index$.
  - $point2D(index) = (x, y) \in \mathbb{R}^2$ for denoting the x, y coordinates of $index$. We alternatively use the notation $xycoord(index) = (x, y) \in \mathbb{R}^2$ for denoting the x, y coordinates of $index$.
  - $xcoord(index) = x \in \mathbb{R}$ for denoting the x coordinate of $index$.
  - $ycoord(index) = y \in \mathbb{R}$ for denoting the y coordinate of $index$.
  - $zcoord(index) = z \in \mathbb{R}$ for denoting the z coordinate of $index$.
- For a random edge $e(a, b)$ of a 3-dimensional graph with $a, b \in \mathbb{Z}$, we use the notations:
  - $source(e) = a \in \mathbb{Z}$ for denoting the index of the source point of e.
  - $destination(e) = b \in \mathbb{Z}$ for denoting the index of the destination point of e.
- We access the i-th component of a list (or a vector) SW with the underscore notation for the index i, i.e sw_i. We consider that the indexes of a list (or a vector) start from 0. In addition, we distinguish between the name of the list, which is denoted with upper case letters (i.e. SW), and the elements of the list themselves that are denoted with lower case letters, i.e. sw_0.
- Given a list (or a vector) denoted with SW, we use the notation $length(SW)$ to denote its length.
- In order to state that an object is not empty we will use the predicate symbol $IsNotEmpty(object)$, which will be true when the object is not empty, and false otherwise.
- We will use the *Null* pointer notation as in the C++ programming language, whenever an algorithm has "nothing" as its returning value.

We recall that a *path* in the 3-dimensional graph is a sequence of consecutive edges in a graph, and a *cycle* (*circuit*) is a path that ends at the vertex it begins. In addition, a *loop* is an edge that connects a vertex to itself, and *multiple edges* are two or more edges connecting the same two vertices, see [Diestel, 2005] for details.

We assume that the 3-dimensional graph is simple (regular), i.e. it has no multiple edges or loops as in Figure 3.3. For our purpose, we are interested in the projection of a 3-dimensional graph that always consists of several cycles, see Figure 3.4 for an example. We consider the edges of a 3-dimensional graph $\mathcal{G}$ to be "small" edges, i.e. the projection of any edge of $\mathcal{G}$ has at most one crossing point. If this property is not true for a certain pair of edges from a 3-dimensional graph, then we report a failure message during runtime.

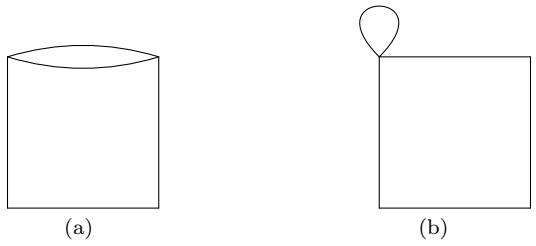


Figure 3.3: (a) A graph with multiple edges. (b) A graph with a loop.

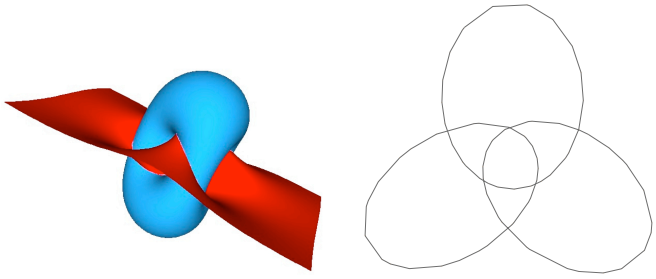


Figure 3.4: (a) Two algebraic surfaces that implicitly define as their intersection a closed and smooth space algebraic curve computed as a 3-dimensional graph $\mathcal{G}$ with 3 cycles. (b) The projection of the 3-dimensional graph $\mathcal{G}$ with 3 cycles from (a). Pictures produced with GENOM3CK in Axel, see Chapter 5 for details.

Remark 12. The 3-dimensional graph $\mathcal{G}$ that we study in this subsection represents the piecewise linear approximation of a closed and smooth implicitly defined space algebraic curve. We compute this curve as the ϵ-link L_ϵ of the singularity $(0,0)$ of a plane complex algebraic curve $\mathcal{C}$. For sufficiently small ϵ, from Chapter 2, Theorem 10 we know that L_ϵ equals the link of the singularity $(0,0)$ of $\mathcal{C}$ and it characterizes completely the topology of the curve $\mathcal{C}$ around its singularity $(0,0)$. For instance, in Figure 3.4 we visualize the link of the singularity $(0,0)$ of the plane complex algebraic curve defined by the squarefree polynomial $x^3 - y^3 = 0$. In the literature [Sendra and Alcazar, 2005], [Liang et al., 2008], the 3-dimensional graph computed as the piecewise linear approximation of an implicitly defined space algebraic curve is called the topology of the curve. We use the Axel [Wintz et al., 2006] free algebraic geometric modeler to compute the 3-dimensional graph as presented in Section 3.2. For the special case of smooth implicitly defined space

algebraic curves, Axel uses certified algorithms to compute their topology as explained in Subsection 3.2.2. Thus in our study, the 3-dimensional graph $\mathcal{G}$ represents the approximation denoted with $Graph(L_\epsilon)$ of the ϵ-link L_ϵ of the singularity $(0,0)$ of the plane complex algebraic curve $\mathcal{C}$. We mention that basically the ϵ-link L_ϵ is computed using the Axel system as a graph data structure, and thus in our study $\mathcal{G} = Graph(L_\epsilon)$.

We state the subproblem that we want to solve:

Subproblem 1. *Given the following:*

(i) a 3-dimensional graph $\mathcal{G} = \langle V, E \rangle$ as in Definitions 59 and 60, which has only "small" edges, which is regular and which consists of several cycles,

our goal is:

(1) to compute the intersection points among all the edges of the projection of $\mathcal{G}$. In addition, we compute some extra information:

(1.1) for each intersection point Q find the pair of edges (e_m, e_n) that contains it.

(2.2) the pair of edges (e_m, e_n) is ordered, i.e. e_m is under e_n in the 3-dimensional Euclidean space $\mathbb{R}^3$.

Basic Coordinate Geometry Algorithms

To solve Subproblem 1 we first need to design several algorithms from coordinate geometry, which we call basic coordinate geometry algorithms, since they are straightforward and they do not imply a lot of complicated computations. These algorithms allow us:

- to compute the slope of a line, given two points in $\mathbb{R}^2$;
- to compute the Y-intercept of a line, given two points in $\mathbb{R}^2$;
- to compute the equation of a line, given its slope and its Y-intercept;
- to compute the equation of a line, given two points in $\mathbb{R}^2$;
- to decide whether a point in $\mathbb{R}^2$ lies on a line of a certain given equation.

We denote the slope of a line in the coordinate plane system Oxy with m. We define m as the ratio of the change in the y-value over the corresponding change in the x-value between two distinct points on the line. We consider $A(a,b)$ and $B(u,v)$ two points with given coordinates in the 2-dimensional Euclidean plane $\mathbb{R}^2$. It follows that the slope of the line AB denoted with m is defined as the ratio $m = \dfrac{v-b}{u-a}$, under the hypothesis that $(u-a) \neq 0$. We notice that if $(u-a) = 0$, then the value of the slope m is ∞, thus undefined. In fact if $m = \infty$, then the corresponding line is a vertical line, i.e. a parallel line to the y-axis. We recall that if two lines have the same slope, then they are parallel. We now describe the algorithm `GetSlope`(A, B), which computes the slope of the line AB determined by two points A, B in the Euclidean plane $\mathbb{R}^2$.

Algorithm 3 Slope of the line AB, determined by two points A, B in the Euclidean plane: `GetSlope`(A, B)

Input: $A(a, b),\ B(u, v) \in \mathbb{R}^2$ two points in the Euclidean plane $\mathbb{R}^2$.
Output: $m \in \mathbb{R}$,
where m equals the slope of the edge (and of the line) AB, determined by the two points A, B.

1: **if** $(u - a) \neq 0$ **then**
2: **return** $m = \dfrac{v - b}{u - a}$
3: **else**
4: **print** `Zero denominator! The slope is undefined!`
5: **end if**

We denote the Y-intercept of a line in the coordinate plane system Oxy with n. We define n as the distance on the y-axis from the origin $O(0, 0)$ to the point where the line intercepts the y-axis of the coordinate plane system Oxy. If we consider $P(0, n)$ to be the point where the line intercepts the y-axis, then the Y-intercept equals the y-coordinate of the point P. Given the same points $A(a, b),\ B(u, v)$ as in **Algorithm** 3-`GetSlope` and supposing that the equation of the line AB determined by the two points A and B is $y = mx + n$, where $m = \dfrac{v - b}{u - a}$ with $u - a \neq 0$ is the slope of the line as computed with the **Algorithm** 3-`GetSlope`(A, B) and n is the Y-intercept of the line AB, then we can compute the value of the Y-intercept as $n = y - mx = y - \dfrac{v - b}{u - a}x$, assuming that $u - a \neq 0$. Since the point $B(u, v)$ belongs to the line AB, we obtain the value for the Y-intercept to be $n = v - \dfrac{v - b}{u - a}u = \dfrac{b \cdot u - a \cdot v}{u - a}$, for $(u - a) \neq 0$. If $u - a = 0$, then we notice that the Y-intercept of the line AB is undefined. We present the algorithm `GetYIntercept`(A, B) for computing the Y-intercept of the line AB, determined by two points A, B in the Euclidean plane $\mathbb{R}^2$.

Algorithm 4 Y-intercept of the line AB, determined by two points A, B in the Euclidean plane: `GetYIntercept`(A, B)

Input: $A(a, b),\ B(u, v) \in \mathbb{R}^2$ two points in the Euclidean plane $\mathbb{R}^2$.
Output: $n \in \mathbb{R}$,
where n equals the Y-intercept of the edge (and of the line) AB, determined by the two points A, B.

1: **if** $(u - a) \neq 0$ **then**
2: **return** $n = \dfrac{b \cdot u - a \cdot v}{u - a}$
3: **else**
4: **print** `Zero denominator! The Y-intercept is undefined!`
5: **end if**

We recall that a line in the coordinate plane system Oxy, with slope m and Y-intercept n, has the defining equation $y = mx + n$ or equivalently $mx - y + n = 0$. Given the same points $A(a, b),\ B(u, v)$ with $m = \dfrac{v - b}{u - a}$ and $n = \dfrac{b \cdot u - a \cdot v}{u - a}$ for $u - a \neq 0$ as computed in **Algorithm** 3-`GetSlope` and respectively in **Algorithm** 4-`GetYIntercept` for $(u - a) \neq 0$, we get the following form for the defining equation of the line AB : $y = \dfrac{v - b}{u - a} \cdot x - \dfrac{b \cdot u - a \cdot v}{u - a}$. A straightforward computation produces the equation of the

line $AB : (b-v)x + (u-a)y + (a \cdot v - b \cdot u) = 0$. We notice that for the defining equation of an arbitrary line AB computed from two points A, B in $\mathbb{R}^2$, it is enough to return the coefficients of the defining equation of the line in the variables x and y (if this equation is defined), as described in the following algorithm `EqnLine`(A, B).

Algorithm 5 Equation of the line AB, determined by two points A, B in the Euclidean plane: `EqnLine`(A, B)

Input: $A(a, b),\ B(u, v) \in \mathbb{R}^2$ two points in the Euclidean plane $\mathbb{R}^2$.
Output: $\alpha, \beta, \gamma \in \mathbb{R}$,
where α, β, γ are the real coefficients of the equation of the line $AB : \alpha x + \beta y + \gamma = 0$.

1: **if** $(u - a) \neq 0$ **then**
2: $\alpha = (b - v)$
3: $\beta = (u - a)$
4: $\gamma = a \cdot v - b \cdot u$
5: **return** (α, β, γ)
6: **else**
7: **print** `The equation of the line` AB `is undefined!`
8: **end if**

In the rest of this thesis, we consider an edge in the projection of a 3-dimensional graph data structure as a line in the Euclidean plane. We notice that if we are given an edge as a pair $e(s, d)$, where s is the index of the source point of e and d is the index of the destination point of e, then we can compute the equation of the edge e using the **Algorithm** 5-`EqnLine`(A, B), where the coordinates of the source point and of the destination point of e are given by $A\big(xcoord(s), ycoord(s)\big)$ and $B\big(xcoord(d), ycoord(d)\big)$. Thus if we are given an arbitrary point $Q(q, r)$ and an arbitrary edge as a pair $e(s, d)$, we can first compute the equation of the edge e and then we can decide whether the point $Q(q, r)$ belongs to the edge $e(s, d)$ or not. If we suppose that the equation of the edge returned by the **Algorithm** 5-`EqnLine`(A, B) is defined and is given by the equation $\alpha x + \beta y + \gamma = 0$, then we can compute the quantity $value = \alpha q + \beta r + \gamma$. It follows that the point $Q(q, r)$ belongs to the edge $e(s, d)$ if and only if $value = 0$. We now describe the algorithm `EvalAtPointEqnLine`$\big(e(s, d), Q\big)$, which computes the value of the equation of the edge $e(s, d)$ evaluated at the point $Q(q, r)$.

Algorithm 6 Value of the equation of the edge $e(s, d)$ evaluated at the point Q from the Euclidean plane: `EvalAtPointEqnLine`$\big(e(s, d), Q\big)$

Input: $e(s, d)$ an edge in the Euclidean plane $\mathbb{R}^2$,
$Q(q, r) \in \mathbb{R}^2$ a point in the Euclidean plane $\mathbb{R}^2$.
Output: $value \in \mathbb{R}$,
where $value$ equals the real value of the equation of the edge e evaluated at Q.

1: $a = xcoord(s), b = ycoord(s)$
2: $u = xcoord(d), v = ycoord(d)$
3: consider $A(a, b), B(u, v)$
4: **if** $(u - a) \neq 0$ **then**
5: (α, β, γ)=`EqnOfLine`(A, B)
6: **return** $value = \alpha q + \beta r + \gamma$
7: **else**
8: **print** `The equation of the edge` e `is undefined!`
9: **end if**

We notice that all the basic coordinate geometry algorithms require $O(1)$ constant time and $O(1)$ constant space (memory) for their computation.

Methodology

To solve Subproblem 1, we secondly compute the intersection points of all the edges of the projection of a 3-dimensional graph, and for each intersection point, we compute the pair of edges that contains it. For this purpose, we design a sweep line based algorithm as the Bentley-Ottmann algorithm from [Berg et al., 2008]. We distinguish several steps for our adapted Bentley-Ottmann algorithm, that we describe in comparison with the original Bentley-Ottmann algorithm:

Step 1 (Ordering criteria). The edges of the projection of the 3-dimensional graph $\mathcal{G}$ are oriented from left to right and they are ordered in the list of edges $E = \{e_0, ..., e_N\}$ as in Figure 3.5: (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.

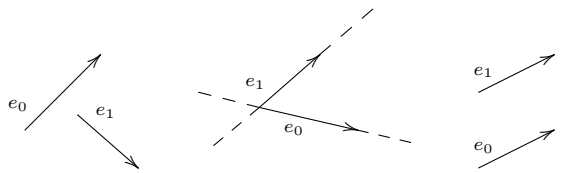


Figure 3.5: Ordering criteria for the edges.

Step 2 (Sweep line paradigm). As in the Bentley-Ottmann 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, which are stored in a list denoted SW and that we call the sweep list. The status of SW changes while l sweeps the plane and it is updated only at certain points of the edges from E called event points. As in the Bentley-Ottmann algorithm the event points are the source and the destination points of the edges (segments) and the detected intersection points. In this algorithm, the sweep list SW is ordered by the y-coordinates of the intersections of the edges of E with the sweep line l. Hence, as in the Bentley-Ottmann algorithm, SW represents the status of the algorithm and it contains the ordered sequence of segments intersecting the sweep line.

Step 3 (Initialization). We consider E the list of ordered edges as described in **Step 1**, and SW the sweep list as described in **Step 2**. We denote with I the list of intersection points of all the edges of the projection of the 3-dimensional graph $\mathcal{G}$, and with EI the list of all pairs of intersection edges that contain the intersection points. At the end of the algorithm, the i-th element of the list EI represents the pair of edges that contains the i-th intersection point from the list I, with the extra information that the first edge from the pair of edges is under the second edge from the pair of edges in $\mathbb{R}^3$. In the initialization step of our adapted Bentley-Ottmann algorithm, E contains all the ordered edges of the projection of $\mathcal{G}$, SW contains always the first two edges of E, whereas I and EI are empty.

Step 4 (Sweep line management). We observe that in E each *index* appears two times since E always contains several cycles. This allows us to manage SW in a simpler way in our adapted Bentley-Ottmann 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 $SW = \{e_0^i, e_1^i, e_2^i, ..., e_k^i\}$, with $e_q^i \in E$ 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 $e_0^i, ..., e_j^i$ with l, and smaller than the y-coordinates of all the intersections of $e_{j+1}^i, ..., e_k^i$ with l. This index j can be found by checking all the signs of the determinants constructed with $(xycoord(s_m), 1), (xycoord(s_j^i), 1)$ and $(xycoord(d_j^i), 1)$. Then we insert e_m in SW between the two edges e_j^i and e_{j+1}^i and we obtain $SW = \{e_0^i, e_1^i, ..., e_j^i, e_m, e_{j+1}^i, ..., e_k^i\}$. When we insert an edge from E into SW on the right position, we have to additionally update SW depending on the encountered event points:

- we test each inserted edge in SW against its two neighbours for intersection. If an intersection point P is found we report it together with the pair of edges that contains it. In addition, we swap the edges that intersect in SW. As opposed to the original Bentley-Ottmann algorithm after swapping the edges in SW, we do not test the edges against their new neighbours for intersections because we consider only "small" edges.

- we test each inserted edge in SW against its two neighbours for common destination. In addition, when two edges are swapped in SW after reporting their intersection point, we test them against their new neighbours for common destination. Whenever we find two consecutive edges with common destinations we erase them from SW. As opposed to the original Bentley-Ottmann algorithm after deleting edges from SW, we do not test the new neighbours for intersection because we consider only "small" edges.

We notice that in the adapted Bentley-Ottmann algorithm we basically process the pre-ordered list of edges E in a for-loop in a way which makes the explicit use of a data structure for storing the event points redundant. We recall that in the Bentley-Ottmann algorithm a separate data structure such as a balanced binary search tree is required for storing the event points, data structure which is not needed in our adapted version of the algorithm.

Remark 13. We mention briefly a way to modify the adapted Bentley-Ottmann algorithm such that in the case of a 3-dimensional graph $\mathcal{G}$ with "long" edges (i.e. the projection of any edge of $\mathcal{G}$ has at least one crossing point), the algorithm would detect all the intersection points and would not only report a failure message at runtime. The main idea is to update the ordered list of edges E and the sweep list SW each time the algorithm reports an intersection point as follows: if the algorithm reports the intersection point $P(x, y) \in \mathbb{R}^2$ together with the pair of edges (e_1, e_2) that contains $P(x, y)$, then for $i = 1, 2$ we split each edge of intersection e_i in two new edges e_i^l, e_i^r. The new vertices e_i^l are determined by the source point of e_i and by the coordinates of $P(x, y)$, whereas the new edges e_i^r are determined by the coordinates of $P(x, y)$ and by the destination point of e_i, as described in Figure 3.6. Then we update the lists SW and E as follows: we replace the edges e_i by e_i^l in SW, and we insert the edges e_i^r in E following the ordering criteria from **Step 1**, Figure 3.5.

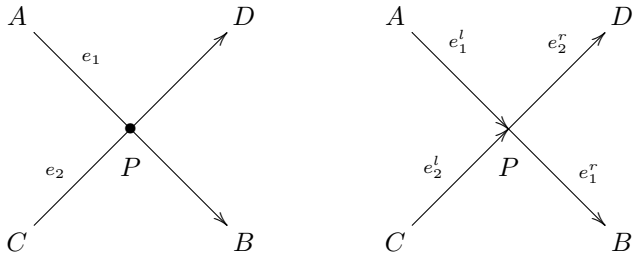


Figure 3.6: Refinements of the adapted Bentley-Ottmann algorithm. If the intersection point P is reported together with its corresponding pair of edges (e_1, e_2), then each edge e_1, e_2 is split in two new edges, i.e. e_1 is split in e_1^l, e_1^r, and e_2 is split in e_2^l, e_2^r. The new vertices e_i^l are determined by the source point of e_i and by the coordinates of P, while the new edges e_i^r are determined by the coordinates of P and by the destination point of e_i, for $i \in \{1, 2\}$. We replace the edges e_i by e_i^l in SW, and we insert the edges e_i^r in E following the ordering criteria from **Step 1**, Figure 3.5.

In the following we assume that we have computed: (1) a list $I = \{(x_i, y_i) \in \mathbb{R}^2\}$ of the intersection points of all the edges of the projection of a 3-dimensional graph; (2) and a list EI of pairs of edges for I such that the i-th element of EI represents the pair of edges that contains the i-th intersection point from I. In the example from Figure 3.4, our adapted Bentley-Ottmann algorithm computes all the 6 intersection points together with the list of pairs of edges that contain these intersection points.

To solve Subproblem 1, we now have to order each pair of edges from EI depending on the Euclidean space coordinates of the intersection points from I. For instance, in Figure 3.7 we consider $P(x, y) \in I$ the intersection point of the pair of edges $(e_1, e_2) \in EI$. We order this pair such that the first component always lies under the second component in $\mathbb{R}^3$. We assume that for $i = \{1, 2\}$ the source and the destination points of e_i are $A_i(a_i, b_i, 0)$, $B_i(u_i, v_i, 0)$, which are the projections of $A_i^{'}(a_i, b_i, c_i)$, $B_i^{'}(u_i, v_i, w_i)$ from $\mathbb{R}^3$. To order the pair of edges we proceed as follows:

1. For $i = \{1, 2\}$ we compute the equations of the support lines L_i for the edges e_i in $\mathbb{R}^2$. We use the determinant formula for the equations of the lines L_i and we obtain:

$$L_i(x, y) : \det \begin{pmatrix} a_i & b_i & 1 \\ u_i & v_i & 1 \\ x & y & 1 \end{pmatrix} = 0\,, \tag{3.8}$$

   and thus $L_i(x, y) : (b_i - v_i)x + (u_i - a_i)y + a_i v_i - b_i u_i = 0$ for $u_i - a_i \neq 0$.

2. We compute the coordinates z_1, z_2 of $P_1(x, y, z_1)$ and $P_2(x, y, z_2)$ in $\mathbb{R}^3$ as in Figure 3.7. As an example we compute z_1 (we proceed in the same way for z_2). Firstly we compute α_1 from

$$\alpha_1 L_2(A_1) + (1 - \alpha_1) L_2(B_1) = 0\,. \tag{3.9}$$

   Then we compute z_1 as $z_1 = \alpha_1 c_1 + (1 - \alpha_1) w_1$.

3. If $z_1 < z_2$, then e_1 is under e_2 in $\mathbb{R}^3$ and we return the pair (e_1, e_2) for $P(x, y)$ $\big($i.e. e_1 is the undergoing edge and e_2 is the overgoing edge for $(e_1, e_2)\big)$; otherwise e_2 is under e_1 in $\mathbb{R}^3$ and we thus return the pair (e_2, e_1) for $P(x, y)$ $\big($i.e. e_2 is the undergoing edge and e_1 is the overgoing edge for $(e_2, e_1)\big)$, as in the example from Figure 3.7.

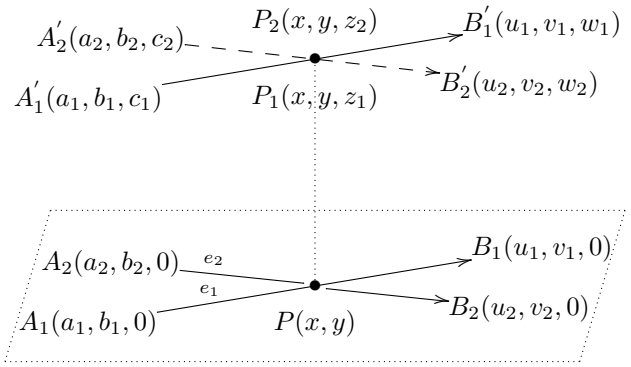


Figure 3.7: Ordering the pair of edges (e_1, e_2) that contains an intersection point $P(x, y)$ with respect to the Euclidean space coordinates of the edges e_1, e_2.

Description of the Sweep-Line Algorithms

We now describe the algorithms that we design for solving Subproblem 1, algorithms that are based on the methods presented in the previous paragraph. We basically distinguish two basic algorithms for solving Subproblem 1, i.e.:

(1) An algorithm for computing all the intersection points among all the edges of the projection of a 3-dimensional graph. In addition, for each detected intersection point, this algorithm computes the pair of edges that contains it. We refer to this algorithm as the algorithm for computing the intersection points.

(2) An algorithm that arranges the pair of edges reported for each computed intersection point depending on the Euclidean space coordinates of the edges and of the detected intersection point. For instance, if $I(x, y)$ is the intersection point reported for the pair of edges (e_1, e_2) with $I(x, y) = e_1 \cap e_2$, then this algorithm will report the pair of edges (e_1, e_2) if e_1 is below e_2 in the 3-dimensional Euclidean space, or the algorithm will report the pair (e_2, e_1) if e_2 is below e_1 in the 3-dimensional Euclidean space. We refer to this algorithm as the algorithm for arranging the pair of intersecting edges.

As we discussed in the previous paragraph, the first algorithm is a sweep-line algorithm, while the second algorithm depends on the first one. Therefore we call these algorithms sweep-line algorithms. We sometimes refer to both of these algorithms as an adapted-version of the Bentley-Ottmann algorithm.

Algorithm for computing the intersection points. For solving Subproblem 1, we first need a basic algorithm to compute the intersection point of two edges e_1, e_2, if such an intersection point exists. We call this algorithm `FindIntersection`(e_1, e_2). This algorithm uses an auxiliary algorithm `ComputeIntersection`(e_1, e_2), which effectively computes the coordinates of the intersection point $I(x, y)$ of the pair of edges (e_1, e_2). In the following we describe the main idea of the algorithm `ComputeIntersection`. To find the coordinates of the intersection point $I(x, y)$, we solve a linear system of two equations in the x, y unknowns, system that is determined by the defining equations of the two edges e_1, e_2. We first extract the coordinates of the source and of the destination points of each edge e_1, e_2. Using the **Algorithm** 3-`GetSlope` and the **Algorithm** 4-`GetYIntercept`, we compute the slopes and the Y-intercepts of the two edges denoted with m_1, n_1 for e_1 and respectively with m_2, n_2

for e_2. We get the following linear system of equations in the x, y indeterminates, system which is formed by the equations of the two edges e_1, e_2:

$$\begin{cases} m_1 x - y + n_1 = 0 \\ \\ m_2 x - y + n_2 = 0. \end{cases} \tag{3.10}$$

We assume that $m_1 - m_2 \neq 0$ and by using Cramer's rule we get as solutions to System (3.10) the coordinates (x, y) of the intersection point $I(x, y)$ of the pair of edges (e_1, e_2), where $x = \dfrac{n_2 - n_1}{m_1 - m_2}$ and $y = \dfrac{m_1 \cdot n_2 - m_2 \cdot n_1}{m_1 - m_2}$. We add that if $m_1 - m_2 = 0$ (i.e. the two edges e_1 and e_2 are parallel), then we report a failure message. We now present the auxiliary algorithm `ComputeIntersection`(e_1, e_2) for computing the coordinates (x, y) of the intersection point $I(x, y)$ of the pair of edges (e_1, e_2), if such an intersection point exists.

Algorithm 7 Compute the coordinates of the intersection point $I(x, y) \in \mathbb{R}^2$ of the pair of edges (e_1, e_2) in the Euclidean plane: `ComputeIntersection(`e_1, e_2`)`

Input: $e_1(s_1, d_1), e_2(s_2, d_2)$ two edges in the Euclidean plane $\mathbb{R}^2$.
Output: $(x, y) \in \mathbb{R}^2$,
where the pair (x, y) represents the coordinates of the intersection point $I(x, y)$ of the pair of edges (e_1, e_2), i.e. $I(x, y) = e_1 \cap e_2$.

1: $a_1 = xcoord(s_1), b_1 = ycoord(s_1), u_1 = xcoord(d_1), v_1 = ycoord(d_1)$
2: $a_2 = xcoord(s_2), b_2 = ycoord(s_2), u_2 = xcoord(d_2), v_2 = ycoord(d_2)$
3: consider $A_1(a_1, b_1), B_1(u_1, v_1)$
4: consider $A_2(a_2, b_2), B_2(u_2, v_2)$
5: $m_1 =$ `GetSlope`$(A_1, B_1), n_1 =$ `GetYIntercept`(A_1, B_1)
6: $m_2 =$ `GetSlope`$(A_2, B_2), n_2 =$ `GetYIntercept`(A_2, B_2)
7: **if** $(m_1 - m_2 \neq 0)$ **then**
8: x=$\dfrac{n_2 - n_1}{m_1 - m_2}$, y=$\dfrac{m_1 \cdot n_2 - m_2 \cdot n_1}{m_1 - m_2}$
9: **return** (x, y)
10: **else**
11: **print** `The two edges are parallel and thus they do not intersect!`
12: **end if**

Next, we explain the idea of the algorithm `FindIntersection`(e_1, e_2). This algorithm tests whether two edges e_1, e_2 intersect or not. If the edges intersect, then the algorithm uses the auxiliary algorithm called `ComputeIntersection`(e_1, e_2) to compute the coordinates of the intersection point $I(x, y)$ of the pair of edges (e_1, e_2). If the edges do not intersect, then the algorithm returns the *Null* pointer. We present the test for deciding whether two edges e_1, e_2 intersect or not. Given two edges $e_1(s_1, d_1), e_2(s_2, d_2)$ we first extract the coordinates of their source and of their destination points. We assume that e_1 has the source point $A_1(a_1, b_1)$ and the destination point $B_1(u_1, v_1)$, whereas e_2 has the source point $A_2(a_2, b_2)$ and the destination point $B_2(u_2, v_2)$. We compute the defining equations of the two edges e_1 and e_2 using their slopes and their Y-intercepts, i.e. m_1, n_1 for e_1 and m_2, n_2 for e_2. Hence, we compute the equations of the two edges e_1 and e_2 denoted with $L_1(x, y)$, and respectively with $L_2(x, y)$. By a straightforward computation we obtain $L_1(x, y) : m_1 \cdot x - y + n_1 = 0$ and $L_2(x, y) : m_2 \cdot x - y + n_2 = 0$. If the two edges e_1 and e_2 do intersect, then the following two conditions have to be simultaneously true:

1. condition 1: we consider $L_1(x, y)$ the equation of the edge e_1. If e_1 intersects e_2, then the points A_2 and B_2 have to be on opposite semiplanes determined by e_1, i.e. the condition $L_1(A_2) \cdot L_1(B_2) < 0$ has to be true;

2. condition 2: we consider $L_2(x, y)$ the equation of the edge e_2. If e_2 intersects e_1, then the points A_1 and B_1 have to be on opposite semiplanes determined by e_2, i.e. the condition $L_2(A_1) \cdot L_2(B_1) < 0$ has to be true.

We now describe the algorithm `FindIntersection`(e_1, e_2), which returns the coordinates (x, y) of the intersection point $I(x, y)$ of the pair of edges (e_1, e_2) in $\mathbb{R}^2$, if such an intersection point exists, and which returns the *Null* pointer if such an intersection point does not exist.

Algorithm 8 Decide whether two edges e_1 and e_2 intersect in the Euclidean plane and compute the intersection point in the affirmative case: `FindIntersection`(e_1, e_2)

Input: $e_1(s_1, d_1), e_2(s_2, d_2)$ two edges in the Euclidean plane $\mathbb{R}^2$.
Output: If the two edges e_1 and e_2 intersect, then return their intersection point $I(x, y)$ with $I = e_1 \cap e_2$, otherwise return the *Null* pointer.

1: $a_1 = xcoord(s_1), b_1 = ycoord(s_1), u_1 = xcoord(d_1), v_1 = ycoord(d_1)$
2: $a_2 = xcoord(s_2), b_2 = ycoord(s_2), u_2 = xcoord(d_2), v_2 = ycoord(d_2)$
3: consider $A_1(a_1, b_1), B_1(u_1, v_1)$
4: consider $A_2(a_2, b_2), B_2(u_2, v_2)$
5: $(m_1, n_1, p_1) =$ `EqnLine`(A_1, B_1)
6: $(m_2, n_2, p_2) =$ `EqnLine`(A_2, B_2)
7: $value_1$=`EvalAtPointEqnLine` $\big(A_2, e_1(s_1, d_1)\big)$
8: $value_2$=`EvalAtPointEqnLine` $\big(B_2, e_1(s_1, d_1)\big)$
9: $value_3$=`EvalAtPointEqnLine` $\big(A_1, e_2(s_2, d_2)\big)$
10: $value_4$=`EvalAtPointEqnLine` $\big(B_1, e_2(s_2, d_2)\big)$
11: **if** $(m_1 - m_2) = 0$ **then**
12: **return** *Null* {the edges are parallel}
13: **end if**
14: **if** $(value_1 \cdot value_2 < 0)$ and $(value_3 \cdot value_4 < 0)$ **then**
15: **return** $(x, y) =$`ComputeIntersection`(e_1, e_2) {the edges do intersect}
16: **else**
17: **return** *Null* {the edges do not intersect}
18: **end if**

For solving Subproblem 1, we secondly need a basic algorithm to introduce a current edge $e(s, d)$ from E into the right position in the sweep list SW as described in **Step 4 (Sweep line management)** of the adapted Bentley-Ottmann algorithm. The main purpose of this algorithm is thus to keep the sweep list SW ordered. We call this basic algorithm `InsertSW`(e, SW). The algorithm `InsertSW`(e, SW) uses an auxiliary algorithm `ComputeDet`(A, B, P). This auxiliary algorithm computes the value of the determinant formed by the coordinates of the three points $A(a, b), B(u, v), P(m, n)$.

In the following, we assume that an arbitrary edge $sw(s, d)$ from the sweep list has the point $A(a, b)$ as its source point and the point $B(u, v)$ as its destination point. Given the point $P(m, n)$ we want to test whether P lies above or below the edge sw in the 2-dimensional Euclidean plane $\mathbb{R}^2$. If the value of the determinant formed by the three points A, B, P computed with the algorithm `ComputeDet`(A, B, P) is positive, then the point P is above the edge sw. If the value of the determinant is negative, then the point P is below the edge sw. Moreover, if the value of the determinant is zero, then the three points are collinear. In addition, if we assume that the point P is the source point of the current edge $e(s, d)$ from E, which has to be inserted in the sweep list SW in the right position, then by computing the value of the determinant formed by the three points A, B, P, we can decide the position of the edge e from E towards the edge sw from the sweep list SW. We distinguish the following possible cases:

(1) If the value of the determinant is positive, then e is above sw in $\mathbb{R}^2$ and thus e has to be inserted after the edge sw in the sweep list SW, see Figure 3.8, (a).

(2) If the value of the determinant is negative, then e is below sw in $\mathbb{R}^2$ and thus e has to be inserted before the edge sw in the sweep list SW, see Figure 3.8, (b).

(3) If the value of the determinant is zero, then e and sw lie on the same line, i.e. they have a common index point (either the source point or the destination point). In this case, we insert the current edge e from E instead of the edge sw in the sweep list SW, see Figure 3.8, (c).

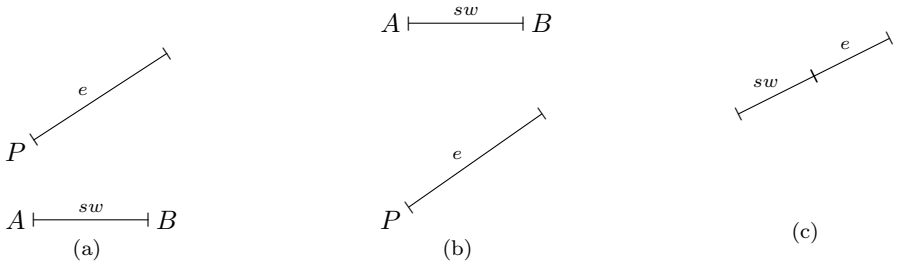


Figure 3.8: Position of an edge $e(s, d)$ from the ordered list of edges E towards an arbitrary edge sw from the sweep list SW.

Algorithm 9 Compute the value of the determinant formed by the Euclidean plane coordinates of three points A, B, P in $\mathbb{R}^2$: `ComputeDet`(A, B, P)

Input: $A(a, b), B(u, v), P(m, n) \in \mathbb{R}^2$.
Output: $\det(A, B, P) \in \mathbb{R}$,
where $\det(A, B, P)$ is the value of the determinant formed by the three points A, B, P.

1: **return** $\det(A, B, P) = -u \cdot b + m \cdot b + a \cdot v - m \cdot v - a \cdot n + u \cdot n$

We now describe the algorithm `InsertSW`(e, SW), which inserts the current edge e from E in the sweep list SW on the right position as described in **Step 4 (Sweep line management)** of the adapted Bentley-Ottmann algorithm. As discussed in the previous paragraph, we assume that the edge $e(s, d)$ has the point $P(m, n)$ as its source point. We consider an arbitrary edge at position i in the sweep list, which we denote with $sw_i(s, d)$, and for which the source point is $A(a, b)$ and the destination point is $B(u, v)$. If we assume that the sweep list SW is ordered as described in **Step 4 (Sweep line management)** of the adapted Bentley-Ottmann algorithm depending on the y-coordinates of its edges, then the position of the edge $e(s, d)$ towards the edges from the sweep list can be described as follows:

(1) either the edge $e(s, d)$ is below sw_i in $\mathbb{R}^2$ and $e(s, d)$ is below all the other edges from the sweep list SW in $\mathbb{R}^2$. In addition, there are no other edges from the sweep list SW below e in $\mathbb{R}^2$. For visualizing the described situation see Figure 3.9, (a);

(2) either the edge $e(s, d)$ is above sw_i in $\mathbb{R}^2$ and $e(s, d)$ is above all the other edges from the sweep list SW in $\mathbb{R}^2$. Moreover, there are no other edges from the sweep list SW above e in $\mathbb{R}^2$. For a better understanding of this situation see Figure 3.9, (b);

(3) or finally, the edge $e(s, d)$ is above the edge sw_i from the sweep list SW in $\mathbb{R}^2$. In this case, it follows that the edge $e(s, d)$ is above all the edges which are below sw_i in $\mathbb{R}^2$

and that the edge $e(s, d)$ is below all the other edges from the sweep list SW in $\mathbb{R}^2$. For a graphical description of this situation, see Figure 3.9, (c).

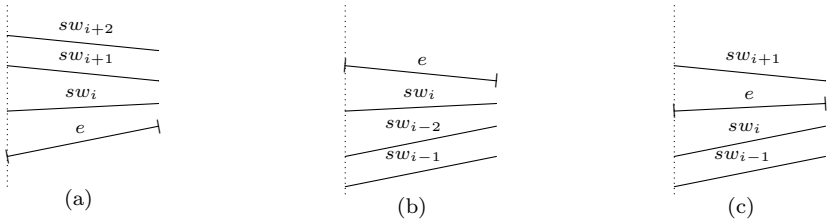


Figure 3.9: Position of an edge $e(s, d)$ from the ordered list of edges E towards the edges from the sweep list SW.

We now describe the algorithm `InsertSW`(e, SW), which inserts the current edge e from the ordered list of edges E in the sweep list SW as described in **Step 4 (Sweep line management)** of the adapted Bentley-Ottmann algorithm.

Algorithm 10 Insert an edge e from the ordered list of edges E in the sweep list SW in the right position as described in **Step 4** of the adapted Bentley-Ottmann algorithm: `InsertSW`(e, SW)

Input: $e(s, d)$ an edge in the 2-dimensional Euclidean plane $\mathbb{R}^2$,
SW the sweep list ordered as in **Step 4** of the adapted Bentley-Ottmann algorithm.
Output: SW the sweep list ordered as in **Step 4** of the Bentley-Ottmann algorithm, in which we insert e on the right position.

```
1: for i = 0 to length(SW) do
2:   if we find sw_i in SW such that source(e) = destination(sw_i) then
3:     insert e instead of sw_i in SW
4:   else
5:     take P ← source point of e
6:     take A ← source point of sw_i
7:     take B ← destination point of sw_i
8:     value =ComputeDet(A, B, P)
9:     if (value < 0) and (there are no edges in SW below e) then
10:      insert e before sw_i in SW                {e is below sw_i and above no edges}
11:    else if (value ≥ 0) and (there are no edges in SW above e) then
12:      insert e after sw_i in SW                 {e is above sw_i and below no edges}
13:    else
14:      insert e after sw_i in SW                 {e is above sw_i and below several edges}
15:    end if
16:  end if
17:  return SW.
18: end for
```

We now assume that we know the following algorithms: (i) the `FindIntersection`(e_1, e_2) algorithm for finding the intersection point of two edges e_1, e_2, if this intersection point exists, together with its auxiliary algorithm called `ComputeIntersection`(e_1, e_2); (ii) and the `InsertSW`(e, SW) algorithm for inserting an edge e from the ordered list of edges E into the sweep list SW on the right position, together with its auxiliary algorithm called `ComputeDet`(A, B, P). In the following, we describe the algorithm `SweepPlane`$(\mathcal{G}, V, E)$.

This algorithm operates on the projection $\mathcal{G}$ of a 3-dimensional graph, with V denoting its list of vertices and with E denoting its list of edges. The algorithm computes the intersection points among all the edges E of the projection $\mathcal{G}$ of the 3-dimensional graph and for each computed intersection point the pair of edges that contains it. The sweep line algorithm uses three auxiliary algorithms denoted `HandleCase1`, `HandleCase2` and `HandleCase3`. These three auxiliary algorithms manage the sweep list SW in different ways, depending on the position on which each edge e from the list of ordered edges E is inserted in the sweep list SW. We define a non-trivial position of the sweep list SW as a position of SW different from its first or its last position. This means that the i-th position of SW is non-trivial if $i \in \{1, ..., length(SW)-1\}$. We distinguish the following cases in the `SweepPlane` algorithm:

(1) If an edge e is inserted on the first position in the sweep list SW, then the `SweepPlane` algorithm calls the `HandleCase1` algorithm.

(2) If an edge e is inserted on the last position in the sweep list SW, then the `SweepPlane` algorithm calls the `HandleCase2` algorithm.

(3) If an edge e is inserted on a non-trivial position in the sweep list SW, then the `SweepPlane` algorithm calls the `HandleCase3` algorithm.

Algorithm 11 Perform the sweep-line algorithm on the projection $\mathcal{G}$ of a 3-dimensional graph, where V is its list of vertices and E is its list of edges. The sweep line algorithm computes the intersection points among all the edges from E, and for each intersection point, the pair of edges that contains it: `SweepPlane`$(\mathcal{G}, V, E)$

Input: $\mathcal{G}$ the projection of a 3-dimensional graph data structure $Graph(L_\epsilon)$,
V the list of vertices of $\mathcal{G}$ with $p_i(index, x_i, y_i)$ for each $p_i \in V$, where $i \in \mathbb{Z}$,
E the list of edges of $\mathcal{G}$ with $e_i(s_i, d_i)$ for each $e_i \in E$, where $s_i, d_i \in V$ and $i \in \mathbb{Z}$. The list of edges E is ordered as in **Step 1** of the adapted Bentley-Ottmann algorithm.
Output: I the list of intersection points among all the edges from E,
EI the list of pairs of edges that contain all the intersection points.

1: $E \leftarrow \{e_0, e_1, e_2, ..., e_n\}$ and $SW \leftarrow \{e_0, e_1\}$
2: $I \leftarrow \emptyset$ and $EI \leftarrow \emptyset$
3: **for** $i \leftarrow 2$ to $length(E)$ **do**
4: $p \leftarrow$ `InsertSW`(e_i, SW) {p is the position on which e_i is inserted in SW}
5: **if** $(p = 0)$ **then**
6: `HandleCase1` {e_i is inserted on the first position in SW}
7: **else if** $\big(p = length(SW)\big)$ **then**
8: `HandleCase2` {e_i is inserted on the last position in SW}
9: **else**
10: `HandleCase3` {e_i is inserted on a non-trivial position in SW}
11: **end if**
12: **end for**
13: **return** $\langle I, EI \rangle$

We now describe in details the three auxiliary algorithms `HandleCase1`, `HandleCase2` and `HandleCase3`, algorithms which are needed for the `SweepPlane` algorithm. These three auxiliary algorithms are designed based on the **Step 4 (Sweep line management)** of the adapted Bentley-Ottmann algorithm. We present the `HandleCase1` algorithm, which is called by the `SweepPlane` algorithm if the edge e is inserted on the first position in the sweep list SW. The `HandleCase1` algorithm proceeds in the following way:

- Whenever we insert an edge e from the ordered list of edges E on the first position in the sweep list SW, we test it for intersection only with its right neighbour, since in this case we know that the left neighbour of e does not exist.
- We call an intersection point of two edges degenerate if the two edges have a common destination point and the intersection point coincides with the common destination point of the two edges. Since we do not want to detect degenerate intersection points, we test e and its neighbour for intersection with the `FindIntersection` algorithm only if the two edges do not have the same destination point.
- If an intersection point is detected, then we insert it in the list of intersection points denoted with I. In addition, the corresponding pair of edges that contains the detected intersection point is inserted in the list of pairs of intersection edges denoted with EI.
- Moreover, whenever an intersection point P is reported for the pair of edges (e, f) from SW, we have to swap the order of the two edges in the sweep list.
- If an edge e and its neighbours have the same destination points, then we have to erase them from the sweep list SW to ensure the correctness of the algorithm. Therefore whenever we insert an edge in the sweep list we have to test it for common destination point against its neighbours. If we detect that an edge e and its neighbour have a common destination point, we erase the two edges from the sweep list SW. We mention that if two neighbouring edges with the same destination points are not erased from the sweep list SW, then the adapted Bentley-Ottmann algorithm does not detect all the intersection points.
- We notice that when we insert an edge e on the first position in the sweep list SW, we perform two different types of operations depending on whether e intersects its right neighbour or not. We distinguish the following two operations:
  - if the edge e intersects its right neighbour from the sweep list SW denoted with sw_1, then after reporting the intersection point, the two intersecting edges (e, sw_1) are swapped in the sweep list SW. After the swapping process, the edge e has a new right neighbour and it has a left neighbour that coincides with the original right neighbour of e, see Figure 3.10 for visualizing this situation. In this case, the edge e has to be tested for common destination point both with its left and with its right neighbour;
  - if the edge e and its original right neighbour from the sweep list SW denoted sw_1 do not intersect but they have the same destination point, then we erase the two edges from the sweep list, see Figure 3.11 for a graphical description of this situation.

0	1	2	...	n
e	sw_1	sw_2	...	$sw-n$

$\Longrightarrow$

0	1	2	...	n
sw_1	e	sw_2	...	sw_n

Figure 3.10: Insertion of an edge e from the ordered list of edges E on the first position of the sweep list SW with an intersection point detected. If the edge e intersects its right neighbour sw_1, then we report the computed intersection point and we swap the pair of edges (e, sw_1) in the sweep list SW. After the swapping process, the edge e is tested for common destination point with its left neighbour sw_1 and with its right neighbour sw_2.

We present the `HandleCase2` algorithm, which is called by the `SweepPlane` algorithm if the edge e is inserted on the last position in the sweep list SW. The `HandleCase2` algorithm proceeds similarly to the `HandleCase1` algorithm with the required modifications as follows:

0	1	2	...	n
e	sw_1	sw_2	...	$sw-n$

$\Longrightarrow$

0	1	2	...	n
e	sw_1	sw_2	...	sw_n

Figure 3.11: Insertion of an edge e from the ordered list of edges E on the first position of the sweep list SW with no intersection point detected. If the edge e does not intersect its right neighbour sw_1, then no intersection point is reported and no swapping process is performed in the sweep list SW. The edge e is tested for common destination point with its right neighbour sw_1.

Algorithm 12 Manage the sweep list SW in the algorithm `SweepPlane` when an edge e from the ordered list of edges E is inserted on the first position in SW : `HandleCase1`

Input: Same input as in the `SweepPlane` algorithm.
Output: I and EI as in the `SweepPlane` algorithm.

1: **if** $ycoord(sw_p) \neq ycoord(sw_{p+1})$ **then**
2: $v =$ `FindIntersection`(sw_p, sw_{p+1}) {exclude degenerate case}
3: **end if**
4: **if** $IsNotEmpty(v)$ **then**
5: insert the intersection point v to the list I
6: insert the pair of edges (sw_p, sw_{p+1}) to the list EI
7: $swap(sw_p, sw_{p+1})$ {intersection detected}
8: **if** $ycoord(sw_{p+1}) = ycoord(sw_{p+2})$ **then**
9: erase the edges sw_{p+1}, sw_{p+2} from SW {assure correctness}
10: **end if**
11: **end if**
12: **if** $ycoord(sw_p) = ycoord(sw_{p+1})$ **then**
13: erase the edges sw_p, sw_{p+1} from SW {assure correctness}
14: **end if**
15: **return** $\langle I, EI \rangle$

- Whenever we insert an edge e on the last position in the sweep list SW, we test it for intersection only with its left neighbour, since in this case we know that the right neighbour of the edge e does not exist.
- As in the case of the `HandleCase1` algorithm, since we do not want to detect degenerate intersection points, we always test e and its neighbour for intersection with the `FindIntersection` algorithm only if the two edges do not have the same destination point.
- If an intersection point is detected, then we insert it in the list of intersection points denoted with I. Moreover, the corresponding pair of edges that contains the detected intersection point is inserted in the list of pairs of intersection edges denoted with EI.
- As in the case of the `HandleCase1` algorithm, whenever we insert an edge e on the last position in the sweep list SW, we perform two different types of operations depending on whether e intersects its left neighbour or not. We distinguish the following two operations:
  - if the edge e intersects its left neighbour from the sweep list SW denoted with sw_{n-1}, then after reporting the intersection point, the two intersecting edges (sw_{n-1}, e) are swapped in the sweep list SW. After the swapping process, the edge e has a new left neighbour and it has a left neighbour which coincides with the original right neighbour of e, see Figure 3.12 for visualizing this situation. In this case, the edge e has to be tested for common destination point both with its left and with its right neighbour;
  - if the edge e and its original left neighbour from the sweep list SW denoted sw_{n-1} do not intersect but they have the same destination point, then we erase the two edges from the sweep list, see Figure 3.13 for a graphical description of this situation.

0	...	$n-2$	$n-1$	n
sw_0	...	sw_{n-2}	sw_{n-1}	e

$\Longrightarrow$

0	...	$n-2$	$n-1$	n
sw_0	...	sw_{n-2}	e	sw_{n-1}

Figure 3.12: Insertion of an edge e from the ordered list of edges E on the last position of the sweep list SW with an intersection point detected. If the edge e intersects its left neighbour sw_{n-1}, then we report the computed intersection point and we swap the pair of edges (sw_{n-1}, e) in the sweep list SW. After the swapping process, the edge e is tested for common destination point with its left neighbour sw_{n-2} and with its right neighbour sw_{n-1}.

0	...	$n-2$	$n-1$	n
sw_0	...	sw_{n-2}	sw_{n-1}	e

$\Longrightarrow$

0	...	$n-2$	$n-1$	n
sw_0	...	sw_{n-2}	sw_{n-1}	e

Figure 3.13: Insertion of an edge e from the ordered list of edges E on the last position of the sweep list SW with no intersection point detected. If the edge e does not intersect its left neighbour sw_{n-1}, then no intersection point is reported and no swapping process is performed in the sweep list SW. The edge e is tested for common destination point with its left neighbour sw_{n-1}.

We present the `HandleCase3` algorithm, which is called by the `SweepPlane` algorithm if the edge e is inserted on a non-trivial position in the sweep list SW, i.e. the edge e is

Algorithm 13 Manage the sweep list SW in the algorithm `SweepPlane` when an edge e from the ordered list of edges E is inserted on the last position in SW : `HandleCase2`

Input: Same input as in the `SweepPlane` algorithm.
Output: I and EI as in the `SweepPlane` algorithm.

```
1: if ycoord(sw_{p-1}) ≠ ycoord(sw_p) then
2:    v = FindIntersection(sw_{p-1}, sw_p)                {exclude degenerate case}
3: end if
4: if IsNotEmpty(v) then
5:    insert the intersection point v to the list I
6:    insert the pair of edges (sw_{p-1}, sw_p) to the list EI
7:    swap(sw_{p-1}, sw_p)                                {intersection detected}
8:    if ycoord(sw_{p-1}) = ycoord(sw_{p-2}) then
9:       erase the edges sw_{p-1}, sw_{p-2} from SW       {assure correctness}
10:   end if
11: end if
12: if ycoord(sw_{p-1}) = ycoord(sw_p) then
13:   erase the edges sw_{p-1}, sw_p from SW              {assure correctness}
14: end if
15: return ⟨I, EI⟩
```

inserted on a position different from the first or the last position in the sweep list SW. The `HandleCase3` algorithm proceeds similarly to the `HandleCase1` and to the `HandleCase2` algorithm, with the required modifications as follows:

- Whenever we insert an edge e on a non-trivial position in the sweep list SW, we want to test it for intersection with both its left and its right neighbour. We apply the same strategies as in the case of `HandleCase1` and of `HandleCase2` algorithms. In the `HandleCase3` algorithm we distinguish between different cases as follows:
  - Firstly, we treat the case of the edge e and its left neighbour. If the two edges have different destination points, then we test them for intersection. If an intersection is detected, then we report it together with the pair of edges that contains it. In addition, we swap the order of the edges of intersection in the sweep list SW. If the edge e has common destination points with its neighbours, then we erase the edges from the sweep list SW, see Figure 3.14 for a graphical description of this situation.
  - Secondly, we consider the case of e and its right neighbour. If the two edges have different destination points, then we test them for intersection. If an intersection point is detected, then we report it together with the pair of edges that contains it, and we swap the order of edges of intersection in the sweep list SW. Moreover, if the edge e has common destination points with its neighbours, then we erase the edges from the sweep list SW, see Figure 3.15.
  - Finally, we handle the case in which the edge e does not intersect any of its neighbours. In this case, we only have to test the edge e against its left and right neighbour for common destination points. If the edge e has common destination point with its left or its right neighbour, then the edges are removed from the sweep list SW, see Figure 3.16.
  - Whenever two consecutive edges from the sequence

$$\{\textit{left-neighbour}(e), e, \textit{right-neighbour}(e)\}$$

have a common destination point, they are erased from the sweep list. Since we have considered only "small" edges, the edge e cannot intersect in the same time both its right and its left neighbour, so this case was not included in the treatment of the adapted Bentley-Ottmann algorithm.

...	$i-2$	$i-1$	i	$i+1$	...
...	sw_{i-2}	sw_{i-1}	e	sw_{i+1}	...

$\Rightarrow$

...	$i-2$	$i-1$	i	$i+1$	...
...	sw_{i-2}	e	sw_{i-1}	sw_{i+1}	...

Figure 3.14: Insertion of an edge e from the ordered list of edges E on a non-trivial position of the sweep list SW with an intersection point detected between e an its left neighbour. If the edge e intersects its left neighbour sw_{i-1}, then we report the computed intersection point and we swap the pair of edges (sw_{i-1}, e) in the sweep list SW. After the swapping process, the edge e is tested for common destination point with its left neighbour sw_{i-2} and with its right neighbour sw_{i-1}.

...	$i-1$	i	$i+1$	$i+2$	...
...	sw_{i-1}	e	sw_{i+1}	sw_{i+2}	...

$\Rightarrow$

...	$i-1$	i	$i+1$	$i+2$	...
...	sw_{i-1}	sw_{i+1}	e	sw_{i+2}	...

Figure 3.15: Insertion of an edge e from the ordered list of edges E on a non-trivial position of the sweep list SW with an intersection point detected between e an its right neighbour. If the edge e intersects its right neighbour sw_{i+1}, then we report the computed intersection point and we swap the pair of edges (e, sw_{i+1}) in the sweep list SW. After the swapping process, the edge e is tested for common destination point with its left neighbour sw_{i+1} and with its right neighbour sw_{i+2}.

...	$i-1$	i	$i+1$	...
...	sw_{i-1}	e	sw_{i+1}	...

$\Rightarrow$

...	$i-1$	i	$i+1$	...
...	sw_{i-1}	e	sw_{i+1}	...

Figure 3.16: Insertion of an edge e from the ordered list of edges E on a non-trivial position of the sweep list SW with no detected intersection point. If the edge e does not intersect neither its left neighbour sw_{i-1} nor its right neighbour sw_{i+1}, then no intersection point is reported and no swapping process is performed in the sweep list SW. The edge e is tested for common destination point with its left neighbour sw_{i-1} and with its right neighbour sw_{i+1}.

Algorithm for arranging the pair of intersection edges. As output to the `SweepPlane` algorithm we obtain the list of intersection points denoted with I, and the list of pairs of edges that contain all the intersection points from I denoted with EI. For instance, if P_i is the i-th intersection point from I, then the i-th pair of edges (e_i, f_i) from EI represents the pair of intersection edges that contains the intersection point P_i, i.e. $P_i = e_i \cap f_i$. We need an algorithm that orders each pair of intersection edges. We say that the pair of intersection edges (e_1, e_2) that contains the intersection point P is ordered if e_1 is under e_2 in $\mathbb{R}^3$. We give the main idea of the algorithm for arranging the pair of intersecting edges, see Figure 3.7 for a consistent graphical description.

Algorithm 14 Manage the sweep list SW in the algorithm `SweepPlane` when an edge e from the ordered list of edges E is inserted on a non-trivial position in SW, i.e. e is inserted in SW on a position different from the first or the last position of SW : `HandleCase3`

Input: Same input as in the `SweepPlane` algorithm.
Output: I and EI as in the `SweepPlane` algorithm.

1: **if** $ycoord(sw_{p-1}) \neq ycoord(sw_p)$ **then**
2: $v =$ `FindIntersection`(sw_{p-1}, sw_p) {exclude degenerate case}
3: **end if**
4: **if** $IsNotEmpty(v)$ **then**
5: insert the intersection point v to the list I
6: insert the pair of edges (sw_{p-1}, sw_p) to the list EI
7: $swap(sw_{p-1}, sw_p)$ {intersection detected}
8: **if** $ycoord(sw_{p-1}) = ycoord(sw_{p-2})$ **then**
9: erase the edges sw_{p-1}, sw_{p-2} from SW {assure correctness}
10: **end if**
11: **end if**
12: **if** $ycoord(sw_p) \neq ycoord(sw_{p+1})$ **then**
13: $v =$ `FindIntersection`(sw_p, sw_{p+1}) {exclude degenerate case}
14: **end if**
15: **if** $IsNotEmpty(v)$ **then**
16: insert the intersection point v to the list I
17: insert the pair of edges (sw_p, sw_{p+1}) to the list EI
18: $swap(sw_p, sw_{p+1})$ {intersection detected}
19: **if** $ycoord(sw_{p+1}) = ycoord(sw_{p+2})$ **then**
20: erase the edges sw_{p+1}, sw_{p+2} from SW {assure correctness}
21: **end if**
22: **end if**
23: **if** $ycoord(sw_{p-1}) = ycoord(sw_p) = ycoord(sw_{p+1})$ **then**
24: erase the edges sw_{p-1}, sw_p, sw_{p+1} from SW {assure correctness}
25: **end if**
26: **if** $ycoord(sw_{p-1}) \neq ycoord(sw_p)$ and $ycoord(sw_p) = ycoord(sw_{p+1})$ **then**
27: erase the edges sw_p, sw_{p+1} from SW {assure correctness}
28: **end if**
29: **if** $ycoord(sw_{p-1}) = ycoord(sw_p)$ and $ycoord(sw_p) \neq ycoord(sw_{p+1})$ **then**
30: erase the edges sw_{p-1}, sw_p from SW {assure correctness}
31: **end if**
32: **return** $\langle I, EI \rangle$

- For both edges e_1, e_2 we first extract the coordinates of their source and of their destination points in $\mathbb{R}^2$. We denote the source and the destination points of e_1 with the points $A_1(a_1, b_1), B_1(u_1, v_1)$, and the source and destination points of e_2 with the points $A_2(a_2, b_2), B_2(u_2, v_2)$. Assuming that the two edges e_1, e_2 do intersect, we need to compute the coordinates of their intersection point $P(x, y)$ in $\mathbb{R}^2$ with the `ComputeIntersection` algorithm.

- We compute the equations of the two edges e_1, e_2 with the `EqnLine` algorithm. We denote the equation of e_1 with $L_1(x, y)$, and the equation of e_2 with $L_2(x, y)$.

- We remember that e_1, e_2 are projections of the edges of a 3-dimensional $Graph(L_\epsilon)$ from $\mathbb{R}^3$ that we denote with $e_1^{'}, e_2^{'}$. For both $e_1^{'}, e_2^{'}$ we extract the coordinates of their source and of their destination points in $\mathbb{R}^3$, which we denote with the points $A_1^{'}(a_1, b_1, c_1), B_1^{'}(u_1, v_1, w_1)$ for $e_1^{'}$, and with the points $A_2^{'}(a_2, b_2, c_2), B_2^{'}(u_2, v_2, w_2)$ for $e_2^{'}$. The intersection point $P(x, y) = e_1 \cap e_2$ from $\mathbb{R}^2$ has two corresponding points in $\mathbb{R}^3$:

  - a point $P_1^{'}(x, y, z_1)$ that lies on $e_1^{'}$ with the special property that P and $P_1^{'}$ divide e_1 and $e_1^{'}$ in the same proportion factor α_1 since e_1 is the projection of $e_1^{'}$, i.e. e_1 and $e_1^{'}$ are parallel;

  - and a point $P_2^{'}(x, y, z_2)$ that lies on $e_2^{'}$ with the special property that P and $P_2^{'}$ divide e_2 and $e_2^{'}$ in the same proportion factor α_2 since e_2 is the projection of $e_2^{'}$, i.e. e_2 and $e_2^{'}$ are parallel.

We notice that the points $P_1^{'}$ and $P_2^{'}$ in $\mathbb{R}^3$ differ only by their z-coordinate. In fact, if $z_1 < z_2$, then $e_1^{'}$ is under $e_2^{'}$ in the 3-dimensional Euclidean space $\mathbb{R}^3$. Since e_1, e_2 are the projections of $e_1^{'}, e_2^{'}$, if $z_1 < z_2$, then we say that e_1 is under e_2 in $\mathbb{R}^3$, otherwise we say that e_2 is under e_1 in $\mathbb{R}^3$. In this way, the criteria for ordering the pair of intersection edges (e_1, e_2) of the intersection point $P(x, y)$ reduces to computing the corresponding coordinates z_1, z_2 in $\mathbb{R}^3$ as described above. For computing the coordinates z_1, z_2 we need an algorithm that computes the proportion factors α_1, α_2. For instance, we compute α_1. For computing α_2 we proceed in the same way. We consider the points $A_1(a_1, b_1), B_1(u_1, v_1)$ in $\mathbb{R}^2$ and the equation of the line $L_2(x, y)$ computed as before. We compute α_1 from the equation $\alpha_1 \cdot L_2(A_1) + (1 - \alpha_1) \cdot L_2(B1) = 0$. We get $\alpha_1 = \dfrac{L_2(B_1)}{L_2(B_1) - L_2(A_1)}$, as described in the `GetAlpha` algorithm.

We continue with computing the z_1 coordinate of the point $P_1^{'}(x, y, z_1)$. We assume that we computed α_1 with the `GetAlpha` algorithm. We consider the edge e_1 as the projection in $\mathbb{R}^2$ of the edge $e_1^{'}$ from $\mathbb{R}^3$. In this case the coordinates of the source point of the edge $e_1^{'}$ in the 3-dimensional Euclidean space are given by $A_1^{'}(a_1, b_1, c_1)$. In the same way, the coordinates of the destination point of the $e_1^{'}$ in $\mathbb{R}^3$ are given by $B_1^{'}(u_1, v_1, w_1)$. Under these assumptions, we compute $z_1 = \alpha_1 \cdot c_1 + (1 - \alpha_1) \cdot w_1$ as described in the `GetZCoordinate` algorithm. We recall that in this thesis we use the following terminology: we call the algorithm that computes all the intersection points of a set of edges in the 2-dimensional Euclidean space as described in the `SweepPlane` algorithm and that computes the pair of ordered edges of intersections as described in the `GetZCoordinate` algorithm, an adapted version of the Bentley-Ottmann algorithm.

Algorithm 15 First auxiliary algorithm for arranging the pair of intersection edges (e, f) (with e determined by $A, B \in \mathbb{R}^2$ and $f(s, d)$ the projection of the edge $f^{'}$ from $\mathbb{R}^3$). In addition, $(e, f) \in EI$ with the intersection point $P = e \cap f$, returned by the `SweepPlane` algorithm: `GetAlpha`(A, B, f, P)

Input: $A(a, b), B(u, v)$ in $\mathbb{R}^2$, points that determine the edge $e \in \mathbb{R}^2$,
which is the projection of the edge $e^{'} \in \mathbb{R}^3$ from $Graph(L_\epsilon)$,
$f(s, d) \in \mathbb{R}^2$ an edge given with its source index s and with its destination index d,
which is the projection of the edge $f^{'} \in \mathbb{R}^3$ from $Graph(L_\epsilon)$,
$P(x, y)$ the intersection of the pair of edges (e, f) and
$P^{'}(x, y, z)$ lies on $e^{'}$ in $\mathbb{R}^3$.

Output: $\alpha \in \mathbb{Z}_{>0}$,
where α is the proportion factor of the edges $e, e^{'}$ with the property that the points P and $P^{'}$ divide the edges e and $e^{'}$ in the same proportion factor α.

1: $m = xcoord(s), n = ycoord(s)$
2: $p = xcoord(d), q = ycoord(d)$
3: consider $M(m, n)$ and $N(p, q)$
4: $(\beta, \gamma, \delta) =$`EqnLine`$(M, N)$
5: $L(x, y) = \beta x + \gamma y + \delta$
6: $value_1 =$ `EvalAtPointEqnLine`$\big(L(x, y), A(a, b)\big)$
7: $value_2 =$ `EvalAtPointEqnLine`$\big(L(x, y), B(u, v)\big)$
8: **if** $(value_2 - value_1 \neq 0)$ **then**
9: **return** $\alpha = \dfrac{v_2}{v_2 - v_1}$
10: **else**
11: **print** `The proportion factor for the two edges is undefined!`
12: **end if**

Algorithm 16 Second auxiliary algorithm for arranging the pair of intersection edges (e, f) (with e determined by $A, B \in \mathbb{R}^2$ and $f(s, d)$ the projection of the edge $f^{'}$ from $\mathbb{R}^3$). In addition, $(e, f) \in EI$ with the intersection point $P = e \cap f$, returned by the `SweepPlane` algorithm: `GetZCoordinate`(A, B, f, P)

Input: $A(a, b), B(u, v)$ in $\mathbb{R}^2$, points that determine the edge $e \in \mathbb{R}^2$,
which is the projection of the edge $e^{'} \in \mathbb{R}^3$ from $Graph(L_\epsilon)$,
edge that is determined by $A^{'}(a, b, c) \in \mathbb{R}^3$ and $B^{'}(u, v, w) \in \mathbb{R}^3$,
$f(s, d) \in \mathbb{R}^2$ and edge given with its source index s and with its destination index d,
which is the projection of the edge $f^{'} \in \mathbb{R}^3$ from $Graph(L_\epsilon)$,
$P(x, y)$ the intersection of the pair of edges (e, f) and
$P^{'}(x, y, z)$ lies on $e^{'}$ in $\mathbb{R}^3$.

Output: $z \in \mathbb{R} \setminus \{0\}$,
where z is the Euclidean coordinate of $P^{'}(x, y, z)$ in $\mathbb{R}^3$, as described in Figure 3.7.

1: $\alpha =$ `GetAlpha`(A, B, f, P)
2: **return** $z = \alpha \cdot c + (1 - \alpha) \cdot w$

We assume that we know the `GetAlpha` and the `GetZCoordinate` algorithms. For a pair of intersection edges (e_1, e_2) of the intersection point $P(x, y)$ in $\mathbb{R}^2$, we compute the corresponding coordinate z_1, z_2 in $\mathbb{R}^3$ with the `GetAlpha` and the `GetZCoordinate` algorithms. If $z_1 < z_2$, then e_1 is under e_2 in $\mathbb{R}^3$ and thus the pair (e_1, e_2) is ordered. If $z_1 > z_2$, then e_1 is over e_2 in $\mathbb{R}^3$. In this case, we swap the two edges in the pair of intersection edges (e_1, e_2) and we obtain the new ordered pair (e_2, e_1). After applying the `SweepPlane` algorithm on

the projection $\mathcal{G}$ of a 3-dimensional graph $Graph(L_\epsilon)$ returned by Axel, we always perform the `ArrangeEdgesIntersect` algorithm on the list of pairs of intersection edges denoted with EI, which is returned by the `SweepPlane` algorithm. Consequently, we order the pairs of intersection edges from EI.

Algorithm 17 Arrange the pair of intersection edges from the list EI returned by the `SweepPlane` algorithm: `ArrangeEdgesIntersect`(EI)

Input: EI as returned by the `SweepPlane` algorithm and representing the pair of intersection edges of all the intersection points detected for all the edges of a 3-dimensional graph.

Output: The ordered list EI,
where for each pair $(e_i, f_i) \in EI$, e_i is under f_i in the Euclidean space $\mathbb{R}^3$.

1: **for all** $i = 0$ to $length(EI)$ **do**
2: consider the pair of intersection edges (e_i, f_i) from EI
3: consider A_1, B_1 the source and destination points of e_i
4: consider A_2, B_2 the source and destination points of f_i
5: $z_1 =$ `GetZCoordinate`(A_1, B_1, f_i)
6: $z_2 =$ `GetZCoordinate`(A_2, B_2, e_i)
7: **if** $(z_1 > z_2)$ **then**
8: $swap(e_i, f_i)$
9: **end if**
10: **return** EI
11: **end for**

3.3.2 Combinatorial Algorithms from Knot Theory

In this subsection, we assume that for each singularity of a plane complex algebraic curve we computed its approximate differentiable link L_ϵ as a 3-dimensional graph data structure $Graph(L_\epsilon)$ by using the subdivision algorithms of the Axel algebraic geometric modeler as described in Section 3.2. The $Graph(L_\epsilon) = \langle V, E \rangle$ is defined as a pair between its set of vertices V (or points in $\mathbb{R}^3$) together with their Euclidean coordinates and between its set of edges E. In addition, we assume that for the projection of the 3-dimensional graph $Graph(L_\epsilon)$ we computed the set of intersection points I and the set of pairs of edges that contain all the intersection points using the algorithm `SweepPlane`$\big(Graph(L_\epsilon), V, E\big)$. Moreover, we assume that we ordered the pairs of edges from the set EI using the algorithm `ArrangeEdgesIntersect`(EI).

We next want to compute more combinatorial information on the special projection of the 3-dimensional $Graph(L_\epsilon)$ in $\mathbb{R}^2$, special projection that we call the diagram of the approximate link as introduced in Chapter 2, Subsection 2.3.3. For the rest of this thesis we denote the diagram of the approximate link with $D\big(Graph(L_\epsilon)\big)$. This is the main reason for which we call the algorithms presented in this subsection combinatorial algorithms from knot theory. We present the following combinatorial algorithms for computing the diagram $D\big(Graph(L_\epsilon)\big)$ of a 3-dimensional graph $Graph(L_\epsilon)$, graph that represents the approximation of an approximate link L_ϵ :

1. an algorithm for computing the knot components of the approximate link, which represents the knot components of the diagram $D\big(Graph(L_\epsilon)\big)$;

2. an algorithm for computing the arcs of the diagram $D\big(Graph(L_\epsilon)\big)$, as defined in Chapter 2, Subsection 2.3.3.

Combinatorial Algorithms for Detecting the Knot Components of an Approximate Link

First of all we present the combinatorial algorithm for computing the knot components of an approximate link L_ϵ of a plane curve singularity, approximate link that is represented as a 3-dimensional graph $Graph(L_\epsilon)$.

We mention that if the approximate link L_ϵ of a singularity has several knot components, then the $Graph(L_\epsilon)$ data structure contains also several cycles represented by piecewise linear knots that are the approximations of the differentiable knots. For computing the diagram of the approximate link, we certainly need to compute all the piecewise linear knot components. We observe that the piecewise linear knot components of the approximate link represent the cycles of the 3-dimensional graph $Graph(L_\epsilon)$. From now on in our study we consider only piecewise linear knot components, which we simply call knot components. Thus in this paragraph we present an algorithm for constructing the knot components of the diagram of an approximate link from the projection of the 3-dimensional graph $Graph(L_\epsilon)$. The algorithm also returns the total number of knot components in the diagram.

We consider the set of ordered edges E from $Graph(L_\epsilon)$ as in Subsection 3.3.1. It follows that the edges from the set E are ordered according to the ordering criteria from **Step 1** of the adapted Bentley-Ottmann algorithm from Subsection 3.3.1. We denote a positive edge in $\mathbb{R}^2$ satisfying the relation $xcoord(s) < xcoord(y)$ with $e(s,d)$, and we denote its corresponding negative edge satisfying the relation $xcoord(s) > xcoord(y)$ with $-e(d,s)$. We notice that the positive edges are oriented from left to right, while the negative ones are oriented from right to left, see Figure 3.17.

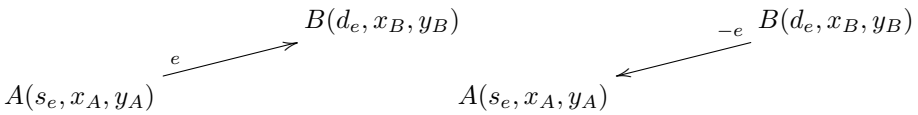


Figure 3.17: Orientation for a positive edge e and for its corresponding negative edge $-e$ in a 3-dimensional graph data structure.

We denote the knot components of an approximate link with $K_j, j \in \mathbb{N}$. All the knot components K_j of a link must satisfy the following two properties:

- **Property 1:** for each edge $e_k(s_k, d_k) \in K_j$ there exists an edge $e_{k+1}(s_{k+1}, d_{k+1}) \in K_j$ with $d_k = s_{k+1}$; in this case, we call e_{k+1} the right consecutive edge of e_k, and we call the sequence $\{e_k, e_{k+1}\}$ a suitable sequence of two consecutive edges in a knot component.

- **Property 2:** for each $K_j = \{e_0(s_0, d_0), ..., e_n(s_n, d_n)\}$, with $j \in \mathbb{N}$: $d_n = s_0$. It follows that the source index of the first edge in a knot component coincides with the destination index of the last edge in the same knot component. This property assures that the knot component is always a circuit in the graph.

As we mentioned before, we need an algorithm that constructs all the knot components $K_j, j \in \mathbb{N}$ with the two properties introduced in the paragraph above. As opposed to the list of ordered edges E that contains only positive edges oriented from left to right, we notice that each list of knot components K_j contains both positive and negative edges. We present the way in which the first knot component of a link can be computed from the projection of the 3-dimensional $Graph(L_\epsilon)$ data structure. 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 the list of ordered edges E

that has a common index, either source or destination, with d_0. If we find such an edge $e_n(d_0, d_n) \in E$, 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. In this case, we notice that we need to swap the source and the destination index points of the positive edge $e_n(s_n, d_0)$ to obtain its negative corresponding edge $-e_n(d_0, s_n)$, which is in fact inserted in K_0. We call the edge e_n the right consecutive edge of e_0. After we insert e_n in the list of knot component K_0, we erase it from the list of ordered edges E. We mention that we always find such an edge e_n in E for the edge e_0, because each *index* such as d_0 appears two times in E. We continue with inserting edges from E in K_0 in the same manner until the *destination* of an inserted edge coincide with the source s_0 of the first edge e_0 from K_0. We apply the same strategy to constructs all the other knot components K_j of the diagram $D\big(Graph(L_\epsilon)\big)$ until the list of ordered edges E is empty, increasing j each time a new knot component starts being constructed. At the end of the algorithm, the index $j+1$ returns the total number of knot components of $D\big(Graph(L_\epsilon)\big)$. We notice that all the knot components that are constructed from the list (or set) of ordered edges E have always a counterclockwise orientation. In Example 22 we show the way in which the combinatorial algorithm for computing the knot components of an approximate link L_ϵ represented as a 3-dimensional graph $Graph(L_\epsilon)$ behaves on the list of edges of $Graph(L_\epsilon)$.

Example 22. We consider $E = \{e_0, e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8, e_9, e_{10}, e_{11}\}$ as in Figure 3.18. We assume that E represents the set of ordered edges of the 3-dimensional graph data structure $Graph(L_\epsilon) = \langle V, E\rangle$, which approximates the ϵ-link L_ϵ of a plane curve singularity. We notice that the set of vertices of the graph $Graph(L_\epsilon)$ is not explicitly given as input to the problem. The combinatorial algorithm for constructing the knot components of L_ϵ consists of the following two steps:

- **Step 1:** We initialize the first knot component with $K_0 = \{e_0\}$. We continue with inserting edges from the ordered list of edges E into the knot component K_0 into the right direction (either from left to right for the positive edges or from right to left for the negative edges) until two consecutive edges have the same source index. While inserting edges from E in K_0, we erase them from E. We obtain:

$$E = \{\cancel{e_0}, \cancel{e_1}, e_2, e_3, \cancel{e_4}, \cancel{e_5}, e_6, e_7, \cancel{e_8}, e_9, \cancel{e_{10}}, e_{11}\},$$

$$K_0 = \{e_0, e_4, e_{10}, -e_8, -e_5, -e_1\}.$$

- **Step 2:** We initialize $K_1 = \{e_2\}$. We proceed in the same way as in **Step 2** and we obtain:

$$E = \{\cancel{e_2}, \cancel{e_3}, \cancel{e_6}, \cancel{e_7}, \cancel{e_9}, \cancel{e_{11}}\},$$

$$K_1 = \{e_2, e_6, e_{11}, -e_9, -e_7, -e_3\}.$$

  The algorithm terminates since we notice that at this step $E = \emptyset$. The algorithm also returns the number of knot components in L_ϵ, which in this case is 2.

For a graphical description of the way the combinatorial algorithm for computing the knot components of an approximate link works on the ordered list of edges E from this example see Figure 3.18. We now describe the `CreateKnots` algorithm.

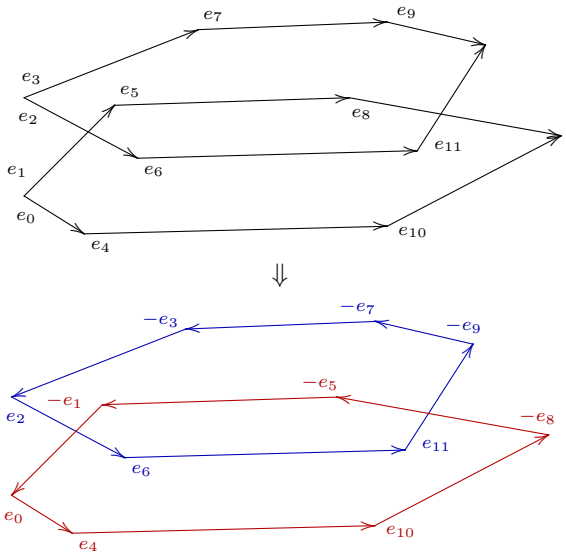


Figure 3.18: Creating the knot components of an approximate link represented as a 3-dimensional graph $Graph(L_\epsilon)$.

Algorithm 18 Compute the knot components of an approximate link L_ϵ represented as a 3-dimensional graph $Graph(L_\epsilon) = \langle V, E\rangle$ with V being its set of vertices and E being its set of edges: `CreateKnots`$\big(Graph(L_\epsilon), V, E\big)$

Input: G the projection of the 3-dimensional graph $Graph(L_\epsilon)$ that approximates L_ϵ,
V the set of points of G with $p_i(index, x_i, y_i) \in V$, and $i \in \mathbb{N}$,
E the set of ordered edges as in **Step 1** of the adapted Bentley-Ottmann algorithm with $e_i(s_i, d_i), s_i, d_i \in V$, and $i \in \mathbb{N}$.

Output: $K_j, j \in \mathbb{N}$ lists of ordered edges from E and $count \in \mathbb{N}$,
where $K_j, j \in \mathbb{N}$ represent all the knot components of L_ϵ constructed from the list of ordered edges E. Each $K_j, j \in \mathbb{N}$ satisfies **Property 1** and **Property 2**. Moreover, $count + 1$ represents the total number of knot components of L_ϵ.

1: $count \leftarrow -1$
2: **while** $IsNotEmpty(E)$ **do**
3: $count++$
4: consider $K_{count} \leftarrow \emptyset$ a knot component of the approximate link L_ϵ
5: insert e_0 in K_{count}
6: erase e_0 from E
7: **repeat**
8: consider e the last edge in K_{count}
9: **for all** $i \leftarrow 0$ to $length(E)$ **do**
10: find e_i the right consecutive edge of e in K_{count} with a common index (either source or destination) with $destination(e)$
11: **if** $destination(e) = source(e_i)$ **then**
12: insert e_i after e in K_{count}
13: erase e_i from E
14: **end if**
15: **if** $destination(e) = destination(e_i)$ **then**
16: $swap\big(source(e_i), destination(e_i)\big)$
17: insert e_i after e in K_{count}
18: erase e_i from E
19: **end if**
20: **end for**
21: **until** two suitable consecutive edges have the same source points
22: **end while**
23: **return** $\big\langle\{K_j, j \in \mathbb{N}\}, count\big\rangle$

Combinatorial Algorithms for Detecting the Arcs in the Diagram of a Link

We now present the algorithm for computing the arcs of the diagram $D\big(Graph(L_\epsilon)\big)$ of a 3-dimensional graph $Graph(L_\epsilon)$, graph that represents the approximation of the ϵ-link L_ϵ. We recall from Chapter 2, Subsection 3.2 that for the diagram of a link we can always distinguish between the type of its crossings, i.e. we distinguish between a lefthanded and a righthanded crossing. We say that a crossing is lefthanded if the underpass traffic goes from left to right and we say that a crossing is righthanded if the underpass traffic goes from right to left. We denote a lefthanded crossing with -1 (or sometimes with LH) and a righthanded crossing with $+1$ (or sometimes with RH), see Figure 3.19.

Moreover, we define the notion of an arc in the diagram of a link. An arc is the part of a diagram between two undercrossings. We now describe the combinatorial algorithm for constructing the arcs for each knot component of a 3-dimensional graph representing the approximation of an ϵ-link. The algorithm also decides the type of crossings (righthanded

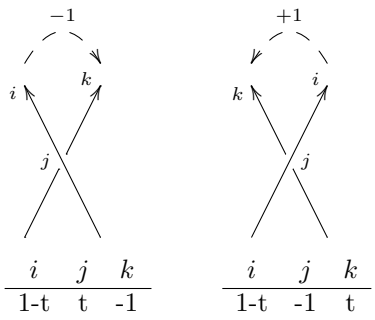


Figure 3.19: Type of crossings in a diagram: lefthanded crossing (denoted with -1) and righthanded crossing (denoted with $+1$) together with their corresponding arcs labelling.

or lefthanded) for each knot component. For constructing the arcs, we consider the set of ordered edges E as in the algorithm `CreateKnots` and as in **Step 1** of the adapted Bentley-Ottmann algorithm presented in Subsection 3.3.1. This algorithm operates on the output of the algorithm `SweepPlane` and on the output of the algorithm `CreateKnots`, i.e. it operates on the list of intersection points I together with the list of ordered pairs of intersection edges EI, and it operates on the the lists of edges $K_j, j \in \mathbb{N}$, which represent all the knot components in the diagram of the ϵ-link. The key point of this combinatorial algorithm is to search in all the knot components $K_j, j \in \mathbb{N}$ all the undergoing edges from the list of ordered pairs of edges EI and to split these undergoing edges in two parts. For instance in Figure 3.20, we consider a diagram of the trefoil knot and we compute the arcs of this diagram. We assume that for the list of ordered edges

$$E = \{e_0, ..., e_n, e_m, ..., e_l, e_k..., e_t, e_s, ..., e_{last}\},$$

we compute the following outputs with the algorithm `SweepPlane` and with the algorithm `CreateKnots`:

$$I = \{(x_1, y_1), (x_2, y_2), (x_3, y_3)\},$$

$$EI = \{(-e_n, e_m), (e_l, e_k), (e_s, -e_t)\},$$

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

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 the following modified knot component denoted with $K_0^{'}$:

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

From the definition of an arc, it follows that an arc contains the list of edges from a modified knot component $K_j^{'}, j \in \mathbb{N}$ starting with an edge of type $e_j^u, j \in \mathbb{N}$ from $K_j^{'}$ and ending with the next consecutive edge of type $e_k^d, k \in \mathbb{N}$ from $K_j^{'}$. While we insert edges from a modified knot component $K_j^{'}$ into the list of edges representing the corresponding arc we erase them from $K_j^{'}$. Thus in our example from Figure 3.20, from the modified knot component $K_0^{'}$ we compute the following three arcs until $K_0^{'}$ is empty:

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

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

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

$$arc_1 = \{e_l^u, .., -e_t, .., -e_n^d\},$$

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

$$arc_2 = \{e_n^u, .., -e_1, e_0, .., e_k, .., e_s^d\},$$

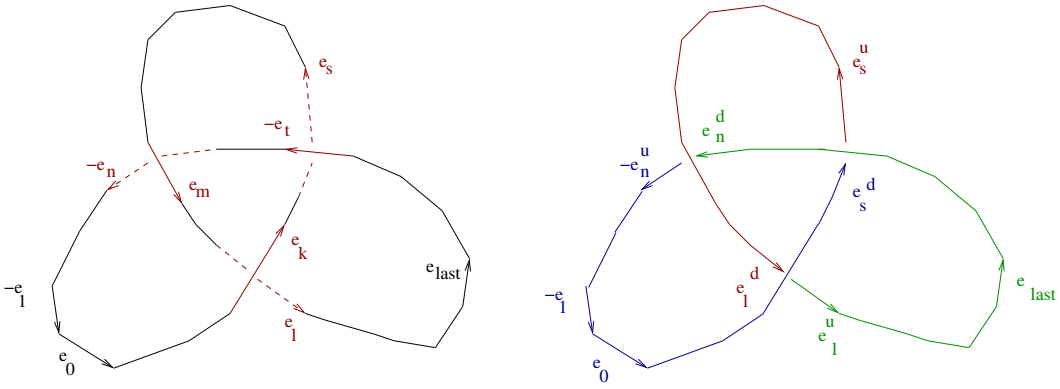


Figure 3.20: Creating the arcs of a trefoil knot diagram.

We describe the algorithm `CreateArcs`, which takes as input the following data:

(1) the list of intersection points I among all the edges of a 3-dimensional graph data structure $Graph(L_\epsilon)$ representing the approximation of an ϵ-link L_ϵ;

(2) the list of ordered pairs of intersection edges EI containing all the intersection points from I;

(3) the lists of edges $K_j, j \in \mathbb{N}$ representing all the knot components of the ϵ-link L_ϵ;

(4) the total number of knot components in the ϵ-link L_ϵ denoted with $count + 1$, with $j \in \mathbb{N}$. We mention that the list I and the list EI are computed with the `SweepPlane` algorithm, whereas the lists $K_{j,j\in\mathbb{N}}$ are computed with the `CreateKnots` algorithm.

The algorithm `CreateArcs` computes as output the lists of edges representing all the arcs for the diagram $D\big(Graph(L_\epsilon)\big)$ of the ϵ-link L_ϵ.

Algorithm 19 Create the arcs in the diagram of an ϵ-link L_ϵ approximated by a 3-dimensional graph $Graph(L_\epsilon) = \langle V, E\rangle$ with I being the list of intersection points among all the edges from E, with EI being the list of ordered pairs of edges containing all the intersection points from I. Moreover the lists $K_j, j \in \mathbb{N}$ represent all the knot components in L_ϵ and $count + 1$ represents the total number of knot components in L_ϵ. The algorithm is denoted as follows: `CreateArcs`$\big(\langle I, EI\rangle, \langle\{K_j, j \in \mathbb{N}\}, count\rangle\big)$

Input: I and EI as computed by the `SweepPlane` algorithm,
$K_j, j \in \mathbb{N}$ and *count* as computed by the `CreateKnots` algorithm.

Output: $arc_k, k \in \mathbb{N}$ lists of ordered edges from $K_j, j \in \mathbb{N}$,
where the lists of edges $arc_k, k \in \mathbb{N}$ represent the arcs in all the knot components $K_j, j \in \mathbb{N}$, and $m + 1$, with $m \in \mathbb{N}$ representing the total number of arcs in the approximate link.

1: **for all** $i \leftarrow 0$ to $length(EI)$ **do**
2: consider (e_i, f_i) the i-th ordered pair of intersection edges with e_i under f_i in $\mathbb{R}^3$ as computed with `ArrangeEdgesIntersect`(EI)
3: split the undergoing edge e_i into the following pair of edges $e_i \rightarrow (e_i^d, e_i^u)$
4: search for e_i in all of the $K_j, j \in \mathbb{N}$ knot components and when e_i is found in the corresponding knot component K_j make the following substitution $e_i \rightarrow (e_i^d, e_i^u)$
5: **end for**
6: $m \leftarrow -1$
7: **for all** $l \leftarrow 0$ to *count* **do**
8: consider K_l the l-th knot component
9: **while** $IsNotEmpty(K_l)$ **do**
10: $m++$
11: consider $arc_m \leftarrow \emptyset$ an arc of the knot component K_l
12: insert into arc_m all the edges from K_l between the first edge of type e^u and the first consecutive edge of type e^d
13: delete all the edges inserted in arc_m from K_l
14: **end while**
15: **end for**
16: **return** $\{\langle arc_k, k \in \mathbb{N}\rangle, m\}$

For deciding the type of crossings in the diagram of an ϵ-link L_ϵ, we observe that in each knot component $K_j, j \in \mathbb{N}$ for a positive edge $e_i(s_i, d_i)$ the property $xcoord(s_i) < xcoord(d_i)$ is true, whereas for a negative edge $-e_j(s_j, d_j)$ the property $xcoord(s_j) > xcoord(d_j)$ is true. Each type of crossing (i.e. lefthanded crossing or righthanded crossing) depends on the ordered pair of intersection edges (e_{under}, e_{over}) that contains the corresponding intersection point, and that is:

(1) the type of crossings depends on the orientation of the edge e_{under}, i.e. the type of crossing depends on whether the edge e_{under} is oriented from left to right (i.e. it is a positive edge) or whether the edge e_{under} is oriented from right to left (i.e. it is a negative edge), see Figure 3.17;

(2) the type of crossings depends on the orientation of the edge e_{over}, i.e. the type of crossing depends on whether the edge e_{over} is oriented from left to right (i.e. it is a positive edge) or whether the edge e_{over} is oriented from right to left (i.e. it is a negative edge), see Figure 3.17;

(3) the type of crossings depends on the relation between the slope of e_{under} and the slope of e_{over}.

Depending on these three properties, we have $2^3 = 8$ 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 assume that $xcoord(s_l) > xcoord(d_l)$ for the negative undergoing edge e_l, and we assume that $xcoord(s_k) < xcoord(d_k)$ for the positive overgoing edge e_k. If additionally we suppose that $slope(e_l) < slope(e_k)$, then we notice that c is a lefthanded crossing. For instance, in Figure 3.21 we can decide each type of crossings as follows:

- $c_1 = (-e_n, e_m)$ is a lefthanded crossing, since $xcoord(s_n) > xcoord(d_n)$, $xcoord(s_m) < xcoord(d_m)$ and $slope(e_m) < slope(-e_n)$;

- using the same reasoning as for the crossing c_1, we notice that $c_2 = (e_l, e_k), c_3 = (e_s, -e_t)$ are both lefthanded crossings. For $c_2 = (e_l, e_k)$ we notice that $xcoord(s_l) > xcoord(d_l)$, $xcoord(s_k) < xcoord(d_k)$ and $slope(e_l) < slope(-e_k)$. In the same way for $c_3 = (e_s, -e_t)$ we observe that $xcoord(s_s) > xcoord(d_s)$, $xcoord(s_t) < xcoord(d_t)$ and $slope(e_s) < slope(-e_t)$.

We now describe the `DecideTypeCrossings` algorithm. This algorithm uses the `GetSlope` algorithm from Subsection 3.3.1, which computes the slope of an edge $e(s, d)$. The algorithm for deciding the type of crossings denoted with `DecideTypeCrossings` first extracts the x and y coordinates of the source point and of the destination point of the edge e obtaining two points in $\mathbb{R}^2$ denoted with $A\big(xcoord(e.s), ycoord(e.s)\big), B\big(xcoord(e.d), ycoord(e.d)\big)$. Then the algorithm computes the slope of e in $\mathbb{R}^2$ with the `GetSlope`(A, B) algorithm. For simplicity reasons, we assume that we compute the slope of the edge e as explained before using the procedure called $slope(e)$. We use the procedure $slope(e)$ in the `DecideTypeCrossings` algorithm whenever we want to compute the slope of an edge $e(s, d)$ from a 3-dimensional graph. We mention that the procedure $slope(e)$ basically computes the slope of the projection of the edge $e(s, d)$ from a 3-dimensional graph $Graph(L_\epsilon)$, which approximates the ϵ-link L_ϵ of a plane curve singularity. The output of the `DecideTypeCrossings` algorithm is a list T of $\{-1, +1\}$ elements. The length of the list T is the same as the length of the set I of intersections points among all the edges E from the 3-dimensional graph $Graph(L_\epsilon)$. The ith element of the list T represents the type of the crossing corresponding to the ith intersection point from the list I.

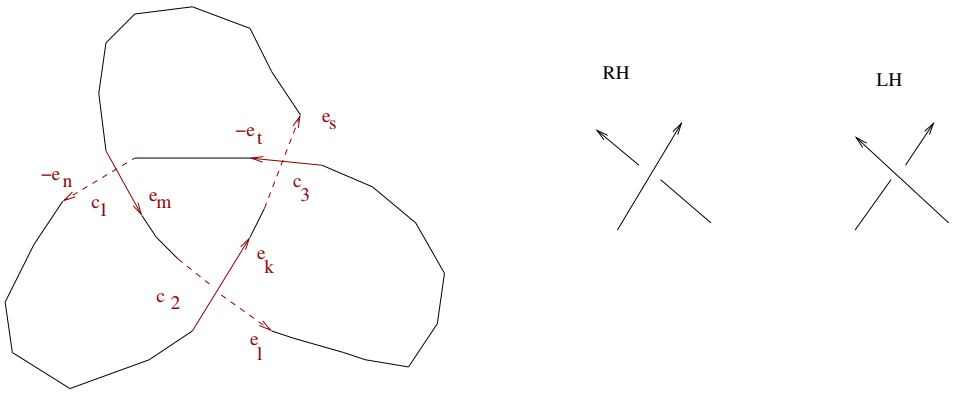


Figure 3.21: Deciding the type of crossings for the diagram of an ϵ-link represented as a 3-dimensional graph data structure. A crossing is lefthanded (denoted with -1 or with LH) if the underpass traffic goes from left to right, whereas a crossing is righthanded (denoted with $+1$ or RH) if the underpass traffic goes from right to left.

Algorithm 20 Decide the type of crossings for the diagram of an ϵ-link L_ϵ represented as a 3-dimensional graph data structure $Graph(L_\epsilon) = \langle V, E\rangle$, where EI represents the ordered pairs of intersections points for all the intersection points I reported among all the edges from E. The algorithm is denoted with: `DecideTypeCrossings`(EI)

Input: EI as computed with the `SweepPlane` algorithm.
Output: T a list of $\{-1, 1\}$ elements with the same `length` as EI,
where the i-th element from T represents the type of the crossing corresponding to the i-th ordered pair of intersection edges (e_i, f_i) from EI.

1: **for all** $i \leftarrow 0$ to $length(EI)$ **do**
2: consider (e_i, f_i) the i-th ordered pair of intersection edges with e_i under f_i in $\mathbb{R}^3$ as computed with `ArrangeEdgesIntersect`(EI)
3: **if** $(xcoord(e_i.s) < xcoord(e_i.d)$ and $xcoord(f_i.s) < xcoord(f_i.d)$ and $slope(e_i) > slope(f_i))$ or
$(\ xcoord(e_i.s) < xcoord(e_i.d)$ and $xcoord(f_i.s) > xcoord(f_i.d)$ and $slope(e_i) < slope(f_i)\ )$ or
$(\ xcoord(e_i.s) > xcoord(e_i.d)$ and $xcoord(f_i.s) < xcoord(f_i.d)$ and $slope(e_i) < slope(f_i)\ )$ or
$(\ xcoord(e_i.s) < xcoord(e_i.d)$ and $xcoord(f_i.s) < xcoord(f_i.d)$ and $slope(e_i) > slope(f_i)\ )$ **then**
4: $t_i \leftarrow 1$
5: **end if**
6: **if** $(xcoord(e_i.s) < xcoord(e_i.d)$ and $xcoord(f_i.s) < xcoord(f_i.d)$ and $slope(e_i) < slope(f_i)\ )$ or
$(\ xcoord(e_i.s) < xcoord(e_i.d)$ and $xcoord(f_i.s) > xcoord(f_i.d)$ and $slope(e_i) > slope(f_i)\ )$ or
$(\ xcoord(e_i.s) > xcoord(e_i.d)$ and $xcoord(f_i.s) < xcoord(f_i.d)$ and $slope(e_i) > slope(f_i)\ )$ or
$(\ xcoord(e_i.s) < xcoord(e_i.d)$ and $xcoord(f_i.s) < xcoord(f_i.d)$ and $slope(e_i) < slope(f_i)\ )$
then
7: $t_i \leftarrow -1$
8: **end if**
9: **end for**
10: **return** $T = \{t_i, i \in \{0, ..., length(EI)\}\}$

3.3.3 Description of the Main Algorithm

In this subsection we include the main algorithm for computing the ϵ-Alexander polynomial attached to the ϵ-link L_ϵ of a plane curve singularity, ϵ-link that is approximated as a 3-dimensional graph data structure $Graph(L_\epsilon) = \langle V, E\rangle$, where V represents the set of points together with their Euclidean coordinates in $Graph(L_\epsilon)$ and E represents the set of edges in $Graph(L_\epsilon)$. For designing the algorithm for computing the approximate Alexander polynomial we use the notions defined in Chapter 2, Subsection 2.4.3 and in Section 2.5. In addition, we employ the algorithms discussed in the previous subsections. We notice that basically the computation of the ϵ-Alexander polynomial Δ_ϵ attached to the ϵ-link of a plane curve singularity consists of the following two steps:

1. **Step 1:** from the 3-dimensional graph data structure $Graph(L_\epsilon) = \langle V, E\rangle$, which approximates the ϵ-link L_ϵ, we compute the diagram of L_ϵ denoted with $D\big(Graph(L_\epsilon)\big)$. We assume that V represents the list of vertices in $Graph(L_\epsilon)$ and E represents the

list of edges in $Graph(L_\epsilon)$. We call this algorithm `ApproxDiagram`. The algorithm basically computes the crossings of the diagram and the type of crossings in the diagram. In addition, the algorithm computes the lists of edges from E that represents all the knots components $K_j, j \in \mathbb{N}$ in the diagram and the total number of knot components in the diagram. Moreover the algorithm computes the list of edges from $K_j, j \in \mathbb{N}$ that represents all the arcs of the diagram. For a graphical description of the input-output specification of the `ApproxDiagram` see Figure 3.22.

2. **Step 2:** from the diagram $D\big(Graph(L_\epsilon)\big)$ computed previously in **Step 1** with the `ApproxDiagram` algorithm, we compute the ϵ-Alexander polynomial of the ϵ-link L_ϵ approximated as a graph data structure $Graph(L_\epsilon)$. We call this algorithm `ApproxAlexPoly`. We recall that the Alexander polynomial attached to the link of a singularity is an invariant that uniquely identifies the links of singularities called also algebraic links. It follows that the Alexander polynomial is a complete invariant for algebraic links. We recall how the Alexander polynomial attached to a plane curve singularity can be used to analyse the topology of the plane curve singularity. We assume that Q_1 and Q_2 are the singularities of a plane complex algebraic curve. In addition, we suppose that we computed the links of the singularity Q_1 and Q_2 denoted with L_1 and respectively with L_2. Moreover we consider that we computed the Alexander polynomials of the links L_1 and L_2 denoted with Δ_1 and respectively Δ_2. If the computed Alexander polynomials Δ_1 and Δ_2 of the links L_1 and L_2 are the same, then the topology of the singularity Q_1 and Q_2 is the same. If the computed Alexander polynomials Δ_1 and Δ_2 of the links L_1 and L_2 are different, then the topology of the singularity Q_1 and Q_2 is different. The `ApproxAlexPoly` algorithm basically computes the ϵ-Alexander polynomial Δ_ϵ of the ϵ-link L_ϵ. We recall that Δ_ϵ is a Laurent polynomial. For a graphical visualization of the input-output specification of the `ApproxAlexPoly` see Figure 3.23.

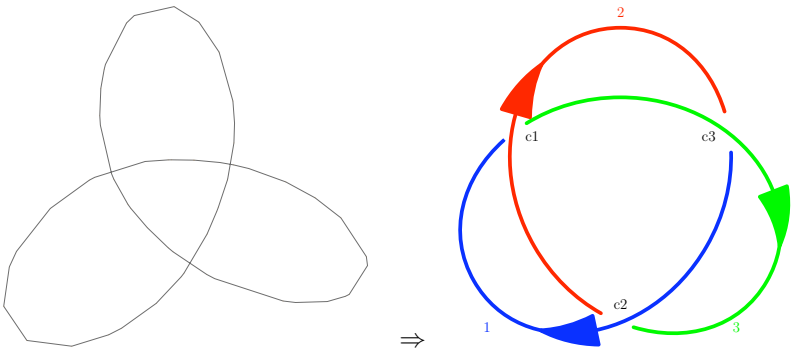
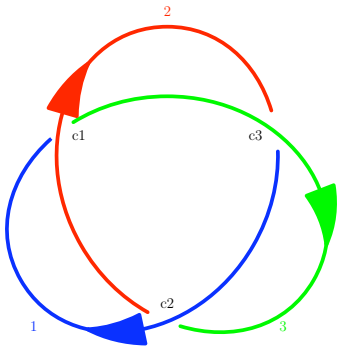


Figure 3.22: Input-output specification of the `ApproxDiagram` algorithm, which from the 3-dimensional graph data structure $Graph(L_\epsilon)$ that approximates the ϵ-link L_ϵ, it computes the diagram of the ϵ-link denoted with $D\big(Graph(L_\epsilon)\big)$.

We give the schematic algorithm `ApproxDiagram` for the computation of the diagram $D\big(Graph(L_\epsilon)\big)$ of an ϵ-link L_ϵ computed as in Section 3.2 and approximated by a 3-dimensional graph data structure $Graph(L_\epsilon)$. This algorithm operates on the $Graph(L_\epsilon)$ data structure. In addition the algorithm `ApproxDiagram` uses the sweep line algorithm `SweepPlane`, the `ArrangeEdgesIntersect` and the combinatorial algorithms `CreateKnots`, `CreateArcs` and `DecideTypeCrossings`.



$\Rightarrow$ Δ_ϵ, the ϵ-Alexander polynomial

Figure 3.23: Input-output specification of the `ApproxAlexPoly` algorithm, which from the diagram of the ϵ-link L_ϵ denoted with $Graph(L_\epsilon)$, it computes the ϵ-Alexander polynomial Δ_ϵ of the ϵ-link L_ϵ approximated as a graph data structure $Graph(L_\epsilon)$.

Algorithm 21 Approximate diagram of the piecewise linear approximate link $Graph(L_\epsilon)$: `ApproxDiagram`$\big(Graph(L_\epsilon)\big)$

Input: $Graph(L_\epsilon) = \langle V, E\rangle$ piecewise linear algebraic link, which approximates
L_ϵ an approximate differentiable algebraic link as computed in Section 3.2,
$Graph(L_\epsilon)$ is a 3-dimensional graph,
V the set of vertices in $Graph(L_\epsilon)$ with their Euclidean coordinates,
E the set of edges in $Graph(L_\epsilon)$.
Output: $D\big(Graph(L_\epsilon)\big)$,
where $D\big((Graph(L_\epsilon)\big)$ is the counterclockwise oriented ϵ-diagram of $Graph(L_\epsilon)$ with the property that $Graph(L_\epsilon)$ can be continuously deformed into L_ϵ, i.e. $Graph(L_\epsilon)$ and L_ϵ are isotopic.

1: Compute the crossings of $D(Graph(L_\epsilon))$ using the sweep-line algorithms described in Subsection 3.3.1.

   (a) Compute I the set of intersections among all the edges of E using the sweep line algorithm `SweepPlane`$(Graph(L_\epsilon), V, E)$. In addition, the same algorithm reports EI the set of pairs of edges containing all the computed intersections points.

   (b) Order the pairs of edges of intersection points, pairs of edges from the set EI using the algorithm `ArrangeEdgesIntersect`(EI).

2: Compute $K_j, j \in \mathbb{N}$ the lists of edges of intersection from E representing all the knot components of $D\big(Graph(L_\epsilon)\big)$ using the combinatorial algorithm `CreateKnots`$\big(Graph(L_\epsilon), V, E\big)$ described in Subsection 3.3.2.

3: Compute the arcs of the diagram $D\big(Graph(L_\epsilon)\big)$ using the combinatorial algorithm `CreateArcs`$\big(\langle I, EI\rangle, \langle\{K_j, j \in \mathbb{N}\}, count\rangle\big)$ described in Subsection 3.3.2.

4: Compute the type of crossings in $D\big(Graph(L_\epsilon)\big)$ using the combinatorial algorithm `DecideTypeCrossings`(EI) described in Subsection 3.3.2.

5: Return $D\big(Graph(L_\epsilon)\big)$.

We now describe the algorithm `ApproxAlexPoly`, which computes the ϵ-Alexander polynomial of an ϵ-link L_ϵ approximated by a 3-dimensional graph data structure $Graph(L_\epsilon)$ by using Definition 50, Definition 51 and Definition 52 from Subsection 2.4.3, Chapter 2. This algorithm takes as input the diagram $D\big(Graph(L_\epsilon)\big)$ of the ϵ-link L_ϵ. The number of knot components in the diagram $D\big(Graph(L_\epsilon)\big)$ is denoted with r, while the number of crossings in the same diagram $D\big(Graph(L_\epsilon)\big)$ is denoted with n. The algorithm returns as output the ϵ-Alexander polynomial of the ϵ-link L_ϵ represented as a 3-dimensional graph $Graph(L_\epsilon)$. We recall that the Alexander polynomial of a plane curve singularity is a Laurent polynomial. We mention that once we compute the ϵ-Alexander polynomial Δ_ϵ we detect the number of variables in Δ_ϵ and the degree of Δ_ϵ. These parameters are required for computing other approximate invariants of a plane complex algebraic curve.

Algorithm 22 Approximate Alexander polynomial of the approximate diagram $D(Graph(L_\epsilon))$: `ApproxAlexPoly`$(D(L_\epsilon), r, n)$

Input: $D\big(Graph(L_\epsilon)\big)$ counterclockwise oriented ϵ-diagram of $Graph(L_\epsilon)$ with r components and n crossings as computed with the **Algorithm** `ApproxDiagram`.
Output: $\Delta_\epsilon(t_1, ..., t_r) \in \mathbb{Z}[t_1^{\pm 1}, ..., t_r^{\pm 1}]$,
where $\Delta_\epsilon(t_1, ..., t_r)$ is the ϵ-Alexander polynomial of L_ϵ with diagram $D\big(Graph(L_\epsilon)\big)$.

1: Denote the arcs and separately the crossings of $D\big(Graph(L_\epsilon)\big)$ with $\{1, ..., n\}$.

2: Compute $LM(L_\epsilon)$ the labelling matrix of $D\big(Graph(L_\epsilon)\big)$ with Definition 50, Subsection 2.4.3, Chapter 2.

3: Compute $PM(L_\epsilon)$ the prealexander matrix of $D\big(Graph(L_\epsilon)\big)$ with Definition 51, Subsection 2.4.3, Chapter 2.

4: If $r = 1$ then:

   (a) Compute M any $(n-1) \times (n-1)$ minor of $PM(L_\epsilon)$.

   (b) Compute D the determinant of the minor M.

   (c) Return $\Delta_\epsilon(t_1) = Normalize(D_\epsilon)$.

5: If $r \geq 2$ then:

   (a) Compute all the $(n-1) \times (n-1)$ minors of $PM(L_\epsilon)$.

   (b) Compute G the greatest common divisor of all the computed minors in 5.(a).

   (c) Return $\Delta_\epsilon(t_1, ...t_r) = Normalize(G)$.

3.3.4 Applications of the Main Algorithm

In this subsection we assume that we computed the ϵ-Alexander polynomial Δ_ϵ of the ϵ-link L_ϵ represented as a 3-dimensional graph $Graph(L_\epsilon)$. We recall that L_ϵ represents the ϵ-link of the singularity Q of a plane complex algebraic curve denoted with $\mathcal{C}$. In addition, we remember that the ϵ-link arises from the intersection of the curve $\mathcal{C}$ with a small sphere $S_\epsilon(Q)$ of radius ϵ centered in the singularity Q. We assume that n denotes the degree of Δ_ϵ and that r represents the number of variables in Δ_ϵ. From Subsection 2.4.4 and from Section 2.5 we notice that we can employ the ϵ-Alexander polynomial to derive a formula for the ϵ-Milnor number μ_ϵ of the singularity Q. From the ϵ-Alexander polynomial attached to the singularity Q of a plane complex algebraic curve $\mathcal{C}$, we also derive a formula for the

ϵ-delta-invariant of the singularity Q of the plane complex algebraic curve $\mathcal{C}$. We describe this algorithm in Subsection 3.4.1. We now describe the algorithm `ApproxMilnorNumber` for computing the ϵ-Milnor number of the singularity Q of a plane complex algebraic curve given the ϵ-Alexander polynomial Δ_ϵ attached to the ϵ-link L_ϵ of the singularity Q.

Algorithm 23 Approximate Milnor number of the singularity Q having approximate Alexander polynomial Δ_ϵ with n being its degree: `ApproxMilnorNumber`(Δ_ϵ, n)

Input: $\Delta_\epsilon(t_1, ..., t_r)$ the ϵ-Alexander polynomial of L_ϵ computed with the algorithm `ApproxAlex`,
L_ϵ the ϵ-link of the singularity $Q(z_0, w_0)$ of the plane complex algebraic curve $\mathcal{C}$,
n the degree of Δ_ϵ.
Output: $\mu_\epsilon \in \mathbb{Z}_{>0}$,
where μ_ϵ is the ϵ-Milnor number of $Q(z_0, w_0)$.

1: If $r = 1$, then return $\mu_\epsilon = n$.

2: If $r \geq 2$, then return $\mu_\epsilon = n + 1$.

3.4 Algorithm for Computing the Approximate Delta-Invariant

3.4.1 Description of the Algorithm

We now present the algorithm `ApproxDelta`(Δ_ϵ, n, r) for computing the ϵ-delta-invariant of the singularity Q of the plane complex algebraic curve $\mathcal{C}$ from the ϵ-Alexander polynomial Δ_ϵ of degree n and with r variables. We mention that the ϵ-Alexander polynomial is computed from the 3-dimensional graph data structure $Graph(L_\epsilon)$, graph that approximates the ϵ-link of the singularity Q of the curve $\mathcal{C}$. Moreover the ϵ-link L_ϵ is computed as the stereographic projection of the intersection of the curve $\mathcal{C}$ with a small sphere $S_\epsilon(Q)$ of radius ϵ and centered in the singularity Q.

Algorithm 24 Approximate delta-invariant of the singularity Q having approximate Alexander polynomial Δ_ϵ with n being its degree and r being its number of variables: `ApproxDelta`(Δ_ϵ, n, r)

Input: $\Delta_\epsilon(t_1, ..., t_r)$ the ϵ-Alexander polynomial of L_ϵ computed with the algorithm `ApproxAlexPoly`,
L_ϵ the ϵ-link of the singularity $Q(z_0, w_0)$ of the plane complex algebraic curve $\mathcal{C}$,
n the degree of Δ_ϵ, r the number of variables in Δ_{L_ϵ}.
Output: $\delta_\epsilon \in \mathbb{Z}_{>0}$,
where δ_ϵ is the ϵ-delta-invariant of $Q(z_0, w_0)$.

1: If $r = 1$, then return $\delta_\epsilon = n/2$.

2: If $r \geq 2$, then return $\delta_\epsilon = (n + r)/2$.

3.4.2 Applications of the Algorithm

Based on Subsection 2.4.5 and on Section 2.5, we observe that from the ϵ-delta-invariants of all the singularities of a plane complex algebraic curve $\mathcal{C}$ of degree m we can derive a formula for the ϵ-genus of the plane complex algebraic curve $\mathcal{C}$, which is discussed in Subsection 3.6.1. We mention that the set of singularities of $\mathcal{C}$ have to be computed in the projective complex plane.

3.5 Algorithm for Computing the Approximate Local Topological Type

From Chapter 2, we notice that in order to understand the local topology of a plane complex algebraic curve around its singular point Q it suffices to understand the link of the singularity Q. Hence, we are interested in studying topological invariants such as the Alexander polynomial of the link of the singularity, which determines completely the local topology of the plane complex algebraic curve around its singular point.

We now describe the algorithm $\texttt{ApproxType}(p, \mathcal{C}, Q, \epsilon)$ for computing the ϵ-local topological type of the singularity $Q(z_0, w_0)$ of the plane complex algebraic curve $\mathcal{C}$ defined by the squarefree polynomial $p(z, w,)$ with exact and with inexact coefficients. The parameter $\epsilon \in \mathbb{R}_{>0}$ represents the radius of the sphere $S_\epsilon(Q)$ centered in the origin Q that we intersect with the input curve $\mathcal{C}$ in order to compute the ϵ-link of the singularity Q.

Algorithm 25 ϵ-local topological type of the singularity O of the plane curve $\mathcal{C}$ defined by $p(z, w)$: $\texttt{ApproxType}(p, \mathcal{C}, Q, \epsilon)$

Input: $p(z, w) \in \mathbb{C}[z, w]$ a squarefree complex polynomial,
$\mathcal{C} = \{(z, w) \in \mathbb{C}^2 \mid p(z, w) = 0\}$ a plane algebraic curve,
$Q(z_0, w_0)$ a numerical singularity of $\mathcal{C}$,
$\epsilon \in \mathbb{R}_{>0}$ a positive real number.

Output: the pair $\big(L_\epsilon(Q), \Delta_\epsilon(Q), \delta_\epsilon(Q)\big)$ representing the ϵ-local topological type of Q, where $L_\epsilon(Q)$ is the ϵ-link of Q,
$\Delta_\epsilon(Q)$ is the ϵ-Alexander polynomial of Q and
$\delta_\epsilon(Q)$ is the ϵ-delta-invariant of Q.

1: Compute the ϵ-link $L_\epsilon(Q)$ of the singularity Q by using the algorithm $\texttt{ApproxLink}(p, \mathcal{C}, Q, \epsilon)$.

2: Compute the graph $Graph(L_\epsilon)$ of the ϵ-link L_ϵ by using subdivision methods from [Liang et al., 2008] implemented in the Axel system.

3: Compute the diagram $D(L_\epsilon)$ with m components and n crossings of $Graph(L_\epsilon)$ by using the algorithm $\texttt{ApproxDiagram}\big(Graph(L_\epsilon)\big)$.

4: Compute the ϵ-Alexander polynomial $\Delta_\epsilon(Q)$ of degree n and with r variables of the singularity Q by using the algorithm $\texttt{ApproxAlexPoly}(D(L_\epsilon), m, n)$.

5: Compute the ϵ-delta-invariant $\delta_\epsilon(Q)$ of the singularity Q by using the algorithm $\texttt{ApproxDelta}(\Delta_\epsilon, \mu, r)$.

6: Return the pair $\big(L_\epsilon(Q), \Delta_\epsilon(Q), \delta_\epsilon(Q)\big)$.

3.6 Algorithm for Computing the Approximate Genus

3.6.1 Description of the Algorithm

We now give the schematic algorithm for computing the ϵ-genus of a plane complex algebraic curve $\mathcal{C}$ defined by a squarefree complex bivariate polynomial $p(z,w) \in \mathbb{C}[z,w]$ with exact and inexact coefficients. We assume that the degree of the curve $\mathcal{C}$ $\big($and thus of the polynomial $p(z,w)\big)$ is m. Moreover for the inexact data in the polynomial $p(z,w)$ we are given a positive real number $\delta \in \mathbb{R}_{>0}$, which measures the error (or noise or tolerance) in the coefficients of $p(z,w)$. Furthermore we assume that we know the input parameter $\epsilon \in \mathbb{R}_{>0}$, which represents the radius of the sphere centered in the singularity Q of $\mathcal{C}$, sphere that we intersect with the curve $\mathcal{C}$ to compute the ϵ-link L_ϵ of Q. In addition, we assume that we are give a subset $B = [-a,a] \times [-b,b]$ of the 2-dimensional plane $\mathbb{R}^2$, subset that we call box and that is needed for computing the numerical singularities of the plane complex algebraic curve.

Algorithm 26 Approximate genus of a plane complex algebraic curve $\mathcal{C}$ defined by a squarefree complex bivariate polynomial $p(z,w)$ of degree m, where $\epsilon \in \mathbb{R}_{>0}$ is an arbitrary input parameter, $\delta \in \mathbb{R}_{>0}$ is the noise in the coefficients of $p(z,w)$ and $B \subset \mathbb{R}^2$: $\texttt{ApproxGenus}(\mathcal{C}, p, m, \epsilon, \delta, B)$

Input: $\mathcal{C} = \{(z,w) \in \mathbb{C}^2 \mid p(z,w) = 0\}$ a plane complex algebraic curve,
$p(z,w) \in \mathbb{C}[z,w]$ a squarefree polynomial with exact and inexact coefficients,
m the degree of the plane complex algebraic curve $\mathcal{C}$,
$\epsilon \in \mathbb{R}_{>0}$ the input parameter,
$\delta \in \mathbb{R}_{>0}$ the noise in the coefficients of the polynomial $p(z,w)$,
$B = [-a,a] \times [-b,b] \subset \mathbb{R}^2$ a subset of $\mathbb{R}^2$.
Output: $genus_\epsilon(\mathcal{C}) \in \mathbb{Z}$,
where $genus_\epsilon(\mathcal{C})$ is the ϵ-genus of $\mathcal{C}$.

1: $sumDeltaInv = 0$.

2: Compute $NumSing(\mathcal{C}) = \texttt{ApproxRealSing}(\mathcal{C}, p, \delta, B)$.

3: For each $Q_i(z_i, w_i) \in NumSing(\mathcal{C})$ do:

(a) Compute $L_\epsilon = \texttt{ApproxLink}(\epsilon, Q_i, \mathcal{C}, p)$ (L_ϵ is approximated by $Graph(L_\epsilon)$).

(b) Compute $D(L_\epsilon) = \texttt{ApproxDiagram}\big(Graph(L_\epsilon)\big)$.

(c) Compute $\Delta_\epsilon(t_1, ..., t_r) = \texttt{ApproxAlexPoly}\big(D(L_\epsilon), r, n\big)$.

(d) Compute $\delta_\epsilon(Q_i) = \texttt{ApproxDelta}(\Delta_\epsilon, n, r)$.

(e) $sumDeltaInv = sumDeltaInv + \delta_\epsilon(Q_i)$.

4: Return $genus_\epsilon(\mathcal{C}) = \dfrac{(m-1)(m-2)}{2} - sumDeltaInv$.

3.6.2 Applications of the Algorithm

In this subsection we assume that we compute the ϵ-genus of a plane complex algebraic curve $\mathcal{C}$ defined by a squarefree bivariate complex polynomial $p(z,w) \in \mathbb{C}[z,w]$ with exact and inexact coefficients. Based on Subsection 2.4.6 and on Section 2.5, we derive a formula for

the ϵ-Euler characteristic of the Riemann surface attached to the resolution of singularities of $\mathcal{C}$. We now describe the algorithm for computing the ϵ-Euler characteristic.

Algorithm 27 Approximate Euler characteristic of the Riemann surface attached to the resolution of singularities of the plane complex algebraic curve $\mathcal{C}$ defined by the squarefree polynomial $p(z,w) \in \mathbb{C}[z,w]$ and having the ϵ-genus $genus_\epsilon(\mathcal{C})$: $\texttt{ApproxEulerChar}\big(\mathcal{C}, p(z,w), \epsilon, genus_\epsilon(\mathcal{C})\big)$

Input: $\mathcal{C} = \{(z,w) \in \mathbb{C}^2 \mid p(z,w) = 0\}$ a plane complex algebraic curve,
$p(z,w) \in \mathbb{C}[z,w]$ a squarefree polynomial with exact and inexact coefficients,
$genus_\epsilon(\mathcal{C})$ the ϵ-genus of $\mathcal{C}$ computed with the algorithm `ApproxGenus`.
Output: $\chi_\epsilon \in \mathbb{Z}$,
where χ_ϵ is the ϵ-Euler characteristic of the Riemann surface attached to the resolution of singularities of $\mathcal{C}$.

1: Return $\chi_\epsilon = 2 - 2genus_\epsilon(\mathcal{C})$.

3.7 Algorithms for Computing Knot Theory Properties

In this section, we assume that we computed the ϵ-link L_ϵ of the singularity Q of the plane complex algebraic curve $\mathcal{C}$. In this case, L_ϵ is called an algebraic link. We assume that we computed the ϵ-Alexander polynomial of L_ϵ of Q denoted with Δ_ϵ, where n denotes the degree and r denotes the number of variables in the polynomial. In addition, we assume that we computed the ϵ-Milnor number μ_ϵ of Q and the ϵ-delta-invariant δ_ϵ of Q, as described in the previous sections. Based on these invariants, we will now compute several properties from knot property for the algebraic link L_ϵ.

The genus of an algebraic knot (i.e. algebraic link with one component). By definition the genus of a knot (and implicitly the genus of an algebraic knot) is the minimum genus of an orientable surface spanning the knot (and implicitly the algebraic knot) in the 3-dimensional Euclidean space $\mathbb{R}^3$. This surface is called Seifert surface. We denote the genus of an algebraic knot K with $g(K)$. We denote the degree of the Alexander polynomial of the algebraic knot K with n, i.e. n is the Milnor number μ of the singularity corresponding to K. From Milnor's fibration theorem we know that the complement $\mathbb{R}^3 \setminus K$ admits a fibration by Seifert surfaces of minimal genus, i.e. K is a fibered knot. From [Seifert, 1934], we obtain that the degree of the Alexander polynomial of a knot never exceeds twice the genus of the knot. Moreover, from [Murasugi, 1958] and [Crowell, 1959] we obtain that the equality holds for any alternating knot, whereas from [Neuwirth, 1963] we get that the equality holds for any fibered knot. Furthermore, for fibered knots, the Alexander polynomial of K evaluated in 0 is ± 1, i.e. $\Delta(0) = \pm 1$, and this is also sufficient for alternating knots to be fibered knots. Based on Neuwirth-Stallings theorem [Neuwirth, 1963] and on Milnor's research [Milnor, 1968], we deduce that the genus of an algebraic knot K is computed via the formula $g(K) = \frac{\mu}{2}$, where μ is the degree of the Alexander polynomial of K, i.e. μ is the Milnor number of the corresponding singularity of K.

The genus of an algebraic link (i.e. algebraic link with more than one component). By definition the genus of a link (and implicitly the genus of an algebraic link) is the minimum genus of an orientable surface spanning the link (and implicitly the algebraic

link) in the 3-dimensional Euclidean space $\mathbb{R}^3$. This surface is called Seifert surface. We denote the genus of an algebraic link L with $g(L)$. In addition, Milnor's fibration theorem [Milnor, 1968] states that the complement $\mathbb{R}^3 \setminus L$ admits a fibration by Seifert surfaces of minimal genus, i.e. L is a fibered link. For the genus of an algebraic link we can only give a lower bound depending on the Milnor number of the corresponding singularity of the algebraic link. This result is based on the Seifert generalization for links made by [Crowell, 1959]. We noted in the above paragraph that Seifert has shown how to construct, for any knot K prescribed by a regular projection, a tame embedding of a connected orientable surface with boundary K called the Seifert surface. R.H. Crowell showed that the same procedure is applicable to links. If we denote with μ the Milnor number of the algebraic link, i.e. μ is the degree of the Alexander polynomial $+1$, then we have the following relation for computing the genus of the link: $g(L) \geq \frac{\mu}{2} = \frac{n+1}{2}$. The genus of knots and links (and implicitly of algebraic knots and algebraic links) it is an invariant for links and algebraic links, since it is an invariant under the Reidemeister moves, see Chapter 2, Subsection 2.3.3.

The unknotting number of an algebraic link. The unknotting number of a link L with r components, $r \geq 1$ (and implicitly of an algebraic link L with r components, $r \geq 1$) is defined as the smallest number of times that must be allowed for L to cross itself during a smooth deformation so as to transform itself into a collection of r unlinked and unknotted circles. We denote the unknotting number of the algebraic link L with $u(L)$. From [Kronheimer and Mrowka, 1993] we know that the delta-invariant of a plane complex algebraic curve singularity Q equals the unknotting number of the algebraic link L of the singularity Q. Thus if $\delta(Q)$ denotes the delta invariant of the singularity Q, if n denotes the degree and if r denotes the number of variables in the Alexander polynomial of the algebraic link L of Q, then the unknotting number of Q is defined as:

- If $r = 1$, then $u(L) = \delta(Q) = \frac{n}{2}$.
- If $r \geq 2$, then $u(L) = \delta(Q) = \frac{n+r}{2}$.

The unknotting number is an invariant for links and algebraic links since it is invariant under the Reidemeister moves, see Chapter 2, Subsection 2.3.3 for more information.

The number of knot components of an algebraic link. The number of knot components of an algebraic link is also called the multiplicity of the algebraic link. This number remains unchanged under the Reidemeister moves, and therefore it is an invariant for links (and for algebraic links), see Chapter 2 for further details. To determine the number of (knot) components of an algebraic link we proceed in the following way: we choose a point on an arc of the diagram of the algebraic link, we then walk along the diagram and we observe that each (knot) component is a completed cycle.

The linking number of an algebraic link. The linking number is defined for links of at least 2 components (and implicitly for algebraic links with at least 2 components). Given an oriented link of 2 components, and implicitly an oriented algebraic link L of 2 (knot) components J and K, it is possible to define the linking number of the two components J and K denoted with $lk(J, K)$ in the following way:

- Each crossing point in the diagram of the link is assigned a sign as follows: if the crossing point is righthanded, then we assign a $+1$ sign to it. If the crossing point is

lefthanded, then we assign a -1 sign to it. For more information on righthanded and lefthanded crossings see Chapter 2, Subsection 2.3.3.

- The linking number $lk(J,K)$ is defined as the sum of the signs of the crossing points where J and K meet, divided by 2. If for each crossing c where J and K meet we denote with $\epsilon(c)$ the sign of the crossing, then $lk(J,K) = \frac{1}{2}\sum_{c\in J\cap K}\epsilon(c)$.

We add that the linking number is always an integer. Moreover, it is an invariant for links and algebraic links, since it is an invariant under the Reidemeister moves, see Chapter 2, Subsection 2.3.3 for more information. Furthermore, the linking number is symmetric, i.e. $lk(J,K) = lk(K,J)$. We also mention that the linking number is an invariant only for links or algebraic links with at least 2 knot components. The linking number of a knot for its crossings is no longer an invariant under the Reidemeister moves. Thus, the linking number is meaningful only for links and algebraic links consisting of 2 or more (knot) components. For instance, the linking number of the Hopf link is ± 1, and the linking number of the unlink is 0. By convention, we consider the linking number of the unknot to be 0, and the linking number of any nontrivial knot as undefined. A nontrivial knot is any knot which is different from the unknot. In formulating this definition for the linking number of an algebraic link, we follow the book of [Livingston, 1993].

The determinant of an algebraic link.

- *The determinant of an algebraic knot, i.e. an algebraic link with one (knot) component.* Let $\Delta(t)$ be the Alexander polynomial of the algebraic knot K. Then the determinant of the algebraic knot K denoted with $\det(K)$ is defined as the absolute value of the Alexander polynomial of K evaluated at -1, i.e. $\det(K) = |\Delta(-1)|$, see [Murasugi, 1996]. The determinant of an algebraic knot is always an odd positive integer (this is not true in the case of algebraic links).

- *The determinant of an algebraic link, i.e. an algebraic link with more than one (knot) components.* Let $\Delta(t_1,...,t_r)$ be the Alexander polynomial of the algebraic link L with $r \geq 2$ (knot components). Then the determinant of the algebraic link L denoted with $\det(L)$ is defined as the absolute value of the Alexander polynomial of L evaluated at $(-1,...,-1)$, i.e. $\det(L) = |\Delta(-1,...,-1)|$. The determinant of an algebraic link is a positive integer.

The determinant of an algebraic knot or of an algebraic link is not an invariant of the algebraic knot or of the algebraic link, it only helps in deciding the colorability property of an algebraic knot or of an algebraic link. The colorability property is an invariant for algebraic knots or for algebraic links since it is invariant under the Reidemeister moves, see Chapter 2, Subsection 2.3.3 for details.

The colorability of an algebraic link. We decide the colorability property of an algebraic link L of r components with $r \geq 1$ being a positive integer following the book of [Livingston, 1993]. We recall that a knot or link (and implicitly an algebraic knot or an algebraic link) diagram is p-colorable if each arc can be labelled with an integer from $\{0,...,p-1\}$ such that the following properties are true: (1) at each crossing the relation $2x - y - z = 0 (mod\ p)$ is true, where x is the label of the overcrossing and y and z are the two labels of the undercrossing, and (2) at least two labels are distinct.

From [Livingston, 1993] we know that a knot or a link (and implicitly an algebraic knot or an algebraic link) is p-colorable if and only if p divides the determinant of the knot or of the

link (and implicitly the determinant of the algebraic knot or of the algebraic link). Thus by computing the determinant of the knot or of the link (and implicitly of the algebraic knot or of the algebraic link), we can decide the p-colorability of the knot or of the link (and implicitly of the algebraic knot or of the algebraic link), in the following way: (1) If $\det(L) = 0$, then L is colorable via any p-numbering, with p being a positive prime integer. (2) If $\det(L) = 1$, then L is not colorable via any p-numbering, with p being a positive prime integer. (3) If $\det(L) \neq 1$ and $\det(L) \neq 0$ and $3|\det(L)$, then L is colorable via a 3-numbering, i.e. L is tricolorable. (4) If $\det(L) \neq 1$ and $\det(L) \neq 0$ and $p|\det(L)$, then L is colorable via any p-numbering, with p being a positive prime integer.

We make the observation that the invariants of algebraic links described in this section (i.e. the genus, the unknotting number, the linking number, the colorability) are not complete invariants for the algebraic links, i.e. there exist different algebraic links having the same value for these invariants.

Algorithm 28 Knot theory properties of the ϵ-link $L_\epsilon(Q)$ of the singularity Q, with $\Delta_\epsilon(Q)$ being its ϵ-Alexander polynomial, $\mu_\epsilon(Q)$ being its ϵ-Milnor number and $\delta_\epsilon(Q)$ being its ϵ-delta-invariant: `PropApproxLink`$\big(L_\epsilon, \Delta_\epsilon, n, r, \mu_\epsilon, \delta_\epsilon\big)$

Input: $L_\epsilon(Q)$ the ϵ-link of the singularity Q,
$\Delta_\epsilon(Q)$ the ϵ-Alexander polynomial of L_ϵ of Q,
n the degree of Δ_ϵ, r the number of variables in Δ_ϵ,
$\mu_\epsilon(Q)$ the ϵ-Milnor number of the singularity Q,
$\delta_\epsilon(Q)$ the ϵ-delta-invariant of Q.
Output: $\big(g(L_\epsilon), u(L_\epsilon), m(L_\epsilon), lk(L_\epsilon), \det(L_\epsilon)\big)$,
where $g(L_\epsilon)$ denotes the genus of L_ϵ,
$u(L_\epsilon)$ denotes the unknotting number of L_ϵ,
$m(L_\epsilon)$ denotes the multiplicity of L_ϵ,
$lk(L_\epsilon)$ denotes the linking numbers of the knot components of L_ϵ, and
$\det(L_\epsilon)$ denotes the determinant of L_ϵ.

1: If $r = 1$, then $g(L_\epsilon) = \dfrac{\mu}{2}$, $u(L_\epsilon) = \delta_\epsilon(Q) = \dfrac{n}{2}$, $m(L_\epsilon) = 1$, $\det L_\epsilon = |\Delta_\epsilon(-1)|$.

2: If $r \geq 2$, then $g(L_\epsilon) \geq \dfrac{\mu}{2} = \dfrac{n+1}{2}$, $u(L_\epsilon) = \delta_\epsilon(Q) = \dfrac{n+r}{2}$, $m(L_\epsilon) = r$, $\det L_\epsilon = |\Delta_\epsilon(-1, ..., -1)|$.

3: If L_ϵ is the unknot or the unlink, then $lk(L_\epsilon) = 0$.

4: If L_ϵ is a nontrivial knot, then $lk(L_\epsilon)$ is "undefined"!
else compute a sequence concerning the linking numbers of all the knot components of L_ϵ.

5: If $\det(L_\epsilon) = 0$, then L is colorable via any p-numbering, with p being a positive prime integer.

6: If $\det(L_\epsilon) = 1$, then L is not colorable via any p-numbering, with p being a positive prime integer.

7: If $\det(L_\epsilon) \neq 1$ and $\det(L_\epsilon) \neq 0$ and $3|\det(L_\epsilon)$, then L is colorable via a 3-numbering, i.e. L is tricolorable.

8: If $\det(L_\epsilon) \neq 1$ and $\det(L_\epsilon) \neq 0$ and $p|\det(L_\epsilon)$, then L is colorable via any p-numbering, with p being a positive prime integer.

9: Return $\big(g(L_\epsilon), u(L_\epsilon), m(L_\epsilon), lk(L_\epsilon), \det(L_\epsilon)\big)$.

Remark 14. We make an observation concerning the way in which we can use the invariants introduced in this subsection to extract more information on a plane complex algebraic curve. We recall that in Chapter 2, Subsection 2.4.3, Definition 52 we define the Alexander polynomial of the unknot (or of the trivial knot) as being equal to 1, and the Alexander polynomial of the unlink (or of the trivial link) as being equal to 0. In addition, from Chapter 2, Subsection 2.4.3, Example 19, we know that the Alexander polynomial of the Hopf link is equal to 1. Consequently, we need some more information to distinguish between the unknot and the Hopf link. For this purpose, we basically use the linking number. We notice that the linking number of the unknot is being equal to 0, whereas the linking number of the Hopf link is equal to ± 1. It follows that we can distinguish between the unknot and the Hopf link in the following way:

- If the Alexander polynomial of a knot K is 1 and the linking number of K is 0, then K coincides with the unknot.
- If the Alexander polynomial of a link L is 1 and the linking number of L is ± 1, then L coincides with the Hopf link..

In fact we notice that we can distinguish between the unknot and the Hopf link also based on the Alexander polynomial and on the number of (knot) components as follows:

- If a link L has one (knot) component and the Alexander polynomial of L is 1, then L coincides with the unknot.
- If a link L has at least 2 (knot) components and the Alexander polynomial of L is 1, then L coincides with the Hopf link. In this case, we assume that we know the number of (knot) components in the link L, i.e. the multiplicity of L.

In Section 3.1, we use subdivison methods from [Mourrain and Pavone, 2009] to compute the set of numerical singularities of a plane complex algebraic curve in the projective plane. We recall that for a plane complex algebraic curve $\mathcal{C}$ defined by a squarefree polynomial $p(z, w) \in \mathbb{C}[z, w]$ with exact and inexact coefficients, for a given tolerance $\delta \in \mathbb{R}_{>0}$ in the coefficients, and for a subset $B \subset \mathbb{R}^2$, these subdivision methods return a list M of boxes in B smaller than δ containing all the singularities of $\mathcal{C}$. Still, the existence and the uniqueness of a singularity in each box is not guaranteed. By considering the middle points Q of each box from M as singularities of $\mathcal{C}$, we compute the ϵ-link L_ϵ of each Q with the algorithm `ApproxLink` from Section 3.2. In addition, we compute the ϵ-Alexander polynomial of L_ϵ with the algorithm `ApproxAlexPoly` from Section 3.3, and the linking number of L_ϵ with the algorithm `PropApproxLink` from this subsection. We can decide the existence of a singular point in the list of boxes M as follows:

- If the ϵ-Alexander polynomial of L_ϵ is 1 and the linking number of L_ϵ is 0, then L_ϵ is the unknot. Thus from [Milnor, 1968], it follows that Q is not a singular point of $\mathcal{C}$.
- If the ϵ-Alexander polynomial of L_ϵ is 0, then L_ϵ is the unlink. From [Milnor, 1968], it follows that Q is not a singular point of $\mathcal{C}$.
- If the ϵ-Alexander polynomial of L_ϵ is 1 and the linking number of L_ϵ is 1, then L_ϵ is the Hopf link. Thus from [Milnor, 1968], it follows that Q is a singular point of $\mathcal{C}$.
- If (L_ϵ has at least 2 knot components and the ϵ-Alexander polynomial of L_ϵ is different from 0) or (L_ϵ has 1 knot component and the ϵ-Alexander polynomial of L_ϵ is different from 1), then L_ϵ is a nontrivial link or a nontrivial knot. Thus from [Milnor, 1968], it follows that Q is a singular point of $\mathcal{C}$.

Chapter 4

Convergence Analysis of the Symbolic-Numeric Algorithms

We mention that in this thesis we approach the algebraic problem of computing topological invariants of a plane complex algebraic curve $\mathcal{C}$ defined by a squarefree polynomial $p(z,w) \in \mathbb{C}[z,w]$ with both exact and inexact data. For the inexact data we associate a positive real number called noise, which measures the error level in the coefficients of the defining polynomial of the input curve. We deal with an ill-posed problem, i.e. small changes in the input data produce huge changes in the output solution. Moreover, we recall that in Chapter 3 we develop symbolic-numeric algorithms for computing approximate topological invariants of a plane complex algebraic curve. These symbolic-numeric algorithms depend on a positive real number called regularization parameter. In this section we show that the symbolic-numeric algorithms designed in Chapter 3 compute approximate solutions, which satisfy the following property (called convergence for noisy data property): as the noise level decreases to zero and as the regularization parameter is chosen according to a specific rule (called parameter choice rule), the approximate solutions computed by the symbolic-numeric algorithms tend to the exact solutions of the considered problem.

Once the approximate Alexander polynomial of a plane curve singularity is known, the computation of the approximate delta-invariant and of the approximate genus are not anymore subject to numerical errors because we use discrete combinatorial algorithms combined with robust computational geometry algorithms for their computation. The approximate Alexander polynomial of a plane curve singularity is determined by the approximate link of the singularity. We need to analyze the numerical behavior of the link of the singularity under tiny perturbations. In this chapter, we basically prove the convergence results for the symbolic-numeric algorithm that computes the approximate local topological type of the singularity. We recall that the approximate local topological type of the singularity Q of the plane complex algebraic curve $\mathcal{C}$ is defined as the pair $\big(L_\epsilon(Q), \Delta_\epsilon(Q), \delta_\epsilon(Q)\big)$, where $L_\epsilon(Q)$ denotes the ϵ-link of the singularity Q (or the approximate link of Q), $\Delta_\epsilon(Q)$ denotes the ϵ-Alexander polynomial of $L_\epsilon(Q)$ (or the approximate Alexander polynomial of Q), and $\delta_\epsilon(Q)$ denotes the ϵ-delta-invariant of Q (or the approximate delta-invariant of Q).

We mention that the results from this section are included in the following two papers [Hodorog and Schicho, 2011a] and in [Hodorog and Schicho,].

4.1 Basic Notations

We denote by V_I the set of coefficient vectors of all the squarefree polynomials from $\mathbb{C}[z, w]$ of degree bounded by some natural number $m \in \mathbb{N} \setminus \{0\}$. The set $\mathcal{P} := \{\mathbb{Z}[t_1] \cup \mathbb{Z}[t_1, t_2] \cup ... \cup \mathbb{Z}[t_1, ..., t_k] \cup ...\}$ represents the set of all normalized Alexander polynomials either in the t_1 variable, or in the t_1, t_2 variables, or in the $t_1, t_2, ...t_k$ sequence of variables with $k \in \mathbb{N} \setminus \{0\}$, etc. We denote by i the imaginary unit. We denote by V_O the discrete set of integer coefficient vectors of all the polynomials from $\mathcal{P}$. For a polynomial $p(z, w)$ of fixed degree we denote with p its corresponding coefficient vector. The set V_I is a metric space by the Euclidean distance of coefficient vectors, denoted with $||\cdot||$. The notation $|\cdot|$ represents the absolute value function.
For $p(z, w) \in \mathbb{C}[z, w]$ we denote by:

$$M_p(z, w) := \begin{pmatrix} \partial_z p(z, w) & \partial_w p(z, w) \\ \overline{z} & \overline{w} \end{pmatrix} \tag{4.1}$$

the two-by-two matrix formed by the partial derivatives of $p(z, w)$ with respect to z and w, and by the complex conjugates $\overline{z}, \overline{w}$. We denote by *Zeroes*$(p)$ the set of zeroes of the polynomial $p(z, w)$.
In addition, we consider the usual topology on the Euclidean space $\mathbb{R}^n$, i.e. the basic open sets are the open balls.

4.2 Basic Results

We use the *Heine-Borel theorem* as a criteria for defining compact subsets of $\mathbb{R}^n$:

Theorem 12. *Every subset of $\mathbb{R}^n$ with the usual topology is compact if and only if it is closed and bounded.*

We employ the *Bolzano-Weierstrass theorem* for ensuring the existence of convergence subsequences in $\mathbb{R}^n$:

Theorem 13. *Each bounded sequence in $\mathbb{R}^n$ has a convergent subsequence.*

For our study, the *Euclidean extreme value theorem* provides us with necessary conditions for the existence of the maximum value for a continuous real-valued function:

Theorem 14. *Any continuous real-valued function on a closed and bounded subset of $\mathbb{R}^n$ attains its maximum and minimum values.*

4.3 Definitions

First we establish a general framework for handling ill-posed algebraic problems using adapted regularization principles from [Engl et al., 1996, Tikhonov and Arsenin, 1977]. We then apply these principles to Problem 1 from Chapter 3, which we treat in this thesis.
We define a well-posed problem as it was first formulated by J. Hadamard: a problem is said well-posed if: (i) there exists a solution to the problem (**existence**); (ii) the solution is unique (**uniqueness**); (iii) the solution depends continuously on the data in some given topological space (**stability**). Otherwise the problem is called ill-posed.

We consider the discontinuous function:

$$E : X \to Y, f \mapsto E(f), \tag{4.2}$$

on the metric spaces X, Y with metrics given by the Euclidean norm. The problem of computing $E(f) \in Y$ for given $f \in X$ is ill-posed as the computed output does not continuously depend on the input, i.e. the **stability** statement from the definition of well-posed problems does not hold. We define a perturbation function as follows:

Definition 61. A perturbation of $f \in X$ is defined as the function $f_- : \mathbb{R}_{>0} \to X, \delta \mapsto f_\delta$ with $||f - f_\delta|| \leq \delta$ for all $\delta \in \mathbb{R}_{>0}$. In this case f is called the exact data, f_δ the perturbed data and δ the noise level (error, tolerance).

We would like to approximate the discontinuous function E by continuous partial functions R_ϵ with the same discrete output set and with varying domains of definition. One can use the additional parameter ϵ to "move away" the input from the set where the function R_ϵ is not defined. More precisely, we define:

Definition 62. For any $\epsilon \in \mathbb{R}_{>0}$, let $U_\epsilon \subset X \times \mathbb{R}_{>0}$ be an open subset and let:

$$R_\epsilon : U_\epsilon \to Y, (f, \epsilon) \mapsto R_\epsilon(f)$$

be a continuous function. The function R_ϵ is called a regularization if there exists a bijective, monotonic function $\epsilon = \alpha(\delta), \alpha : \mathbb{R}_{>0} \to \mathbb{R}_{>0}$ with:

$$\lim_{\delta \to 0} \alpha(\delta) = 0, \tag{4.3}$$

such that for any $f \in X$, $f \in U_{\alpha(\delta')}$ for sufficiently small $\delta' \in \mathbb{R}_{>0}$, and for any perturbation function f_- with $||f - f_\delta|| \leq \delta$ for all $\delta \in \mathbb{R}_{>0}$, the following property holds:

$$\lim_{\delta \to 0} R_{\alpha(\delta)}(f_\delta) = E(f). \tag{4.4}$$

The function α is called a *parameter choice rule*, ϵ is called the *regularization parameter* and R_α is called the *regularized solution* of E. The equation (4.4) is called the *convergence for noisy data* property of R_α. The pair (R_α, α) is called a regularization method for solving the ill-posed problem E if the equations (4.3) and (4.4) are true.

For our problem, we consider X the set V_I of coefficient vectors of squarefree polynomials $p(z, w) \in \mathbb{C}[z, w]$ of degree bounded by some natural number $m \in \mathbb{N} \setminus \{0\}$ and Y the set V_O of integer coefficient vectors of normalized Alexander polynomials. In addition, we let:

$$E : V_I \to V_O, f \mapsto E(f) \tag{4.5}$$

be the exact algorithm for computing the Alexander polynomial of a plane curve singularity. Since V_O is a discrete set, the function E is discontinuous. Therefore, the problem of computing the Alexander polynomial $E(f) \in V_O$ for given $f \in V_I$ is ill-posed.

For every $\epsilon \in \mathbb{R}_{>0}$, we denote by:

$$A_\epsilon : U \subset V_I \times \mathbb{R}_{>0} \to V_O, (p, \epsilon) \mapsto A_\epsilon(p) \tag{4.6}$$

the symbolic-numeric algorithm that computes the ϵ-Alexander polynomial $A_\epsilon(p)$ for given $(p, \epsilon) \in V_I \times \mathbb{R}_{>0}$, as described in Chapter 3. This polynomial arises as the intersection of the sphere S_ϵ with the curve $\mathcal{C}$ defined by p. We notice that A_ϵ is a partial function, because it is not defined in case the intersection $S_\epsilon \cap \mathcal{C}$ has singularities. Still, the function A_ϵ is continuous in its domain of definition denoted by U.

We wish to show that A_ϵ is a regularization function for every $(p, \epsilon) \in U \subset V_I \times \mathbb{R}_{>0}$. Therefore, from Definition 62 we need to find a parameter choice rule $\epsilon = \alpha(\delta)$ with property (4.3) and that satisfies equation (4.4). Consequently, the pair (A_α, α) would be a regularization method for solving the ill-posed Problem 1.

4.4 Convergence Results

In this subsection, we present the lemmas and the theorems that we formulate to prove the convergence for noisy data property of the algorithm A_ϵ considered in (4.6). For constructing the proofs of our lemmas and theorems, we use fundamental notions and results from algebraic geometry and topology as presented in [Bochnak et al., 1998, Marker, 2002] and respectively in [Mumkres, 2000].

First of all, we set the general mathematical setting required for our study. Let $f(z,w) \in \mathbb{C}[z,w]$ be an arbitrary but fixed squarefree polynomial with exact coefficients (i.e. integer numbers of rational numbers). For simplicity we denote $f_\delta(z,w) =: g(z,w) \in \mathbb{C}[z,w]$ with $||g-f|| \leq \delta$. We denote with S_K the sphere centered in $(0,0)$ of radius K, and with B_K the open ball centered in $(0,0)$ of radius K. Moreover, L_K denotes the K-link of the singularity $(0,0)$ of the plane complex algebraic curve defined by the polynomial $f(z,w) \in \mathbb{C}[z,w]$.

We now introduce and we prove some lemmas, which are necessary for our study. We make use of the isomorphism of $\mathbb{C}$ with the set of 2×2 matrices of the form $\begin{pmatrix} x & -y \\ y & x \end{pmatrix}$.

Lemma 2. *Let A be a 4×4 matrix defined over the real numbers as follows:*

$$\begin{pmatrix} x & -y & u & -v \\ y & x & v & u \\ a & b & c & d \\ -b & a & -d & c \end{pmatrix}. \tag{4.7}$$

In addition, consider B to be a 2×2 matrix defined over the complex numbers in the following way:

$$\begin{pmatrix} x+iy & u+iv \\ a-ib & c-id \end{pmatrix}. \tag{4.8}$$

Then $rank(A) = 2rank(B)$.

Proof. We represent the matrices A, B as linear maps in the following way:

$$A : \mathbb{R}^4 \to \mathbb{R}^4$$
$$x \in \mathbb{R}^4 \mapsto Ax \in \mathbb{R}^4$$

and

$$B : \mathbb{C}^2 \to \mathbb{C}^2$$
$$z \in \mathbb{C}^2 \mapsto Bz \in \mathbb{C}^2.$$

We notice that the following equalities are true:

$$\begin{aligned} rank(A) &= dim_{\mathbb{R}}\big(A(\mathbb{R}^4)\big) \text{ and} \\ rank(B) &= dim_{\mathbb{C}}\big(B(\mathbb{C}^2)\big), \end{aligned} \tag{4.9}$$

where $dim_{\mathbb{R}}(A(\mathbb{R}^4))$ represents the real dimension of the image of A, whereas $dim_{\mathbb{C}}(B(\mathbb{C}^2))$ denotes the complex dimension of the image of B. Moreover, we observe that

$$dim_{\mathbb{C}}\big(B(\mathbb{C}^2)\big) = dim_{\mathbb{C}}\big(B(\mathbb{R}^4)\big). \tag{4.10}$$

Hence, from equalities (4.9) and (4.10) we obtain:

$$\begin{aligned} rank(B) = dim_{\mathbb{C}}\big(B(\mathbb{C}^2)\big) = \\ = dim_{\mathbb{C}}\big(B(\mathbb{R}^4)\big) = \frac{dim_{\mathbb{R}}(A(\mathbb{R}^4))}{2} = \frac{rank(A)}{2}, \end{aligned}$$

and thus the lemma is proved. □

Lemma 3. *We consider the matrices A and B defined as in Lemma 2. Then* $\det A = |\det B|^2$.

Proof. By straightforward computation with a computer algebra system the lemma is proved. We first compute the determinant of A :

$$\begin{aligned}\det A = \det \begin{pmatrix} x & -y & u & -v \\ y & x & v & u \\ a & b & c & d \\ -b & a & -d & c \end{pmatrix} = \\ = a^2u^2 + b^2u^2 + a^2v^2 + b^2v^2 - \\ -2acux - 2bdux - 2bcvx + 2advx + \\ +c^2x^2 + d^2x^2 + 2bcuy - 2aduy - \\ -2acvy - 2bdvy + c^2y^2 + d^2y^2.\end{aligned} \tag{4.11}$$

Secondly, we compute the determinant of B and we get:

$$\begin{aligned}\det B = \det \begin{pmatrix} x+iy & u+iv \\ a-ib & c-id \end{pmatrix} = \\ = -au - bv + cx + dy + i(bu - av - dx + cy).\end{aligned} \tag{4.12}$$

We now compute the absolute value of the determinant of B from equality (4.12) and we obtain:

$$\begin{aligned}|\det B|^2 = \\ = \left(\sqrt{(-au - bv + cx + dy)^2 + (bu - av - dx + cy)^2}\right)^2 = \\ = (-au - bv + cx + dy)^2 + (bu - av - dx + cy)^2 = \\ = a^2u^2 + b^2u^2 + a^2v^2 + b^2v^2 - \\ -2acux - 2bdux - 2bcvx + 2advx + \\ +c^2x^2 + d^2x^2 + 2bcuy - 2aduy - \\ -2acvy - 2bdvy + c^2y^2 + d^2y^2.\end{aligned} \tag{4.13}$$

From equalities (4.11) and (4.13) we obtain that $\det A = |\det B|^2$ and thus the lemma is proved. □

Lemma 4. *We consider the matrices A and B as defined in Lemma 2. We consider the following 3×4 matrix defined over the real numbers:*

$$J = \begin{pmatrix} x & -y & u & -v \\ y & x & v & u \\ a & b & c & d \end{pmatrix}. \tag{4.14}$$

Then $rank(J) = 3$ if and only if $rank(A) = 4$ (i.e. $\det A \neq 0$) and equivalently $rank(B) = 2$ (i.e. $\det B \neq 0$).

Proof. First of all, we assume that $rank(J) = 3$ and we prove that $rank(A) = 4$. We know that $rank(J) = 3$ and that $rank(J) < rank(A)$. In addition, we know that $rank(A)$ is even and that A is a 4×4 matrix. Hence, $rank(A) = 4$.

We now assume that $rank(A) = 4$ (i.e. $\det A \neq 0$) and we prove that $rank(J) = 3$. Since $rank(A) = 4$, it follows that $\det A \neq 0$. We now use the determinant expansion by minors to compute $\det A$ and we obtain:

$$\det A = b \cdot M_{41} + a \cdot M_{42} + d \cdot M_{43} + c \cdot M_{44} \neq 0, \tag{4.15}$$

where M_{4j} with $j \in \{1,2,3,4\}$ represents the minor of A obtained by taking the determinant of A with row 4 and column j erased. It follows that at least one of the minors $M_{41}, M_{42}, M_{43}, M_{44}$ from equality (4.15) is different from 0. Hence, at least one 3×3 minor of J is nonzero. We obtain that $rank(J) = 3$ and thus the lemma is proved. □

We formulate and we prove the following important proposition:

Proposition 3. *We consider the plane complex algebraic curve $\mathcal{C} \subset \mathbb{C}^2$ with a singularity in the origin $O(0,0) \in \mathbb{C}^2$, defined by the squarefree polynomial $f(z,w) \in \mathbb{C}[z,w]$. We replace $z = a + ib, w = c + id$ in $f(z,w)$ and we obtain*

$$f(a,b,c,d) = u(a,b,c,d) + iv(a,b,c,d),$$

where $u(a,b,c,d), v(a,b,c,d) \in \mathbb{R}[a,b,c,d]$. We represent $\mathcal{C}$ as a 2-dimensional object in $\mathbb{R}^4$:

$$\mathcal{C} = \{(a,b,c,d) \in \mathbb{R}^4 \mid u(a,b,c,d) = v(a,b,c,d) = 0\},$$

and we consider the 3-dimensional sphere of small radius K centered in the origin $O(0,0)$:

$$S_\epsilon(O) = \{(a,b,c,d) \in \mathbb{R}^4 \mid a^2 + b^2 + c^2 + d^2 = K^2\}.$$

We consider the following system of polynomial equations:

$$\begin{cases} u(a,b,c,d) = 0 \\ v(a,b,c,d) = 0 \\ a^2 + b^2 + c^2 + d^2 = K^2 \end{cases}, \tag{4.16}$$

which represents the intersection of $Zeroes(f) \cap S_K(O)$. We consider the Jacobian J_I of the system (4.16):

$$J_I = \begin{pmatrix} \frac{\partial u}{\partial a} & -\frac{\partial u}{\partial b} & \frac{\partial u}{\partial c} & -\frac{\partial u}{\partial d} \\ \frac{\partial v}{\partial a} & \frac{\partial v}{\partial b} & \frac{\partial v}{\partial c} & \frac{\partial v}{\partial d} \\ a & b & c & d \end{pmatrix}, \tag{4.17}$$

and the following matrices:

$$S_1 = \begin{pmatrix} \frac{\partial u}{\partial a} & -\frac{\partial u}{\partial b} & \frac{\partial u}{\partial c} & -\frac{\partial u}{\partial d} \\ \frac{\partial v}{\partial a} & \frac{\partial v}{\partial b} & \frac{\partial v}{\partial c} & \frac{\partial v}{\partial d} \\ a & b & c & d \\ -b & a & -d & c \end{pmatrix}, \tag{4.18}$$

$$S_2 = \begin{pmatrix} \frac{\partial f}{\partial z} & \frac{\partial f}{\partial w} \\ \overline{z} & \overline{w} \end{pmatrix}, \tag{4.19}$$

where $\overline{z}, \overline{w}$ represent the complex conjugates of z, w. Then the following statements are true:

1. $rank(S_1) = 2rank(S_2)$.
2. $\det S_1 = |\det S_2|^2$.

3. $rank(S_1) = 4$ *if and only if* $rank(J_I) = 3$.

Proof. From the Cauchy-Riemann equations we know:

$$\frac{\partial u}{\partial a} = \frac{\partial v}{\partial b}, \frac{\partial u}{\partial b} = -\frac{\partial v}{\partial a}$$

$$\frac{\partial u}{\partial c} = \frac{\partial v}{\partial d}, \frac{\partial u}{\partial d} = -\frac{\partial v}{\partial c}.$$

Consequently, the matrices J_I, S_1, S_2 are of the same form as the matrices J, A, B from Lemma 2, Lemma 3 and Lemma 4. Based on Lemma 2 it follows that $rank(S_1) = 2rank(S_2)$. From Lemma 3 we obtain that $\det S_1 = |\det S_2|^2$. In addition, by Lemma 4 we get that $rank(S_1) = 4$ if and only if $rank(J_I) = 3$ and thus the proposition is proved. □

Based on Proposition 3, we now formulate and we prove the following proposition, which is essential for our proof concerning the convergence for noisy data property of the designed symbolic-numeric algorithm from Chapter 3:

Proposition 4. *We consider the same setting as in Proposition 3. Then the intersection* $Zeroes(f) \cap S_K(O)$ *has no singularities if and only if the equations*

$$f(z, w) = det(M_f)(z, w) = |z|^2 + |w|^2 - K^2 = 0 \tag{4.20}$$

have no common solutions.

Proof. Based on equality (4.19) and on notation (4.1) we get:

$$\det S_2 = \det M_f. \tag{4.21}$$

Firstly, we suppose that $Zeroes(f) \cap S_K(O)$ has no singularities and we show that the equations given in (4.20) have no common solutions. Since $Zeroes(f) \cap S_K(O)$ has no singularities, from Proposition 3 we get that $rank(J_I) = 3$, which implies $rank(S_1) = 4$. It follows that the system (4.20) has no common solutions. Secondly, we assume that the equations given defined in (4.20) have no common solutions. From Proposition 3, we get that $rank(S_1) = 4$. Consequently, $Zeroes(f) \cap S_K(O)$ has no singularities. □

Let $K > 0$ be such that $S_\epsilon(O) \cap \mathcal{C}$ has no singularities for all $\epsilon, 0 < \epsilon \leq K$. Such K exists by Theorem 10 of Milnor. Then the following system

$$f(z, w) = \det(M_f)(z, w) = 0 \tag{4.22}$$

has no common solution except for $(0, 0)$ in the closed ball $B_K := \left\{(z, w) \in \mathbb{C}^2 : \left(|z|^2 + |w|^2\right)^{1/2} \leq K\right\}$ of radius K around $(0, 0) \in \mathbb{C}^2$, by Proposition 4. Note that $(0, 0)$ is a common solution because

$$f(0, 0) = \det(M_f)(0, 0) = 0. \tag{4.23}$$

In order to prove the convergence for noisy data property of A_ϵ we require a preliminary lemma.

Lemma 5. *There exists* $N > 0$ *such that for all* $\delta > 0$, *and for all* g *with* $||g - f|| \leq \delta$ *there exists no zero for the system of polynomial equations determined by* $g(z, w) = \det(M_g)(z, w) = 0$ *whose length is greater than* $\delta^{1/N}$ *and less than* K.

Proof. In order to prove Lemma 5 we prove the following equivalent statement:

$$\begin{gathered}\exists N>0\ \forall\delta>0\ \forall g:||g-f||\leq\delta\ \forall(z,w):\\ \Big(g(z,w)=\det(M_g)(z,w)=0\text{ and}\\ \Big(|z|^2+|w^2|\Big)^{1/2}\leq K\Big)\Rightarrow\Big(|z|^2+|w^2|\Big)^{1/2}\leq\delta^{1/N}.\end{gathered}\tag{4.24}$$

We take $\delta>0$ and g with $||g-f||\leq\delta$.

Step 1. We introduce the set Z_δ of "special" zeroes of g as follows:

$$\begin{aligned}Z_\delta=\Big\{\Big((z,w),g\Big):&||g-f||\leq\delta,\\ &g(z,w)=\det(M_g)(z,w)=0,\\ &\Big(|z|^2+|w|^2\Big)^{1/2}\leq K\Big\}.\end{aligned}\tag{4.25}$$

We notice that $Z_\delta\subseteq B_K\times V_I$. From equality (4.23) it follows that $Z_\delta\neq\emptyset$ since it contains $\Big((0,0),f\Big)$.

We consider the following function:

$$\begin{gathered}\tau:B_K\times V_I\to\mathbb{R}_{\geq0}\\ \Big((z,w),g\Big)\mapsto\tau\Big((z,w),g\Big)=\Big(|z|^2+|w|^2\Big)^{1/2}.\end{gathered}\tag{4.26}$$

For identifying compact sets we use Theorem 12 of Heine-Borel, which states that every subset of $\mathbb{R}^n$ with the usual topology is compact if and only if it is closed and bounded. Consequently, we observe that $Z_\delta\subset B_K\times V_I$ is compact and that τ is a continuous real valued positive function. In addition, we have that $Z_\delta\neq\emptyset$ and $Z_\delta\subseteq B_K\times V_I$. From Theorem 14 on the Euclidean extreme value of real-valued functions, we know that any continuous real-valued function on a compact subset of $\mathbb{R}^n$ attains its maximum and minimum values. Hence, we obtain that $\tau(Z_\delta)$ attains its maximum value.

We use the maximum of $\tau(Z_\delta)$ to define the function:

$$\begin{gathered}\beta:\mathbb{R}_{>0}\to\mathbb{R}_{\geq0}\\ \delta\mapsto\beta(\delta)=max\ \{\tau(a):a\in Z_\delta\}.\end{gathered}\tag{4.27}$$

We notice that β is a monotonic, semialgebraic function.

Step 2. We now prove the convergence of β:

$$\lim_{\delta\to0}\beta(\delta)=0.\tag{4.28}$$

We consider $(\delta_n)_n\subset\mathbb{R}_{>0}$ an arbitrary but fixed sequence with $\lim\limits_{n\to\infty}\delta_n=0$. We assume that $(\delta_n)_n$ is monotonic with $\delta_n\leq\delta_1$ for all $n\in\mathbb{N}\setminus\{0\}$. To prove equality (4.28) we need to prove that $\lim\limits_{n\to\infty}\beta(\delta_n)=0$.

We define the sequence $\Big((z_n,w_n),g_n\Big)_n\subseteq B_K\times V_I$ such that $\Big((z_n,w_n),g_n\Big)\in Z_{\delta_n}$ and $\tau\Big((z_n,w_n),g_n\Big)=\beta(\delta_n)$ for all $n\in\mathbb{N}$. In addition, we replace V_I in the proof by $V_I=\{g:||g-f||\leq\delta_1\}$. In this case, all the assumptions formulated previously are correct.

Under these assumptions we have that $B_K\times V_I$ is compact, by Theorem 12 of Heine-Borel. From Theorem 13 of Bolzano-Weierstrass on compact sets, we know that each bounded

sequence in $\mathbb{R}^n$ has a convergent subsequence. Consequently, we obtain that the sequence $\Big((z_n, w_n), g_n\Big)_n \subseteq B_K \times V_I$ has a convergent subsequence.

We take this convergent subsequence to be the sequence $\Big((z_{n_m}, w_{n_m}), g_{n_m}\Big)_m$ specified by $(n_m)_m$ with the following property:

$$\begin{cases} \lim\limits_{m\to\infty} z_{n_m} = \tilde{z} \\ \lim\limits_{m\to\infty} w_{n_m} = \tilde{w} \\ \lim\limits_{m\to\infty} g_{n_m} = \tilde{g} \end{cases}. \tag{4.29}$$

Since $\Big((z_{n_m}, w_{n_m}), g_{n_m}\Big) \in Z_{\delta_{n_m}}$ for all $m \in \mathbb{N}$, we get:

$$||g_{n_m} - f|| \leq \delta_{n_m} \text{ and} \tag{4.30}$$

$$g_{n_m}(z_{n_m}, w_{n_m}) = \det(M_{g_{n_m}})(z_{n_m}, w_{n_m}) = 0. \tag{4.31}$$

Under the hypothesis that $\lim\limits_{m\to\infty} \delta_{n_m} = 0$ and from inequality (4.30), we conclude that $\tilde{g} = f$.

Moreover, it follows that

$$f(\tilde{z}, \tilde{w}) = \det(M_f)(\tilde{z}, \tilde{w}) = 0. \tag{4.32}$$

From equalities (4.22) and (4.23) we know that $(0, 0)$ is the only zero for the system of polynomial equations (4.32) and thus $(\tilde{z}, \tilde{w}) = (0, 0)$.

Consequently, we obtain $\lim\limits_{m\to\infty} \beta(\delta_{n_m}) = 0$. Since $\big(\beta(\delta_n)\big)_n$ is also a monotonic sequence we obtain that $\lim\limits_{n\to\infty} \beta(\delta_n) = 0$ and thus we have proved the convergence of β.

Step 3. Finally, we show that the function β is bounded from above by $\delta^{1/N}$. From relation (4.28) we obtain that the function β is bounded from above. We now have to show that β is smaller or equal that $\delta^{1/N}$.

We use the following theorem for estimating the rate of growth of a semialgebraic function of one variable:

Theorem 15. *([Bochnak et al., 1998]) Let $f : (a, \infty) \to \mathbb{R}$ be a semialgebraic function (not necessarily continuous). There exists $b \geq a$ and an integer $N \in \mathbb{N}$ such that $|f(x)| \leq x^N$ for all $x \in (b, \infty)$.*

Moreover, we employ the following theorem for ensuring the piecewise continuity of a semialgebraic function:

Theorem 16. *([Marker, 2002]) Let F be a real closed field and $f : F \to F$ be a semialgebraic function. Then, we can partition F into $I_1 \cup ...I_m \cup X$, where X is finite and I_j are pairwise disjoint open intervals with endpoints in $F \cup \{\pm\infty\}$ such that f is continuous on each I_j with $j \in \{1, ..., m\}$ and $m \in \mathbb{N}$.*

Since β is a semialgebraic function, we obtain from Theorem 16 that β is piecewise continuous on open intervals. We consider β_r the restriction of β to the first open interval.

Case 1. The restriction β_r is 0 in the first open interval. The lemma is true as $(0, 0)$ is the only "special" zero.

Case 2. The restriction β_r is not 0 in the first open interval. Because β_r is continuous and monotonic, it follows that the restriction β_r is bijective. Using the inverse β_r^{-1} we define the function γ as follows:

$$\begin{aligned} \gamma : \mathbb{R}_{>0} &\to \mathbb{R}_{>0} \\ \delta \mapsto \gamma(\delta) &= \frac{1}{\beta_r^{-1}(\delta^{-1})}. \end{aligned} \tag{4.33}$$

Since γ is a semialgebraic function, we apply Theorem 15 and we obtain that there exists $N \in \mathbb{N} \setminus \{0\}$ and $b \in \mathbb{R}_{>0}$ such that:

$$\gamma(\delta) \leq \delta^N, \tag{4.34}$$

for all $\delta > b \in \mathbb{R}_+$. We substitute δ with δ^{-1} in inequality (4.34) and we obtain that there exists $N \in \mathbb{N} \setminus \{0\}$ and $\eta \in \mathbb{R}_{>0}$ such that:

$$\gamma(\delta^{-1}) \leq \delta^{-N}, \tag{4.35}$$

for all $\delta < \eta = b^{-1}$. We rewrite inequality (4.35) using the definition (4.33) of γ and by eliminating the denominators we obtain:

$$\delta^N \leq \beta_r^{-1}(\delta). \tag{4.36}$$

We compose the inequality (4.36) in both sides with β_r. By using $\beta_r \circ \beta_r^{-1} = id_{\mathbb{R}_{>0}}$ we get:

$$\beta_r(\delta^N) \leq \delta. \tag{4.37}$$

By substituting δ with $\delta^{1/N}$ in inequality (4.37) we obtain an upper bound for the semialgebraic function β:

$$\beta_r(\delta) \leq \delta^{1/N},$$

and thus the lemma is proved. □

We use Lemma 5 as a tool for proving the convergence for noisy data statement (4.4) and for ensuring the existence of a proper parameter choice rule (4.3) for the algorithm A_ϵ defined in relation (4.6). This convergence statement is given by the following theorem:

Theorem 17. *There exists $N > 0$ and $\eta \in \mathbb{R}_{>0}$ such that for all $\delta > 0$ with $\delta < \eta$, for all g with $||g - f|| \leq \delta$ and for all $\epsilon \in [\delta^{1/N}, K]$, the following property holds: $A_\epsilon(g) = E(f)$.*

Proof. We take $N > 0$, $\eta \in \mathbb{R}_+$ and $\epsilon = \beta_r(\delta) \leq \delta^{1/N}$ given by Lemma 5. We show that $A_\epsilon(g) = E(f)$.

From Theorem 10 of Milnor we know that $A_\epsilon(f) = E(f)$.

We construct an isotopy between the links $Zeroes(f) \cap S_\epsilon(0)$ and $Zeroes(g) \cap S_\epsilon(0)$ as follows:

$$\begin{aligned} &g_t : \mathbb{C}^2 \times [0,1] \to \mathbb{C} \\ &(z,w) \to g_t(z,w) = tf(z,w) + (1-t)g(z,w), \end{aligned} \tag{4.38}$$

where g_t is a continuous function for all $0 \leq t \leq 1$ with $g_0 = g$ and $g_1 = f$.

To prove the theorem, it suffices to show that $A_\epsilon(g_t)$ is an ϵ-algebraic link, i.e. that $Zeroes(g_t) \cap S_\epsilon(O)$ has no singularities, i.e. that the system of polynomial equations determined by $g_t(z,w) = det(M_{g_t})(z,w) = 0$ has no zero in B_ϵ with $\epsilon = \beta_r(\delta) \leq \delta^{1/N}$. This statement is true by Lemma 5. □

From Theorem 17 it follows that $\epsilon = \delta^{1/N}$ is a parameter choice rule for A_ϵ, for which the convergence for noisy data statement (4.4) of A_ϵ holds. Still, this parameter choice rule depends on N which is unknown. The following lemma provides us with an upper bound for $\delta^{1/N}$ which is independent on N:

Lemma 6. *For all $N > 0$ there exists $\theta \in \mathbb{R}_{>0}$ such that for all $\delta > 0$ with $\delta < \theta$, the inequality $\delta^{1/N} \leq \dfrac{1}{|ln\delta|}$ is true.*

Proof. We prove Lemma 6 by basic calculus and by using l'Hôpital rule. We take $N > 0$, $\eta \in \mathbb{R}_+$ and $\epsilon = \beta_r(\delta) \leq \delta^{1/N}$ given by Lemma 5. We prove that:

$$|ln\delta| \leq \delta^{-1/N}. \tag{4.39}$$

By replacing $x = \delta^{-1}$ in (4.39), we need to prove that:

$$lnx < x^{1/N}, \tag{4.40}$$

for all $x \in \mathbb{R}_{>0}$. By replacing again $x = e^y$ in inequality (4.40), we need to show that:

$$y < e^{y/N}, \tag{4.41}$$

for all $y \in \mathbb{R}_{>0}$. It suffices thus to prove the inequality:

$$1 < \frac{e^{y/N}}{y} \text{ for all } y \in \mathbb{R}_{>0}. \tag{4.42}$$

We notice that $\lim\limits_{y\to\infty} \dfrac{e^{y/N}}{y} = \dfrac{\infty}{\infty}$. We thus apply l'Hôpital rule and we use derivatives to evaluate this indeterminate limit. We get that:

$$\lim_{y\to\infty} \frac{e^{y/N}}{y} = \lim_{y\to\infty} \frac{(e^{y/N})'}{(y)'} = \frac{1}{N} \cdot e^{\infty/N} = \infty. \tag{4.43}$$

The inequality (4.42) follows from equality (4.43), and thus the lemma is proved. □

The preceding two lemmas allow us to formulate the following theorem concerning the existence of a parameter choice rule for A_ϵ which only depends on the given $\delta \in \mathbb{R}_{>0}$:

Theorem 18. *The function $\alpha : \mathbb{R}_{>0} \to \mathbb{R}_{>0}, \alpha(\delta) = \dfrac{1}{|ln\delta|}$ is a parameter choice rule, i.e.*

$$\lim_{\delta\to 0} A_{\alpha(\delta)}(f_\delta) = E(f). \tag{4.44}$$

The theorem is true based on Lemma 5, Theorem 17 and Lemma 6.

Remark 15. The parameter choice rule indicates that the "degree of ill-posed-ness" is rather high (cf with linear regularization theory [Tikhonov and Arsenin, 1977], where $\alpha(\delta) = \delta^{1/2}$ frequently occurs). For fixed input instance f, the smallest function $\alpha : \mathbb{R}_{>0} \to \mathbb{R}_{>0}$ such that (noisy convergence) is true is equal to the function β from Lemma 5. The choice of α was done in order to ensure that α dominates β for every possible f. Here is a series of examples that show that a semi-algebraic parameter choice rule cannot be used as a choice rule.

Example. Let $n > 0$ be an integer. Let $f(z, w) = z^2 - w^{n+2}$. We consider the perturbation $g(z, w) = f_\delta(z, w) = z^2 - w^{n+2} + \delta w^2$, for $\delta \in (0, 1)$. Then we have a special zero of (g, M_g)

at $(z, w) = (0, \delta^{1/n})$. A closer analysis shows that the ϵ-link of g is the Hopf link for every sphere with radius less than $\delta^{1/n}$, while the link of f is equal to the torus link $(2, n+2)$. Consequently, $\beta(\delta) > \delta^{1/n}$ for this choice of f. Since n can be arbitrary, no function which is dominated by a function of the from $\delta \mapsto \delta^{1/m}$ for some m can be chosen as a parameter choice rule.

Chapter 5

Software: The GENOM3CK library

GENOM3CK[1] (**GEN**us c**OM**putation of **P**lane **C**omplex algebrai**C** **C**urves using **K**not theory) is a library designed mainly for computing the approximate genus of a plane complex algebraic curve defined by a bivariate squarefree complex polynomial with coefficients of limited accuracy, i.e. the coefficients may be exact data (i.e. integer or rational numbers) or inexact data (i.e. numerical values). For the inexact data we associate a positive real number $\delta \in \mathbb{R}_{>0}$, which measures the error level (called also the noise or the tolerance level) in the coefficients of the defining polynomial of the input plane complex algebraic curve. The library GENOM3CK basically contains the implementation of the symbolic-numeric algorithms from Chapter 3, algorithms that we developed for solving Problem 1 from Chapter 3. We sometimes refer to the symbolic-numeric algorithms developed in Chapter 3 as the approximate algorithms. We mention that part of the results presented in this Chapter are also included in the paper [Hodorog et al., 2010a].

5.1 Description of the Library

5.1.1 Main Functionality of the Library

In this subsection we remember the main steps of the symbolic-numeric algorithms from Chapter 3 implemented in the library GENOM3CK. We recall that the symbolic-numeric algorithms from Chapter 3 implemented in the library GENOM3CK are based on knot theory and on the topology analysis of the singularities of a plane complex algebraic curve. By using the symbolic-numeric algorithms from Chapter 3 we compute approximate invariants of a plane complex algebraic curve and its singularities. We recall that computing the approximate invariants of a plane complex algebraic curve and its singularities means computing the invariants of the curve depending on an input parameter $\epsilon \in \mathbb{R}_{>0}$. We remember that for designing the symbolic-numeric algorithms from GENOM3CK we ba-

[1]The acronym GENOM3CK, chosen for the name of the library that we develop to implement the symbolic-numeric algorithms for plane complex algebraic curves using knot theory, gives some intuition behind the deep connection between mathematical knot theory and biology. The name GENOM3CK basically represents the acronym of two words, i.e. *genome* and *trick*. In molecular biology, the genome represents all the hereditary material possessed by an organism, and thus has various implications in the past or present discoveries from medicine. For instance, F. H. Crick and J. D. Watson were jointly awarded the Nobel Prize for Medicine in 1962 for their theory concerning the basic structure of the DNA, see [Crick and Watson, 1953]. It is the hope of the author to see/to witness similar fruitful results arising from the interplay of knot theory and biology with practical applications in the field of contemporary medicine.

sically use Milnor's theory [Milnor, 1968] and Yamamoto's result [Yamamoto, 1984], who showed that the Alexander polynomial is a complete invariant for all the algebraic links up to an ambient isotopy, i.e. the Alexander polynomial distinguishes all algebraic links up to ambient isotopy. We mention that an algebraic link is a link that is equivalent to the link of a plane curve singularity. We add that the computation of the approximate genus of a plane complex algebraic curve reduces to the computation of the approximate delta-invariant of each singularity of the plane complex algebraic curve. Using subdivision methods from the algebraic geometric modeler Axel, we compute a piecewise linear approximation of each approximate link of a plane curve singularity as a 3-dimensional graph data structure. We recall that a 3-dimensional graph data structure is a set of vertices together with their Euclidean coordinates in the 3-dimensional Euclidean space, and a set of edges connecting them. We design computational geometry and combinatorial algorithms for the computation of the approximate Alexander polynomial of each approximate algebraic link, i.e. an adapted version of the Bentley-Ottmann algorithm [Berg et al., 2008]. From the approximate Alexander polynomial of each approximate algebraic link we compute the approximate delta-invariant of each singularity.

The symbolic-numeric algorithms implemented in the library GENOM3CK take as *input* the following parameters:

(i) a bivariate squarefree complex polynomial $p(z, w) \in \mathbb{C}[z, w]$ with exact and inexact coefficients, which defines a plane complex algebraic curve $\mathcal{C}$. We add that for the inexact data in the polynomial $p(z, w)$ we associate a positive real number $\delta \in \mathbb{R}_{>0}$, which measures the error level in the coefficients;

(ii) a positive real number $\epsilon \in \mathbb{R}_{>0}$, which represents the radius of the sphere centered around the singular point Q of $\mathcal{C}$, sphere that we intersect with the plane complex algebraic curve $\mathcal{C}$ to compute the ϵ-link of the singularity Q. We mention that the input parameter $\epsilon \in \mathbb{R}_{>0}$ is also called the regularization parameter as discussed in Chapter 4;

(iii) a subset of $\mathbb{R}^3$ denoted with $B = [-a, a] \times [-b, b] \times [-c, c] \subset \mathbb{R}^3$, with $a, b, c \in \mathbb{N} \setminus \{0\}$, for the x, y and z coordinates of the three-dimensional space $\mathbb{R}^3$. We call the subset $B \subset \mathbb{R}^3$ a box. We add that the box B is required for computing the piecewise linear approximation of the ϵ-link of each singularity of $\mathcal{C}$. The piecewise linear approximation for each ϵ-link of a plane curve singularity is computed as a 3-dimensional graph data structure.

Together with its main functionality to compute the approximate genus of a plane complex algebraic curve, the library GENOM3CK computes as *output* other important information about the plane complex algebraic curve as follows:

(1) the set of distinct real numerical singularities of the plane complex algebraic curve $\mathcal{C}$, singularities that are computed in the projective real plane. We recall that a numerical singularity of $\mathcal{C}$ is a point Q such that $p(Q)$, $\partial_z p(Q)$ and $\partial_w p(Q)$ are small in comparison with the coefficients of the polynomial $p(z, w)$;

(2) the approximate link of each singularity of the plane complex algebraic curve $\mathcal{C}$, i.e. the ϵ-link denoted with L_ϵ of each singularity of $\mathcal{C}$. For each numerical singularity Q of the plane complex algebraic curve $\mathcal{C}$, the library outputs the approximate link of Q, that is the ϵ-link of Q. We recall that the ϵ-link L_ϵ of the singularity Q is computed as a smooth and closed space algebraic curve, implicitly defined as the intersection of two space algebraic surfaces S_1, S_2 with the defining polynomials $g_\epsilon(x, y, z), h_\epsilon(x, y, z) \in \mathbb{R}^3$. We mention that the library basically outputs the two polynomials $g_\epsilon(x, y, z), h_\epsilon(x, y, z)$ that

define as their intersection the ϵ-link L_ϵ. The library also outputs the 3-dimensional visualization of the piecewise linear approximation of the ϵ-link L_ϵ as a 3-dimensional graph data structure denoted $Graph(L_\epsilon)$. In addition, the library outputs the 3-dimensional visualization of the two surfaces S_1, S_2. We add that the two surfaces S_1, S_2 are part of the Milnor fibration of the singularity Q of the plane complex algebraic curve $\mathcal{C}$;

(3) the approximate Alexander polynomial of each approximate link of each numerical singularity of the plane complex algebraic curve;

(4) the approximate Milnor number of each numerical singularity of the plane complex algebraic curve;

(5) the approximate number of branches of each numerical singularity of the plane complex algebraic curve;

(6) the approximate delta-invariant of each numerical singularity of the plane complex algebraic curve;

(7) the approximate genus of the plane complex algebraic curve;

(8) the approximate Euler characteristic of the Riemann surface attached to the resolution of singularities of the plane complex algebraic curve.

(9) the set of several invariants and properties from knot theory of each approximate link of each numerical singularity of the plane complex algebraic curve. The library basically outputs the genus, the linking number, the unknotting number and the determinant of each approximate link of each singularity of the plane complex algebraic curve. Moreover, the library also provides an answer to the problem of deciding the tricolorability property of each approximate link of each singularity of the plane complex algebraic curve.

We recall that computing the approximate invariants of a plane complex algebraic curve $\mathcal{C}$ basically means computing the invariants of $\mathcal{C}$ and its singularities depending on the input parameter $\epsilon \in \mathbb{R}_{>0}$. For more details concerning the computation of the approximate invariants of a plane complex algebraic curve $\mathcal{C}$, the reader should consult Chapter 3. In Figure 5.1 we visualize the input-output specification of the GENO3CK library.

The library GENOM3CK contains thus symbolic-numeric algorithms for computing different approximate invariants of a plane complex algebraic curve and its singularities. The output displayed by the library GENOM3CK represents the properties of a plane complex algebraic curve and its singularities. Basically, the output of GENOM3CK is divided into five types of properties as follows: geometric properties, invariant properties, algebraic properties, topological properties and knot theory properties attached to a plane complex algebraic curve and its singularities. In addition, the output of the library contains also the computational time required for performing each type of properties attached to a plane complex algebraic curve and its singularities. In Figure 5.2 we introduce a table that contains the output of GENOM3CK divided into geometric properties, invariant properties, algebraic properties, topological properties and knot theory properties of a plane complex algebraic curve and its singularities.

5.1.2 Short History of the Library

The library GENOM3CK is implemented in the Axel [Wintz et al., 2006] free algebraic geometric modeler and in the Mathemagix [van der Hoeven et al., 2002] free computer algebra system. We mention that the Axel algebraic geometric modeler was originally developed in

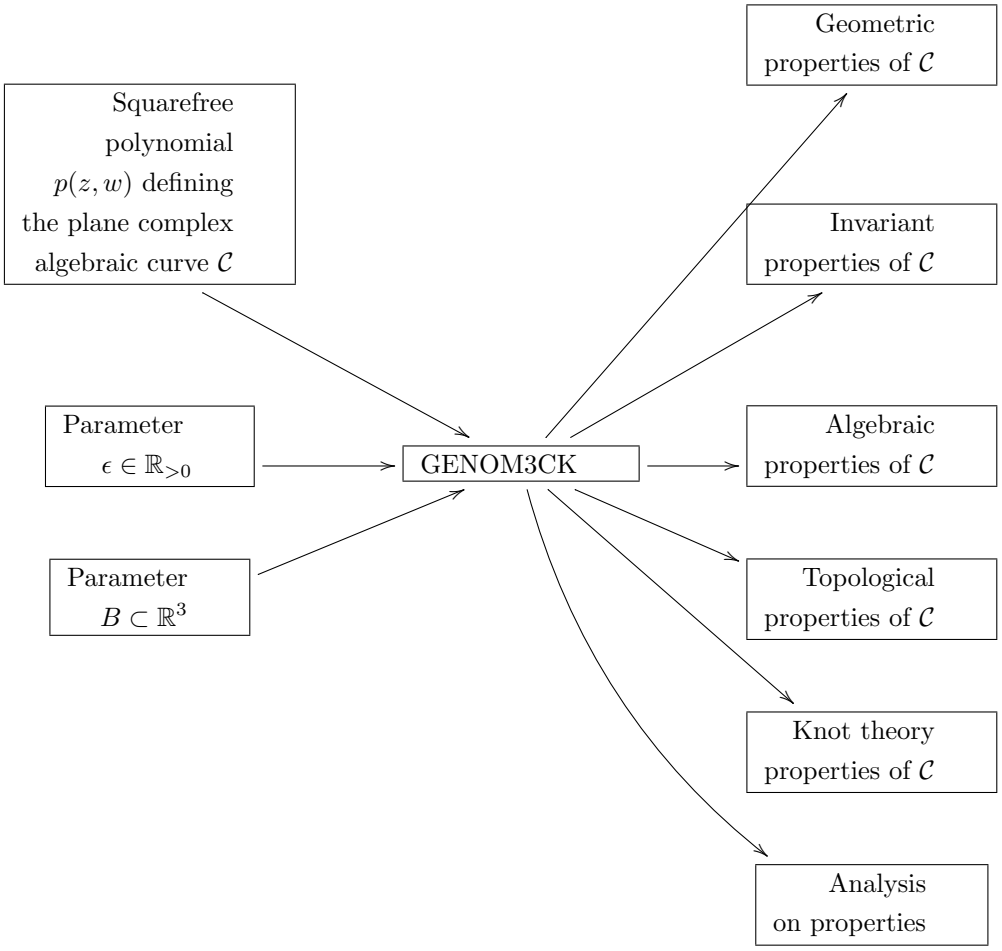


Figure 5.1: Input-output specification of the GENOM3CK library. For more details on each class of properties attached to a plane complex algebraic curve $\mathcal{C}$ and its singularities, see Figure 5.2.

Geometric properties	Invariant properties	Algebraic properties
the approximate link of each numerical singularity of the plane complex algebraic curve $\mathcal{C}$	the approximate Alexander polynomial of each approximate link of each numerical singularity of $\mathcal{C}$	the set of numerical singularities of $\mathcal{C}$ in the projective plane
the approximate Milnor fibration of each numerical singularity of $\mathcal{C}$	the approximate delta-invariant of each numerical singularity of $\mathcal{C}$	for each computed numerical singularity S of $\mathcal{C}$, the defining polynomial of $\mathcal{C}$ moved in the origin
the intersection points of each projected approximate link of each numerical singularity of $\mathcal{C}$	the approximate genus of $\mathcal{C}$	
	decide whether $\mathcal{C}$ admits an approximate rational parametrization	

Topological properties	Knot theory properties	Analysis of operations
the approximate Milnor number of each numerical singularity of the plane complex algebraic curve $\mathcal{C}$	the genus of each approximate link of each numerical singularity of $\mathcal{C}$	the computing time in seconds for the algebraic properties of $\mathcal{C}$
the approximate number of branches through each numerical singularity of $\mathcal{C}$	the unknotting number of each approximate link of each numerical singularity of $\mathcal{C}$	the computing time in seconds for the graph that represents the approximate link of each numerical singularity of $\mathcal{C}$
	the determinant of each approximate link of each numerical singularity of $\mathcal{C}$	the computing time in seconds for the geometric and the invariant properties of $\mathcal{C}$
	decide whether each approximate link of each numerical singularity of $\mathcal{C}$ is colorable	the computing time in seconds for the topological and the knot theory properties attached to $\mathcal{C}$
	the number of knot components for each approximate link of each numerical singularity of $\mathcal{C}$	
	a sequence representing the linking numbers for the knot components of each approximate link of each numerical singularity of $\mathcal{C}$	

Figure 5.2: Functionality of the GENOM3CK library. Visualization of the output produced by the GENOM3CK library, output divided into geometric properties, invariant properties, algebraic properties, topological properties, knot theory properties of a plane complex algebraic curve $\mathcal{C}$ and its singularities. The output of GENOM3CK contains also the computational time required for performing each type of operation in the category called "Analysis of operations".

2006 at INRIA, Sophia−Antipolis in the Galaad (Geometry, Algebra and Algorithms) team. Axel is written in the C++ programming language and it uses the Qt cross-platform application and UI framework [Thelin, 2007], [Blanchette and Summerfield, 2008] and OpenGL (Open Graphics Library) [Kuehne and Sullivan, 2008], [Guha, 2011]. Mathemagix is written in the C++ programming language, it can be embedded into other applications and it can be extended with libraries written in the C or in the C++ programming language. For more detailed information on Axel, Mathemagix and the main programming languages and libraries they are built on see Section 5.2.

For our purpose Axel provides an easy-to-use interface and unique algebraic and geometric tools for performing the following important operations: (i) the visualization of implicit algebraic curves and of implicit algebraic surfaces in the three-dimensional space. (ii) the computation of the topology of space implicit algebraic curves, topology that is computed as a 3-dimensional graph data structure. We recall that a 3-dimensional graph data structures is a set of vertices together with their Euclidean coordinates in $\mathbb{R}^3$, and a set of edges connecting them. Moreover, the existence of plugins in Axel allows us to reuse and to combine all of its computational power with the proposed symbolic-numeric algorithms from Chapter 3 into one library. Furthermore, we use algebraic techniques from Mathemagix, such as, for instance, subdivision techniques for the computation of the set of numerical singularities of a plane algebraic curve. By using the algebraic geometric modeler Axel and the free computer algebra system Mathemagix, we basically integrate symbolic, numeric and graphical capabilities into a single library, which we call GENOM3CK [Hodorog et al., 2010a].

We developed an online platform support for the GENOM3CK library including download, installation instructions and an extended documentation containing several examples that indicate the usage of the library. The online platform support of the library can be checked at the webpage: http://people.ricam.oeaw.ac.at/m.hodorog/software.html. For the creation of the webpage of the GENOM3CK library we employed the free editing platform called GNU TeXmacs [van der Hoeven, 2004], which provides specific features for scientists for producing technical and scientific documentation. Similarly to the LaTeX high-quality typesetting system, GNU TeXmacs provides a user friendly interface for editing organized documents containing text, mathematical formulas, pictures, interactive content, etc. In our study, one of the main advantages of choosing GNU TeXmacs over LaTeX for writing the documentation pages of the GENOM3CK library is the fact that TeXmacs admits a HTML converter, which allows the conversion of the TeXmacs files into HTML files. We add that the TeXmacs to HTML converter can be selected using the option `Edit`→`Preferences`→ `Converters` →`TeXmacs` →`Html`. In this case, the user can choose to export the mathematical formulas from the TeXmacs file to the HTML file as text, as images or as MathML. We mention that MathML (Mathematical Markup Language) is an application of XML (Extensible Markup Language) for describing mathematical notations.

5.1.3 Interface of the Library

The interface of GENOM3CK is part of Axel's interface, generated with Qt cross-platform application and UI framework. All the computational operations performed with the library GENOM3CK are incorporated into a main menu, which is called `Complex Invariant`, and they are divided into geometric properties, invariant properties, algebraic properties, topological properties, knot theory properties and analysis on properties as shown in Figure 5.3. The proposed symbolic-numeric algorithms prove to be efficient, as both its theoretical and its practical complexity analysis shows it.

The main menu of the GENOM3CK library is divided into the following submenus:

1. The submenu containing the geometric properties attached to a plane complex alge-

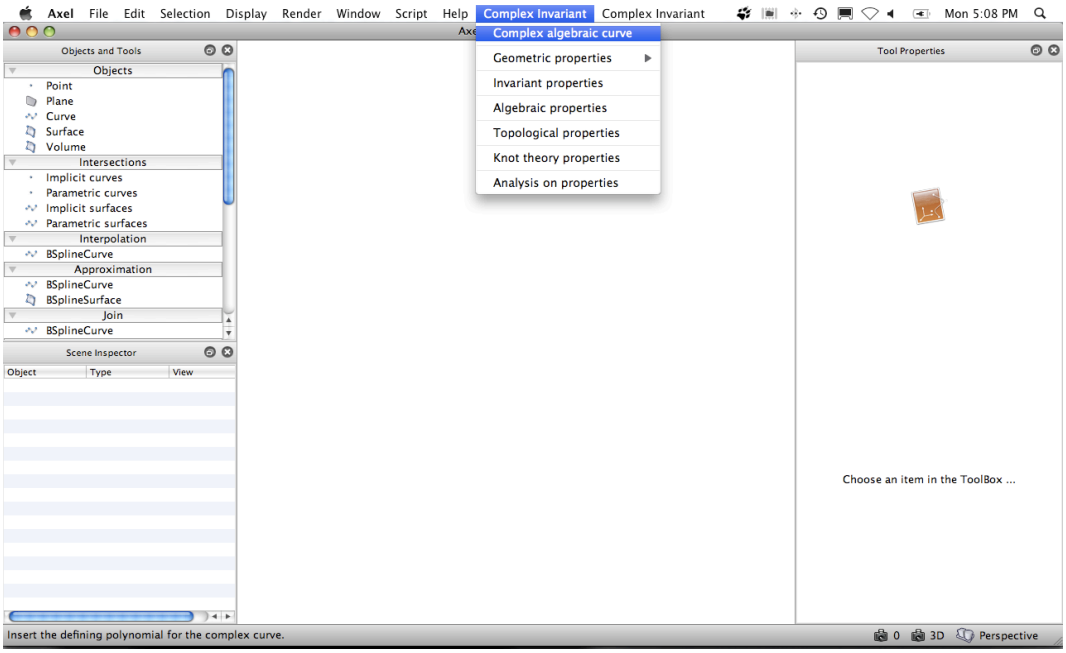


Figure 5.3: Main interface of the GENOM3CK library in Axel. The main menu of GENOM3CK is called "Complex Invariant".

braic curve and its singularities called `Geometric properties`, see Figure 5.4.

2. The submenu containing the invariant properties attached to a plane complex algebraic curve and its singularities called `Invariant properties`, see Figure 5.5.

3. The submenu containing the algebraic properties attached to a plane complex algebraic curve and its singularities called `Algebraic properties`, see Figure 5.6.

4. The submenu containing the topological properties attached to a plane complex algebraic curve and its singularities called `Topological properties`, see Figure 5.7.

5. The submenu containing the knot theory properties attached to a plane complex algebraic curve and its singularities called `Knot theory properties`, see Figure 5.8.

6. The submenu containing the analysis of each type of properties attached to a plane complex algebraic curve and its singularities called `Analysis on properties`, see Figure 5.9.

5.2 Implementation of the Library

5.2.1 Design of the Library

The library GENOM3CK is written in the Axel [Wintz et al., 2006] free algebraic geometric modeler and in the Mathemagix [van der Hoeven et al., 2002] free computer algebra system. We recall that Axel is written in the C++ programming language using Qt cross-platform application and UI framework [Thelin, 2007] and in OpenGL (Open Graphics Library) [Kuehne and Sullivan, 2008]. In addition, we remember that the Mathemagix

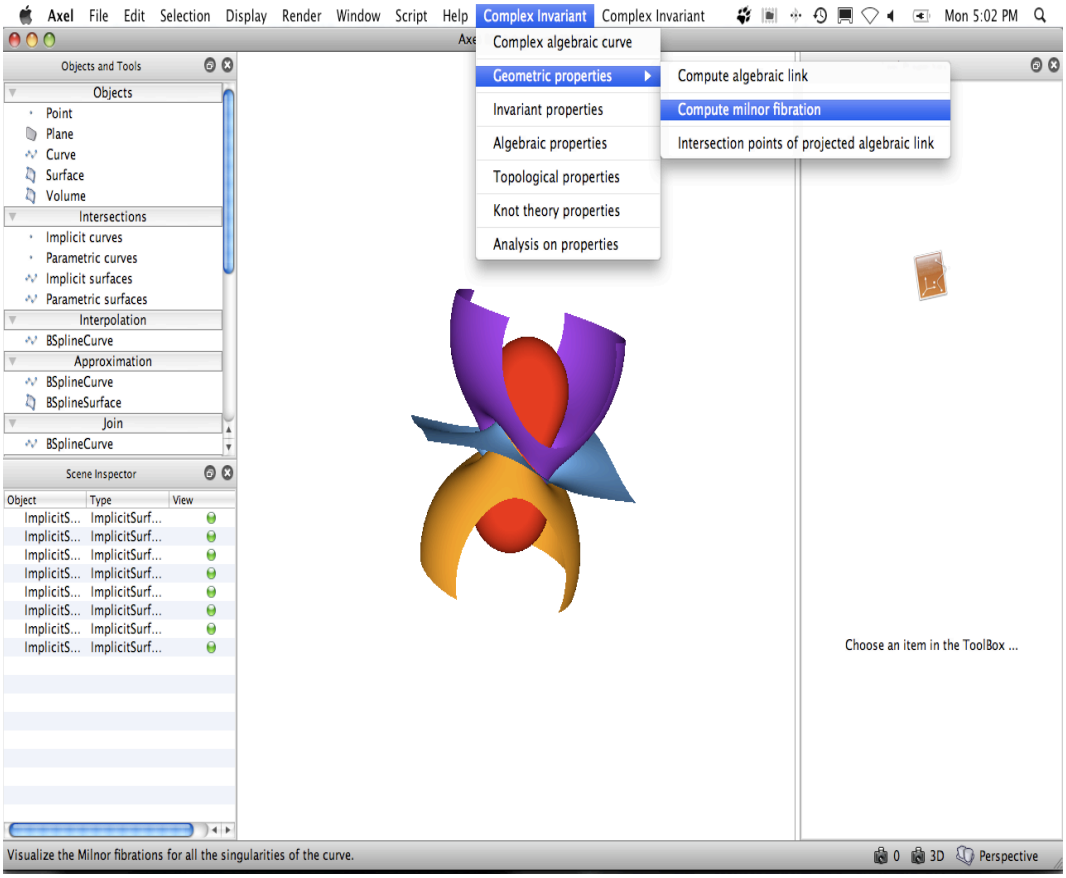


Figure 5.4: Interface of the GENOM3CK library in Axel showing the geometric properties of a plane complex algebraic curve and its singularities. The geometric properties are exemplified on the input plane complex algebraic curve defined by the squarefree bivariate complex polynomial $p(z, w) = z^2 - w^4 \in \mathbb{C}^2$.

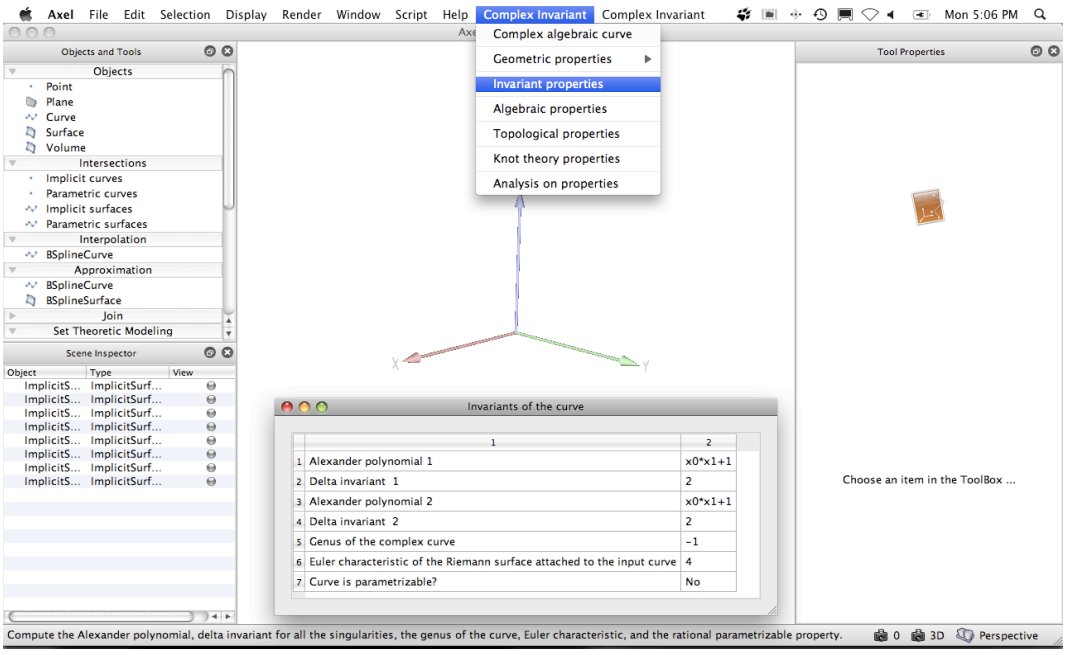


Figure 5.5: Interface of the GENOM3CK library in Axel indicating the invariant properties of a plane complex algebraic curve and its singularities. The invariant properties are exemplified on the input plane complex algebraic curve defined by the squarefree bivariate complex polynomial $p(z, w) = z^2 - w^4 \in \mathbb{C}^2$.

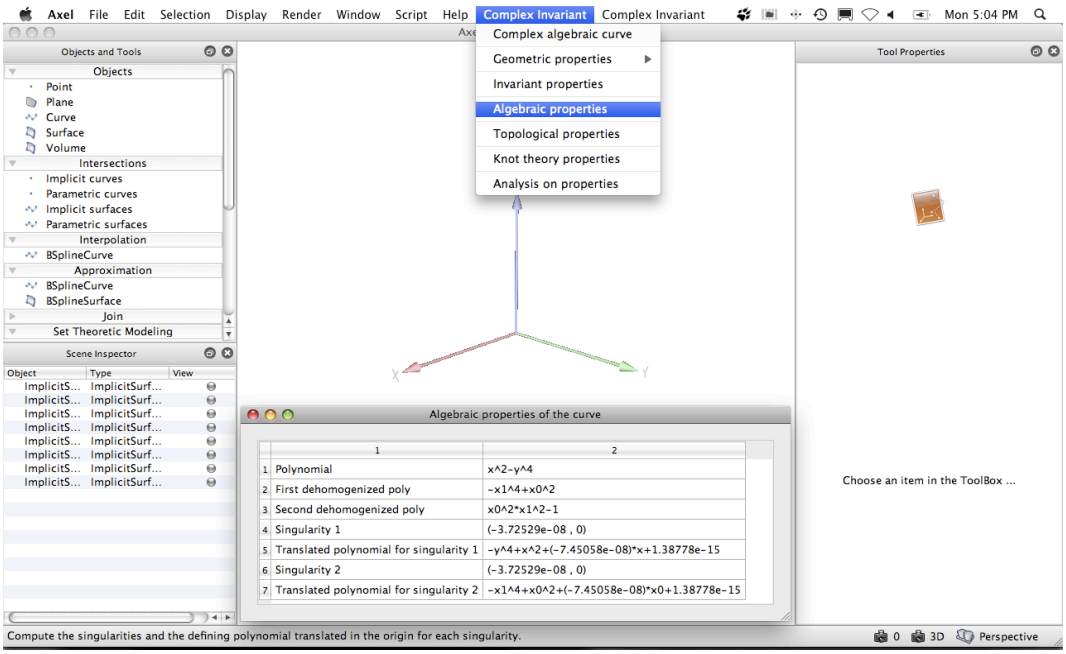


Figure 5.6: Interface of the GENOM3CK library in Axel presenting the algebraic properties of a plane complex algebraic curve and its singularities. The algebraic properties are exemplified on the input plane complex algebraic curve defined by the squarefree bivariate complex polynomial $p(z, w) = z^2 - w^4 \in \mathbb{C}^2$.

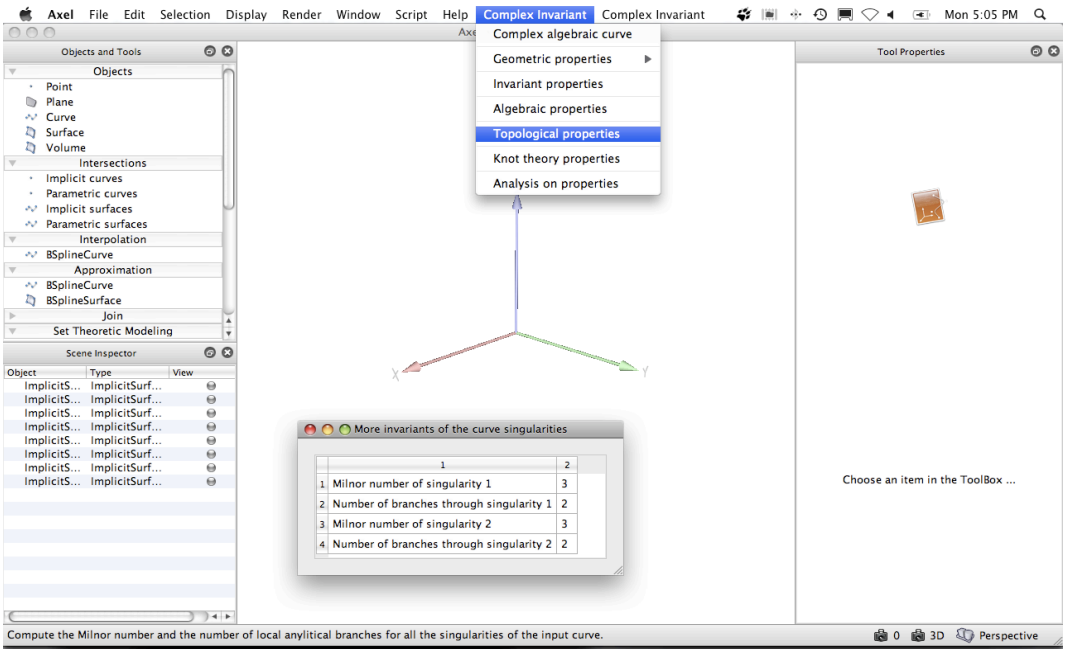


Figure 5.7: Interface of the GENOM3CK library in Axel rendering the topological properties of a plane complex algebraic curve and its singularities. The topological properties are exemplified on the input plane complex algebraic curve defined by the squarefree bivariate complex polynomial $p(z, w) = z^2 - w^4 \in \mathbb{C}^2$.

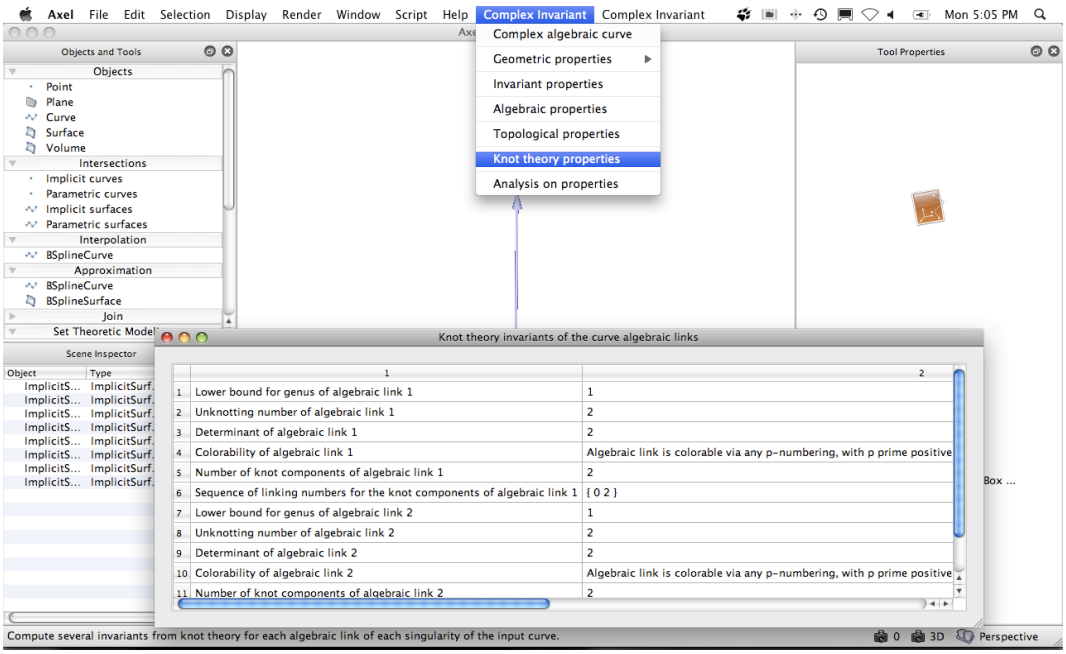


Figure 5.8: Interface of the GENOM3CK library in Axel depicting the knot theory properties attached to a plane complex algebraic curve and its singularities. The knot theory properties are exemplified on the input plane complex algebraic curve defined by the squarefree bivariate complex polynomial $p(z, w) = z^2 - w^4 \in \mathbb{C}^2$.

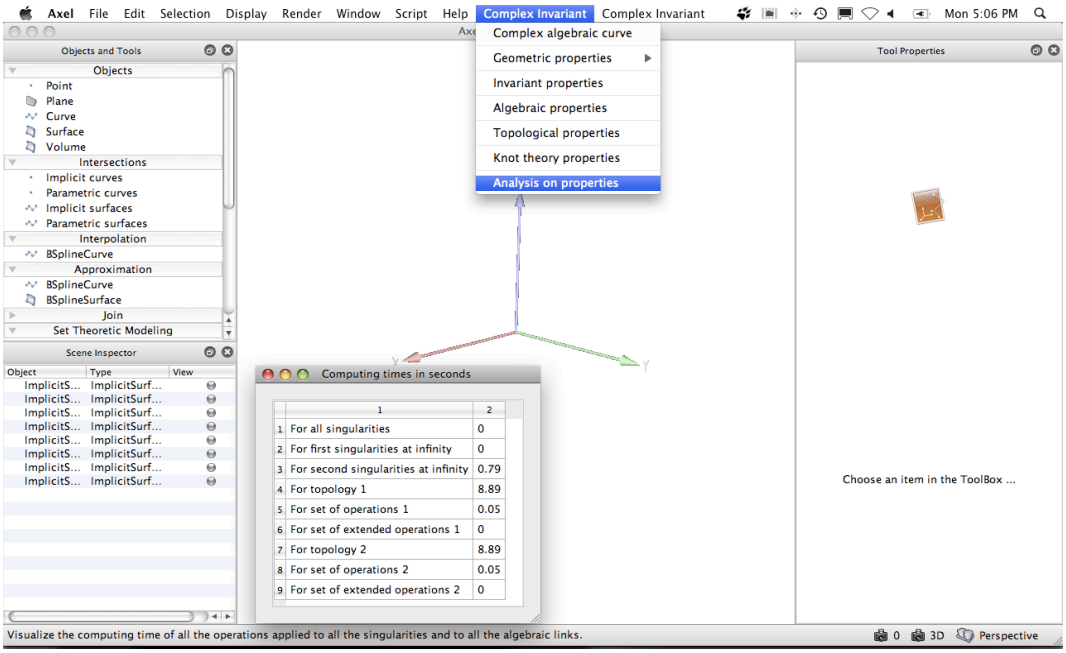


Figure 5.9: Interface of the GENOM3CK library in Axel reporting the analysis of properties attached to a plane complex algebraic curve and its singularities. The analysis of properties is exemplified on the input plane complex algebraic curve defined by the squarefree bivariate complex polynomial $p(z, w) = z^2 - w^4 \in \mathbb{C}^2$.

computer algebra system is written in the C++ programming language. In Figure 5.10 we present the main functionality of the Axel system and of the Mathemagix system. The library GENOM3CK is built on top of the two systems Axel and Mathemagix. We mention that as the Axel algebraic geometric modeler and as the Mathemagix computer algebra system, the library GENOM3CK is released under the GNU General Public License and it is stored as a project in the version control system SVN [Collins-Sussman et al., 2004]. We mention that a version control system allows the developer/the programmer to manage changes in documents, in source code files and in webpages, changes that are stored as computer files. SVN [Collins-Sussman et al., 2004] (or Apache Subversion) is one of the available version control systems, which is a free and an open source system.

In the following paragraphs we include some general remarks concerning the C++ programming language, the Qt cross-platform application and UI framework and the Open Graphics Library, systems that represent the main engines for the Axel system and for the Mathemagix system. By stating these remarks in our study, we do not intend to give a complete survey on the aforementioned engines, but we do wish to familiarize the reader with the main reasons for which these engines are used for building systems as Axel and Mathemagix for algebraic computation and geometric modeling in the scientific society. We assume that the reader is acquainted with basic notions concerning structured programming languages as C and with basic notions concerning object-oriented programming languages as C++ . Assuming the reader has no knowledge of structured or of object-oriented programming languages, the following remarks will still give a rough idea about the main paradigms and principles used in the two types of programming languages. For more details on the C and on the C++ programming languages, the reader is advised to consult [Banahan et al., 1991], and respectively [Allison and Eckel, 2004].

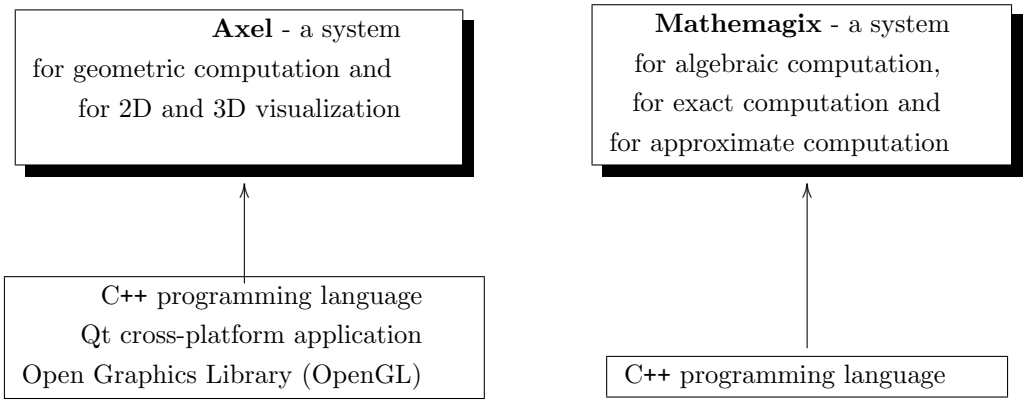


Figure 5.10: Main functionality of the free Axel algebraic geometric modeler and of the free Mathemagix computer algebra system. Axel and Mathemagix are the two systems used for developing the GENOM3CK library.

C++ Programming Language. In our overview concerning the C++ programming language we follow the books of [Eckel, 2000] and of [Allison and Eckel, 2004]. The C++ programming language is an extension of the C programming language, which in addition offers object-oriented programming capabilities. The C++ programming language was basically introduced as a better C programming language, feature that is implied also by its name. The name C++ contains the "++" construction, which is the C syntax employed for incrementing the value of a variable. We distinguish the following main features of the C++ programming language:

- **the object-oriented paradigm**: the object-oriented paradigm offers tools for creating general representations of the elements in the state space of the problem that we want to solve. The elements in the state space of a problem and their general representation are called *objects*. All objects that have the same characteristics (data elements) and behaviours (functionality) are grouped into a *class*. The existence of classes allows the programmer to extend the existing programming language by adding new data types. These new created data types fit the problem that needs to be solved. In this way, the programmer is not constrained to using the existing data types, but he can create his own data-types, depending on his needs. By using and defining classes, we can thus introduce abstract data types, which are basically data types dependent on the operations they support and independent on their structure and on their implementation. The existence of abstract data types in C++ is sometimes referred to as *data abstraction*. We add that the existence of classes is not the only main feature of the object-oriented paradigm. Other essential characteristics of the object-oriented paradigm are for instance *inheritance* and *polymorphism*. We present basic remarks concerning these notions later in this subsection. For more details on the object-oriented paradigm and its main features check [Eckel, 2000].

- **the portability**: the programs written in C++ are independent on the type of computer and on the type of operating systems used by the programmer.

- **the modular programming**: the programs written in C++ are divided into separate files. The modular programming in C++ employs separate interface files and separate implementation files. The interface files have the extension `.h` and they contain declarations of variables and of functions, but they do not contain any source code. In

contrast, the implementation files have the extension `.cpp` and they do contain source code. The implementation files also contain all the declarations from the interface files by using the #`include` directive. Thus a module in the C++ programming language is a collection of functions or classes that perform the same operations. We mention that by using modular programming, a program written in C++ can contain more source files. These different source files are separately compiled and then they are linked together into one executable program. It follows that, whenever a source file is modified, the programmer has to recompile only the modified source file and not all the source files of the program. This feature makes it possible to link C++ code with code produced in other programming languages, such as C.

- **the compatibility with the C programming language**: any program written in the C programming language can be included in the C++ programming language without performing any changes to the original source code.
- **the efficiency**: the efficiency of a C++ program is derived from knowing the type of object at the compilation time. This information is provided in the C++ programming language by the *generic programming* approach, which is obtained by *templates*. We will discuss the notion of templates in C++ later in this subsection.

We report on the following characteristics of the object-oriented C++ programming, characteristics that represent essential and unique tools for performing effective algebraic computations in an efficient computer algebra system:

- **the existence of classes:** we recall that a class is a collection grouping together both data structures and functions/methods in a new type. A class always contains a special method called a constructor, which has the same name as the class itself. The existence of constructors is essential for a class, i.e. every time an instance of a class is created the constructor method is called.
- **the existence of types:** we distinguish between two main types in the C++ programming language: the default types (i.e int, long, float, double, etc) and the user-defined types (i.e. classes, structs, unions).
- **the existence of namespaces:** in the C++ programming language, the namespaces allow the developer to group together under the same name C++ entities such as variables, constants, classes, objects, functions/methods, etc. We add that the default namespace `std` of the C++ programming language contains the entire standard library of the C++ programming language.
- **the existence of inheritance:** the inheritance in the C++ programming language is a mechanism that allows to reuse and to extend a given class without making modifications to the class itself. The inheritance mechanism allows the programmer thus to produce hierarchical relations between classes. The new created class is called the *derived class*, whereas the original class is called the *base class*. We mention that the derived class contains all the data structures and the functions of the base class, except the constructors of the base class.
- **the existence of templates**: a template allows the programmer to define a generic function or a generic class. It follows that a generic function, and respectively a generic class generates a family of functions, and respectively a family of classes. The "Standard Template Library" represents the template library that is included in the C++ programming language. This library contains different container templates and data structures, which allow the programmer to produce robust and complex data

structures in a relatively easy manner. We add that the programming style done by using templates is called *generic programming*. The advantage of generic programming is that the programmer can model the functionality of a class independently on the type of data existent in the class.

- **the existence of polymorphism and of virtual functions:** we add that polymorphism, which is implemented in C++ using virtual functions, is the third essential characteristic of the object-oriented paradigm, after data abstraction and inheritance. Informally, the polymorphism gives different meanings and functionalities to the methods of a class. Formally, the polymorphism is implemented in C++ in the following way: the base class declares one of its method/function as a virtual function using the keyword `virtual`; the derived class then overrides the definition of the virtual function. We add an important remark concerning the binding process for virtual functions in the C++ programming language. We mention that *binding* is the process that connects a function call to a function body. If binding is realized by the compiler and the linker before the program is run, then it is called early binding. It is important to state that a virtual function in C++ causes the compiler to perform *late binding*, as opposed to early binding. Late binding, called also runtime binding or dynamic binding, is a mechanism of dynamic binding using function pointers, which is performed at runtime. For more information on late binding, please consult [Eckel, 2000].

Qt cross-platform application and UI framework. In our outline concerning the Qt framework we follow the books of [Thelin, 2007] and of [Blanchette and Summerfield, 2008]. As stated by its name, Qt is a cross-platform application and UI framework. The Qt framework is mainly used for developing software applications with a graphical user interface (GUI). Still, the Qt framework can also be used for designing software applications without a graphical user interface. We mention that the Qt framework offers mainly application programming interfaces (APIs) for the C++ programming language (or for the JavaScript language), however it is possible to use Qt in another programming languages as well by using language bindings. The Qt framework allows the developer to include also 3-dimensional graphics objects, since the framework itself includes a separate module that incorporates OpenGL (see the next paragraph for more information on OpenGL and its main functionalities). We add that OpenGL is suitable for visualization in the 3-dimensional space. However OpenGL provides little functionality for creating application user interfaces, which are available in the Qt framework. Lately, Qt has known an increased popularity both in industry and in the scientific community.

Open Graphics Library (OpenGL). In our survey concerning the Open Graphics Library we use the books of [Kuehne and Sullivan, 2008] and of [Guha, 2011]. The Open Graphics Library (OpenGL) is a library of functions, a library that allows the programmer to produce 2-dimensional and 3-dimensional images in computer graphics. In addition, OpenGL offers a user friendly interface as the library hides all the functionalities from the user. OpenGL is basically a low-level graphics library. This means that the library demands the user or the programmer to specify exactly the objects that are to be visualized. As presented in [Semwal, 2002], we mention the following main characteristics of OpenGL:

- OpenGL comprises the following collection of basic geometric primitives: points, lines, polygons, images and bitmaps.
- OpenGL basically contains a collection of commands that allow the programmer to define geometric objects in the 2-dimensional and in the 3-dimensional space. These commands produce the geometric objects by using the aforementioned basic geometric

primitives from OpenGL and a collection of procedures that control the way in which the geometric objects are drawn.

- Since OpenGL offers a rather restrictive set of geometric objects represented by basic geometric primitives (i.e. points, lines and polygons), we mention that the OpenGL Utility Toolkit (GLUT) was produced to allow the programmer to design more complicated geometric objects in the 3-dimensional space, such as for instance spheres or tori.

- OpenGL offers an effective but primitive collection of commands for drawing geometric objects. All the high level-drawing in OpenGL has to be done depending on these primitive collection of commands. The OpenGL Utility Library (GLU) and the OpenGL Utility Toolkit (GLUT) are two libraries that help the designer/the programmer to develop the main programming tasks in OpenGL. GLU offers different procedures that employ low-level commands from OpenGL to produce basic operations in OpenGL such as for instance introducing matrices for defining viewing orientations and for defining drawing of surfaces. GLUT implements an easy windowing application programming interface (API) for OpenGL.

As described in [Network, 2002] and in [Semwal, 2002], in the following paragraph we shortly present the main graphics operations that OpenGL performs for drawing an arbitrary image on the screen/display:

1. the first graphics operation is the *construction of shapes* for the considered scene from the basic geometric primitives (i.e. points, lines, polygons, images and bitmaps). At this step a mathematical description of the object is created;

2. the second graphics operation is the *arrangement of the objects in the* 3*-dimensional space* and the selection of the desired viewpoint for the considered scene;

3. the third graphics operation is the *computation of the color of all the objects* that compose the considered scene. We can assign color to objects in OpenGL by performing one of the three following actions: (i) assigning the color explicitly; (ii) determining the color from the specified lighting conditions; (iii) pasting a texture onto the objects. We mention that for computing the color of objects in OpenGL we can also perform a combination of the three aforementioned actions (i), (ii), (iii).

4. the forth and the last graphics operation is the conversion of the mathematical description of the object and of its color information to pixels on the screen. This process is called *rasterization*.

What are the advantages of combining C++ programming language, Qt framework and OpenGL to create computer algebra systems? It follows that by using the C++ object-oriented programming language combined with the Qt cross-platform application and UI framework and the Open Graphics Library, robust computer algebra systems as Mathemagix and effective algebraic geometric modeler as Axel can be successfully produced and employed in the scientific community. First of all, the generic programming offered by the C++ object-oriented programming language allows the developer to create generic algorithms, which can be applied to a large class of data structures. The object-oriented paradigm basically allows the programmer to combine generic implementations with specialized functions. Secondly, the Qt cross-platform application and UI framework allows the developer to create modern user interfaces. Finally, the Open Graphics Library provides tools and procedures for producing 3-dimensional images containing 3-dimensional

geometric objects. Moreover, by using the Qt framework we can extend the systems by adding specific functionalities and we can create plugins such as GENOM3CK. We mention that GENOM3CK is basically one of Axel's plugins. In the next paragraph, we indicate the main packages that GENOM3CK uses from Axel and Mathemagix.

As rendered in Figure 5.11, the GENOM3CK library uses the following specific packages from Axel and Mathemagix:

1. from the Axel system, the library GENOM3CK uses algebraic packages from Mathemagix, packages that are employed for geometric modeling. We mention that the geometric objects displayed with the library GENOM3CK are geometric objects from Axel, see [Wintz et al., 2006] for more details.

2. from the Mathemagix system, the library GENOM3CK uses algebraic computation packages, exact computation packages and approximate computation packages. We add that for implementing the symbolic-numeric algorithms designed in Chapter 3, the approximate computation packages from Mathemagix offer the following efficient and unique tools, which are essential for the purpose of this thesis:

   (a) The subdivision methods for computing the set of numerical singularities of a plane complex algebraic curve.

   (b) The same subdivision methods from Mathemagix are employed by Axel to compute the topology of each approximate link of each numerical singularity of a plane complex algebraic curve. We recall that the topology of an approximate link of a plane curve singularity is computed with subdivision methods from Mathemagix in Axel as a 3-dimensional graph data structure. We remember that a 3-dimensional graph data structure is a set of points (also called vertices) in the 3-dimensional Euclidean space together with their Euclidean coordinates, and a set of edges connecting them.

   The algorithms implemented in GENOM3CK use the following set of algebraic and of exact computation tools from Mathemagix:

   (a) data structures for representing vectors, matrices, univariate polynomials and multivariate polynomials;

   (b) functions for manipulating univariate and multivariate polynomials;

   (c) functions for manipulating homogeneous polynomials;

   (d) functions for manipulating matrices of integers and matrices of univariate and of multivariate polynomials;

   (e) algebraic algorithms for computing the exact greatest common divisors of univariate and multivariate polynomials;

   (f) algebraic algorithms for computing the exact determinant of a matrix of univariate polynomials;

   (g) algebraic algorithms for computing the exact minors of a matrix of multivariate polynomials, etc.

5.2.2 Dependencies of the Library

At present, the library GENOM3CK is available for the Macintosh OS X operating system and for the Linux operating system. We mention that the library GENOM3CK is an ongoing project, and therefore it suffers continuous updates and modifications. For

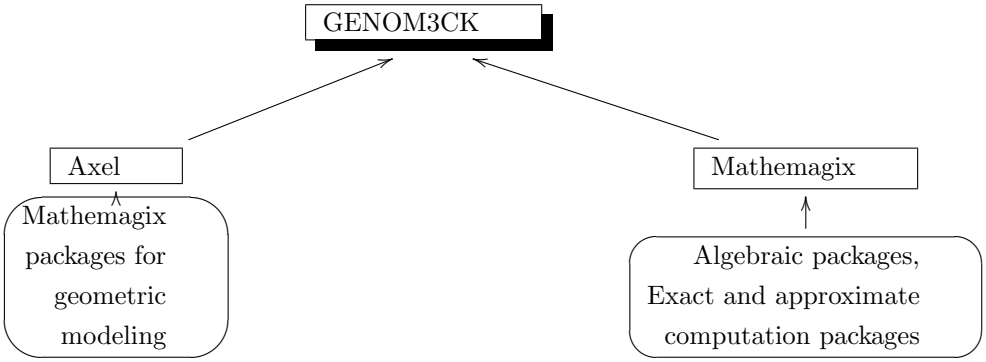


Figure 5.11: Design of the GENOM3CK library. GENOM3CK is built on top of the two free systems Axel and Mathemagix, released under the GNU General Public License.

downloading the latest version of the library, please consult the homepage of the library. In addition, we add that the development of the library GENOM3CK depends on the development of the two systems Axel and Mathemagix. Axel and Mathemagix are ongoing projects, and thus they also suffer constant modifications and improvements. For download and installation instructions concerning the Axel and the Mathemagix systems, please consult the official websites of the two systems, i.e. [Wintz et al., 2006] and respectively [van der Hoeven et al., 2002].

In this paragraph, we assume that for a machine running the Macintosh OS X operating system the Xcode tools are successfully installed and available. For installing the library GENOM3CK on a machine running the Macintosh OS X operating system or the Linux operating system, we need to install the following systems and tools:

- Version 4 of the Qt framework. For more details on the Qt framework, the reader can consult [Corporation, 2008], [Thelin, 2007] and [Blanchette and Summerfield, 2008]. Please notice that for installing the Axel algebraic geometric modeler, the user/the developer is required to install at least version 4 of the Qt framework.
- The GMP library. GMP (GNU Multiple-Precision) [Foundation, 2000] is a free library for arbitrary precision arithmetic on integer numbers, rational numbers and floating point numbers. We mention that the library GMP is written in the C programming language.
- The MPFR library. GNU MPFR [Fousse et al., 2007], [Hanrot et al., 2005] is a free library for multi-precision floating point computations with correct rounding. We add that MPFR is based on the GMP library.
- The CMake system. CMake [Martin and Hoffman, 2003] is an open source build system, which contains tools for building, testing and packaging software. We mention that for installing the Axel algebraic geometric modeler, the user/the developer is required to install at least the version 2.6 of the CMake system.
- The Mathemagix [van der Hoeven et al., 2002] computer algebra system.
- The Axel [Wintz et al., 2006] algebraic geometric modeler.

We mention that for installing these tools on a machine running the Macintosh OS X operating system, the user/the developer can either user the MacPorts [Project, 2002] package management system or the Fink [Project, 2001] package management system. We add that

the Axel algebraic geometric modeler is based on certain libraries (i.e. shape, realroot, newmac, linalg, etc) of the Mathemagix computer algebra system. In the later versions of the Axel system, it should not be necessary to install the whole Mathemagix computer algebra system for building Axel, but only the necessary libraries from Mathemagix on which Axel is built on. Still, if troubleshooting appears during the installation process, the user/the developer is strongly advised to install the whole Mathemagix computer algebra system before installing Axel. As the GENOM3CK library is one of Axel's plugin, the library is part of the Axel system and thus it is installed after the successful installation of the Axel system. For more information on the individual libraries of the Mathemagix computer algebra system, the user/the developer is strongly advised to consult the official webpage of the Mathemagix computer algebra system at [van der Hoeven et al., 2002]. We make some remarks concerning the most important libraries of the Mathemagix computer algebra system, which are used by the Axel algebraic geometric modeler:

- `shape` library. The shape library contains functions and operations on algebraic curves and on algebraic surfaces, defined using their implicit representation by polynomial equation, or defined using their rational parametrization. This library contains methods for computing the topology, the singularities and the intersection of algebraic curves and of algebraic surfaces.

- `realroot` library. The realroot library contains functions and operations on univariate and multivariate polynomials. The library contains methods (i.e. subdivision methods [Mourrain and Pavone, 2009], [Liang et al., 2008]) for computing the real roots of univariate and multivariate polynomial equations and of systems of polynomial equations.

- `newmac` library. The newmac library contains methods for computing the solution set of a system of polynomial equations, using the Gröbner basis method and the Macaulay construction. For more information on this method, the reader is advised to consult [Mourrain and Trébuchet, 2005].

- `linalg` library. The linalg library contains numerical methods from linear algebra. It basically represents the template C++ version of the LAPACK library, for information see [Anderson et al., 1999]. We add that LAPACK is a library containing functions for solving systems of linear equations, methods for solving the eigenvalue problems, etc.

5.3 Usage of the Library

We mention that this section mainly follows the documentation of the GENOM3CK library found on the official homepage of the software.

5.3.1 Instructions for the User

In this section, we describe the main instructions that the user must follow in order to successfully run, employ and exploit the GENOM3CK library. We add that the user, who wants to purely test and to use the library GENOM3CK on different examples, is recommended to download and to install the compiled version of the library GENOM3CK on a machine running the Macintosh OS X operating system. The corresponding dmg file can be downloaded directly from the official homepage of the library. We recall that the input of the library GENOM3CK consists of the following data:

1. A squarefree bivariate complex polynomial $f(z,w) \in \mathbb{C}[z,w]$.
2. A positive real number $\epsilon \in \mathbb{R}_{>0}$ called the input parameter (or the regularization parameter).
3. A subset of the 3-dimensional Euclidean space $B = [-a,a] \times [-b,b] \times [-c,c] \subset \mathbb{R}^3$ called a box, which specify the x, y and z coordinates of the 3-dimensional Euclidean space.

The first thing the user needs to do when running the GENOM3CK library is to insert the input data represented by the polynomial $f(z,w)$, by the regularization parameter $\epsilon \in \mathbb{R}_{>0}$ and by the box $B \subset \mathbb{R}^3$. For introducing the input data, the user has to directly access the button called `Complex algebraic curve`, which causes the appearance of the following boxes:

- A box for introducing the polynomial $f(z,w) \in \mathbb{C}[z,w]$ with both exact and inexact coefficients. We add that the input polynomial has to be introduced in the variables x, y instead of the variables z, w. Moreover, we mention that the input polynomial is given with a certain tolerance (or noise) in its coefficients. For instance, the coefficient 1.083 represents the floating-point number 1.083, which in this case is associated with the noise $\delta = 10^-3$, which basically means that the last digit of 1.083 is unknown. In addition, we add that the power function has to be introduced in the form $x\hat{}n$, for any $n \in \mathbb{N} \setminus \{0\}$, whereas the multiplication function has to be typed in the form $x * y$. Furthermore, whenever the user wants to introduce an equation of the form $(x-1)^2 - (y-2)^3$, the equation has to be introduced in its expanded form, i.e. $9 - 2 * x + x\hat{}2 - 12 * y + 6 * y\hat{}2 - y\hat{}3$.
- Two boxes (one for the numerator $n \in \mathbb{N}$ and another one for the denominator $d \in \mathbb{N} \setminus \{0\}$) for introducing the regularization parameter $\epsilon \in \mathbb{R}_{>0}$ in the form $\epsilon = \dfrac{n}{d}$.
- Three boxes for introducing the subset $B = [-a,a] \times [-b,b] \times [-c,c] \subset \mathbb{R}^3$ for the x, y and z coordinates of the 3-dimensional Euclidean space $\mathbb{R}^3$ with $x \in [-a,a], y \in [-b,b]$ and $z \in [-c,c]$ and $a, b, c \in \mathbb{N} \setminus \{0\}$.

After introducing the input data, the library calls the methods from Chapter 3 to compute the set of all operations on the plane complex algebraic curve defined by the polynomial $f(z,w) \in \mathbb{C}[z,w]$. In the case that the methods from the library successfully compute the desired output, the library will display a successful message. In this case, the user can choose the visualization of one of the main operations performed on the input plane complex algebraic curve, as summarized in Figure 5.2. We notice that the methods implemented in the GENOM3CK library can fail, in which case the library will produce a failure message. The failure message contains a short explanation for the failure behaviour and if possible it presents the user the future options needed for running the library again. The main reasons for the failure of the library are the following:

- the ϵ-link L_ϵ of the singularity P of the input plane complex algebraic curve $\mathcal{C}$ has singularities. We recall that L_ϵ is computed as the projection of the intersection of the curve $\mathcal{C}$ with the sphere $S_\epsilon(P)$ of radius ϵ (where ϵ is the regularization parameter) and centered in the singularity P. In this case, the user can run the library again with another value for the regularization parameter ϵ, or it can be that the intersection of the curve $\mathcal{C}$ with the sphere $S_\epsilon(P)$ has singularities.
- the box B representing the domain for the x, y and z coordinates of $\mathbb{R}^3$, which is required for computing the ϵ-link L_ϵ is not big enough to contain the entire topology

of the ϵ-link L_ϵ. In this case, the topology of L_ϵ, represented by a graph data structure, is not contained in B and therefore the value for the box B has to be increased.

5.3.2 Instructions for the Developer

In this subsection, we include the main instructions that the developer must follow in order to build the library GENOM3CK, to test it and to extend it by adding new functionalities to it. We mention that for the developer, who wants to study more the source code of the GENOM3CK library, it is recommended to build the library GENOM3CK directly from source code. It follows that the developer has to successfully build the following tools on a machine running the Macintosh OS X or the Linux operating systems:

- Version 4 of the Qt framework, see [Corporation, 2008] for building instructions.
- The GMP library, see [Foundation, 2000].
- The MPFR library, see [Hanrot et al., 2005].
- The Mathemagix computer algebra system, see [van der Hoeven et al., 2002].
- The CMake system, see [Commons, 2000].
- The Axel algebraic geometric modeler, see [Wintz et al., 2006]. For building Axel, we must first checkout the svn repository containing the system from the INRIA server and then we can build the system. Once Axel is installed, we download the source code of the GENOM3CK library and we build it.

5.4 Test Experiments

We mention that this section mainly follows the documentation of the GENOM3CK library found on the official homepage of the software. In addition, this section contains results from [Hodorog and Schicho, 2011b].

5.4.1 Examples for the Computation of Approximate Invariants

In this subsection, we include several examples that indicate the computation of the approximate invariants attached to a plane complex algebraic curve. We recall that the computation of the approximate invariants attached to a plane complex algebraic curve is performed by the methods from Chapter 3 implemented in the library GENOM3CK.

The *first example* indicates the *computation of the algebraic properties* attached to the plane complex algebraic curve $\mathcal{C}$ defined by the squarefree bivariate complex polynomial $x^2 - y^3 \in \mathbb{C}[x, y]$. We recall that the algebraic properties are represented by the set of numerical singularities in the projective plane of the input plane complex algebraic curve $\mathcal{C}$. To visualize the set of singularities in the projective plane of the input curve $\mathcal{C}$, we proceed in the following way:

1. We consider the input polynomial $f(x, y) = x^2 - y^3$ of degree 3 and its associated homogenized polynomial $g(x, y, z) = x^2 z - y^3$.
2. We consider the polynomial $f_1(x, y) = g(x, y, 1) = x^2 - y^3$ denoted in GENOM3CK with `Polynomial` and we compute the set of singularities S_1 of the affine curve defined by the polynomial $f_1(x, y)$. In this case we compute $S_1 = \{(-3.72529e - 08, 0)\}$.

3. We consider the polynomial $f_2(y,z) = g(1,y,z) = z - y^3$ denoted in GENOM3CK with `First dehomogenized polynomial` and we compute the set of singularities S_2 of the affine curve defined by the polynomial $f_2(y,z)$. In this case we compute $S_2 = \emptyset$.

4. We consider the polynomial $f_3(x,z) = g(x,1,z) = x^2z - 1$ denoted in GENOM3CK with `Second dehomogenized polynomial` and we compute the set of singularities S_3 of the affine curve defined by the polynomial $f_3(x,z)$. In this case we compute $S_3 = \emptyset$.

5. We obtain the set of singularities $S = S_1 \cup S_2 \cup S_3$.

6. The following output is computed and displayed by the library GENOM3CK as part of a table containing the `Algebraic properties` of the input plane complex algebraic curve $\mathcal{C}$ defined by the squarefree polynomial $x^2 - y^3$:

   - `Polynomial` $\rightarrow x^2 - y^3$. The polynomial $x^2 - y^3$ is displayed as $x_0^2 - x_1^3$ since the library GENOM3CK employs the same notation as in the Axel system, i.e. the index notation for the sequence of variables of a polynomial. For example, in the index notation, the sequence $\{x, y\}$ is equivalent to $\{x_0, x_1\}$.
   - `First dehomogenized polynomial` $\rightarrow z - y^3$. The polynomial $z - y^3$ is displayed as $x_0 - x_1^3$, because in the index notation, the sequence $\{z, y\}$ is equivalent to $\{x_0, x_1\}$.
   - `Second dehomogenized polynomial` $\rightarrow x^2z - 1$. The polynomial $x^2z - 1$ is displayed as $x_0^2x_1 - 1$, as in index notation, the sequence $\{x, z\}$ is equivalent to $\{x_0, x_1\}$.
   - `Singularity 1` $\rightarrow (-3.72529e - 08, 0)$.
   - `Translated polynomial for singularity 1` $\rightarrow -x_1^3 + x_0^2 + (-7.450583 - 08)x_0 + 1.3877e - 15$.

We mention that the library GENOM3CK uses the scientific notation for floating point numbers as in the C++ programming language. In the scientific notation, a number has two parts: the significand and the exponent, which is basically a power of 10. The letter e is used to separate the two parts. Thus $7e2$ is equivalent to $7 * 10^2$ (or 700), whereas $7e - 2$ is equivalent to $7 * 10^{-2}$ (or 0.07). In the example mentioned above the floating point number $-3.72529e - 08$ is equivalent to $-3.72529 * 10^{-8}$.

The *second example* shows the *computation of the geometric operations* attached to the input plane complex algebraic curve $\mathcal{C}$ defined by the squarefree bivariate complex polynomial $x^2 - y^3 \in \mathbb{C}[x, y]$. We recall the main geometric operations attached to a plane complex algebraic curve:

- the ϵ-link L_ϵ of each numerical singularity Q of the plane complex algebraic curve $\mathcal{C}$. We remember that the ϵ-link L_ϵ is a space algebraic curve that is computed as the intersection of two space algebraic surfaces defined by two polynomials $g_\epsilon(x, y, z)$ and $h_\epsilon(x, y, z)$.

- Another geometric property attached to a plane complex algebraic curve is the computation and the visualization of the algebraic surfaces with defining polynomials g_ϵ, h_ϵ, surfaces that are part of the Milnor fibration and that define as their intersection the ϵ-link L_ϵ.

- The final geometric property attached to a plane complex algebraic curve is represented by the intersection points of the projection of the ϵ-link of each numerical singularity of the input curve.

The following output is computed and displayed by the library GENOM3CK as part of a table containing the `Geometric properties` of the input plane complex algebraic curve $\mathcal{C}$ defined by the squarefree polynomial $x^2 - y^3$:

- `Compute algebraic link` produces the visualization of the ϵ-link L_ϵ of each numerical singularity of the input curve $\mathcal{C}$. We mention that L_ϵ is represented by a `Mesh` object in the Axel system. We add that the ϵ-link L_ϵ of the singularity $(-3.72529e-08, 0)$ of the input curve $\mathcal{C}$ defined by the squarefree polynomial $x^2 - y^3$ is represented by the trefoil knot, i.e. by the torus knot of type $(2, 3)$.

- `Compute Milnor fibration` produces the visualization of the algebraic surfaces defined by the polynomials $g_\epsilon, h_\epsilon, g_\epsilon + h_\epsilon, g_\epsilon - h_\epsilon$, which are all part of the Milnor fibration and which define as their intersection the ϵ-link L_ϵ. We add that these surfaces are represented as `ImplicitSurface` objects in the Axel system.

- `Intersection points of the projected algebraic link` produces the visualization of all the intersection points of the projection of the ϵ-link L_ϵ.

In Table 5.1 and in Table 5.2 we include several examples performed with the library GENOM3CK on several input plane complex algebraic curves defined by squarefree polynomials with exact and respectively with inexact coefficients. Basically, these examples indicate the geometric properties attached to the input plane complex algebraic curves. In both tables the regularization parameter $\epsilon \in \mathbb{R}_{>0}$ and the box $B \subset \mathbb{R}^3$ are set to the following values: $\epsilon = 0.25$ and $B = [-4, 4] \times [-6, 6] \times [-6, 6]$. We notice that the input curves from Table 5.1 have their singularities in the origin $Q(0, 0) \in \mathbb{C}^2$, whereas the input curves from Table 5.2 have their singularities close to the origin. We denote all the input curves with $\mathcal{C}$. For computing the ϵ-algebraic link and the ϵMilnor fibration of the singularity $Q(0, 0)$ of each input curve $\mathcal{C}$, we use the methods from Chapter 3 implemented in the library GENOM3CK.
In both tables, we include:

(i) in the first column, the equation of the input plane complex algebraic curve;

(ii) in the second column, the ϵ-link L_ϵ of the singularity Q of the input curve $\mathcal{C}$, computed as a 3-dimensional graph data structure in the GENOM3CK library;

(iii) in the third column, the two implicit algebraic surfaces from $\mathbb{R}^3$ denoted with $\mathcal{S}_1$ and $\mathcal{S}_2$ that define as their intersection the ϵ-link L_ϵ. We recall that these surfaces are part of the Milnor fibration.

(iv) in the forth column, the surfaces $\mathcal{S}_1$ and $\mathcal{S}_2$, plus the sum $\mathcal{S}_1 + \mathcal{S}_2$ and the difference $\mathcal{S}_1 - \mathcal{S}_2$ of these surfaces, which are also part of the Milnor fibration.

The *third example* indicates the *computation of the invariant properties* attached to the plane complex algebraic curve $\mathcal{C}$ defined by the squarefree polynomial $x^2 - y^3 \in \mathbb{C}[x, y]$. The following output is computed and displayed by the library GENOM3CK as part of a table containing the `Invariant properties` of the input plane complex algebraic curve $\mathcal{C}$ defined by the squarefree polynomial $x^2 - y^3$:

- `Alexander polynomial 1` $\rightarrow x_0^2 - x_0 + 1$ represents the ϵ-Alexander polynomial of the ϵ-algebraic link L_ϵ of the numerical singularity $(-3.72529e-08, 0)$.

- `Delta invariant 1` $\rightarrow 1$ represents the ϵ-delta-invariant of the numerical singularity $(-3.72529e-08, 0)$ of the input curve $\mathcal{C}$.

Table 5.1: Topology analysis with GENOM3CK on exact examples

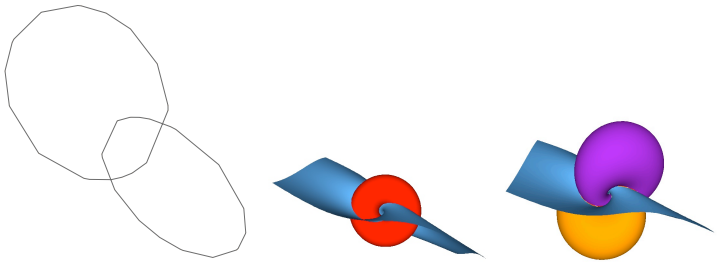
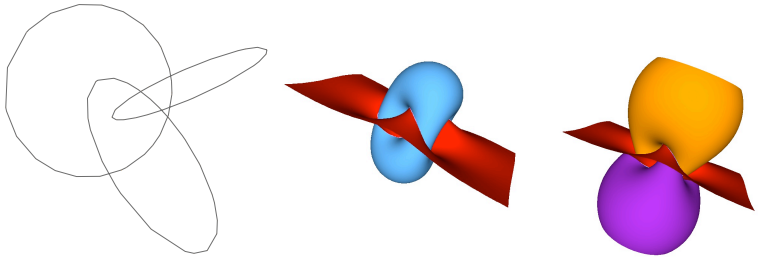
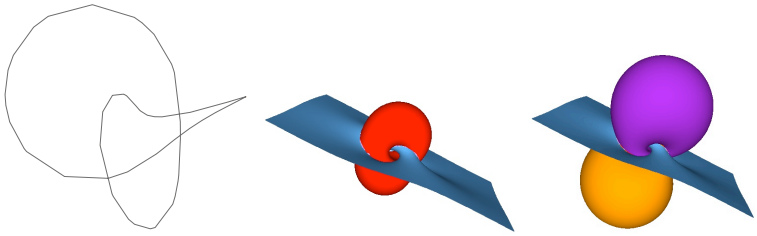
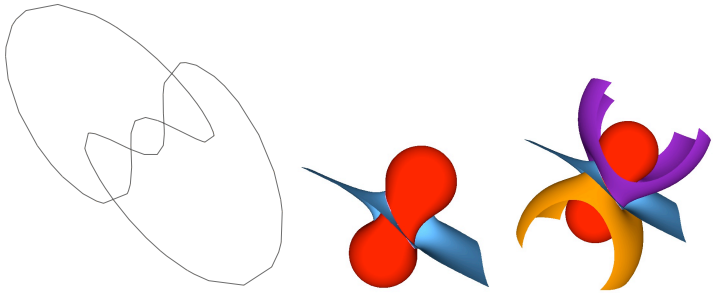
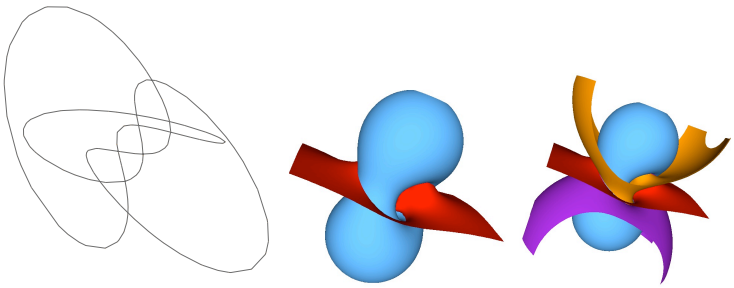
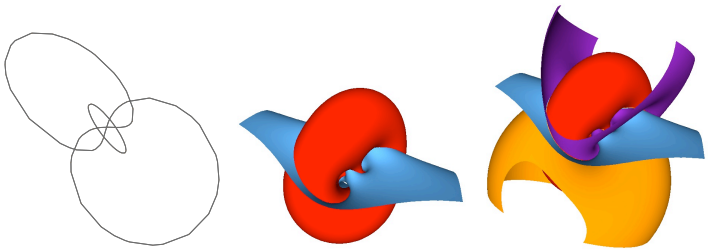
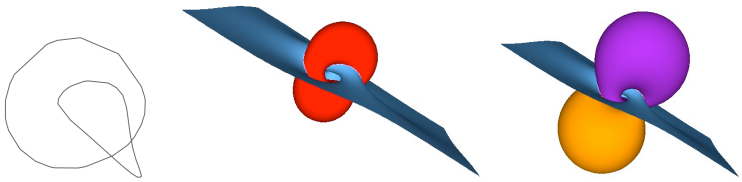
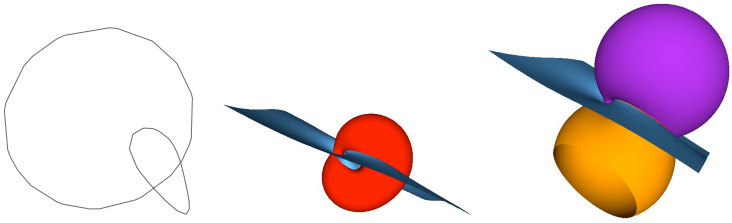
Equation	ϵ-link, ϵ-Milnor fibration of the singularity $(0,0)$
$x^2 - y^2$	
$x^3 - y^3$	
$-x^3 - xy + y^2$	
$x^2 - y^4$	
$x^2 - y^5$	

Table 5.2: Topology analysis with GENOM3CK on inexact examples

Equation	**ϵ-Algebraic link and ϵ-Milnor fibration of the numerical singularity** $(0,0)$
$1.02x^2y + 1.12y^4$	
$-x^3 - xy + y^2 - 0.01$	
$-x^3 - 1.875xy + y^2$	

- `Genus of the complex curve` $\rightarrow 0$ represents the ϵ-genus of the input curve $\mathcal{C}$.
- `Euler characteristic of the Riemann surface attached to the input curve` $\rightarrow 2$ represents the ϵ-Euler characteristic of the Riemann surface attached to the resolution of singularities of the input plane complex algebraic curve $\mathcal{C}$.

The *forth example* indicates the *computation of the topological properties* attached to the plane complex algebraic curve $\mathcal{C}$ defined by the squarefree polynomial $x^2 - y^3 \in \mathbb{C}[x, y]$. The following output is computed and displayed by the library GENOM3CK as part of a table containing the `Topological properties` of the input plane complex algebraic curve $\mathcal{C}$ defined by the squarefree polynomial $x^2 - y^3$:

- `Milnor number 1` $\rightarrow 2$ represents the ϵ-Milnor number of the numerical singularity $(-3.72529e-08, 0)$ of the input curve $\mathcal{C}$.
- `Number of branches 1` $\rightarrow 1$ represents the ϵ-number of branches of the input curve $\mathcal{C}$ through the numerical singularity $(-3.72529e-08, 0)$.

The *fifth example* indicates the *computation of the knot theory properties* attached to the plane complex algebraic curve $\mathcal{C}$ defined by the squarefree polynomial $x^2 - y^3 \in \mathbb{C}[x, y]$. The following output is computed and displayed by the library GENOM3CK as part of a table containing the `Knot theory properties` of the input plane complex algebraic curve $\mathcal{C}$ defined by the squarefree polynomial $x^2 - y^3$:

- `Genus of algebraic link 1` $\rightarrow 1$ represents the genus of the ϵ-link L_ϵ of the numerical singularity $(-3.72529e-08, 0)$ of the input plane complex algebraic curve $\mathcal{C}$. We recall that a link is algebraic if it is equivalent to the link of the singularity of a plane complex algebraic curve.
- `Unknotting number of algebraic link 1` $\rightarrow 1$ represents the unknotting number of the ϵ-link L_ϵ of the numerical singularity $(-3.72529e-08, 0)$ of the input plane complex algebraic curve $\mathcal{C}$.
- `Determinant of algebraic link 1` $\rightarrow 3$ represents the determinant of the ϵ-link L_ϵ of the numerical singularity $(-3.72529e-08, 0)$ of the input plane complex algebraic curve $\mathcal{C}$.
- `Colorability of algebraic link 1` $\rightarrow$ `Algebraic link is colorable via 3-numbering, i.e. the algebraic link is 3-colorable` contains information concerning the colorability property of the ϵ-link L_ϵ of the numerical singularity $(-3.72529e-08, 0)$ of the input plane complex algebraic curve $\mathcal{C}$. In this case, we observe that the ϵ-link L_ϵ of the numerical singularity $(-3.72529e-08, 0)$, which is represented by the trefoil knot, is 3-colorable.
- `Number of knot components of algebraic link 1` $\rightarrow 1$ represents the number of knot components in the ϵ-link L_ϵ of the numerical singularity $(-3.72529e-08, 0)$ of the input plane complex algebraic curve $\mathcal{C}$.
- `Sequence of linking numbers for the knot components of algebraic link 1` $\rightarrow$ `undefined`. If the ϵ-link of an input plane complex algebraic curve contains at least 2-knot components, then we can define a sequence of linking numbers between all the knot components of the link. If the ϵ-link contains only one knot component, then the sequence of linking numbers is undefined. Since the trefoil knot is a link with one knot component, it follows that for the ϵ-link L_ϵ of the singularity $(-3.72529e-08, 0)$ of the input curve $\mathcal{C}$ the sequence of linking numbers is undefined.

The *last example* indicates the *computation times in seconds* required for performing all the operations attached to the plane complex algebraic curve $\mathcal{C}$ defined by the squarefree polynomial $x^2-y^3 \in \mathbb{C}[x,y]$. The following output is computed and displayed by the library GENOM3CK as part of a table containing the `Analysis on properties` information concerning the input plane complex algebraic curve $\mathcal{C}$ defined by the squarefree polynomial x^2-y^3 :

- `For all singularities` $\rightarrow 0.01$ contains the computing time in seconds required for computing the set of singularities of the curve defined by the polynomial $f_1(x,y) = g(x,y,1) = x^2-y^3$. The singularities are computed using subdivision methods from [Mourrain and Pavone, 2009].
- `For first singularities at infinity` $\rightarrow 0$ contains the computing time in seconds required for computing the set of singularities of the curve defined by the polynomial $f_2(y,z) = g(1,y,z) = z-y^3$. The singularities are computed using subdivision methods.
- `For second singularities at infinity` $\rightarrow 0$ contains the computing time in seconds required for computing the set of singularities of the curve defined by the polynomial $f_3(x,z) = g(x,1,z) = x^2z-1$. The singularities are computed using subdivision methods.
- `For topology 1` $\rightarrow 1.68$ contains the computing time in seconds required for computing the graph data structure representing the ϵ-link L_ϵ of the singularity $(-3.72529e-08, 0)$ of the input curve $\mathcal{C}$. The graph data structure representing the ϵ-link L_ϵ is called the topology of the ϵ-link L_ϵ and it is computed using subvidision methods from [Liang et al., 2008].
- `For set of operations 1` $\rightarrow 0.01$ contains the computing time in seconds required for computing the invariant properties attached to the input curve $\mathcal{C}$.
- `For set of extended operations 1` $\rightarrow 0$ contains the computing time in seconds required for computing the topological properties and the knot theory properties attached to the input curve $\mathcal{C}$.

In Table 5.3, in Table 5.4 and in Table 5.5 we include a summary containing the approximate invariants attached to several input plane complex algebraic curves defined by different squarefree bivariate complex polynomials. We mention that the approximate invariants from these tables are computed using the methods developed in Chapter 3 and implemented in the library GENOM3CK.

5.4.2 Examples for the Convergence Property

In this subsection, we give some experimental evidence for the statement that our algorithm is a regularization as explained in Chapter 4. We mention that this subsection contains results from [Hodorog and Schicho, 2011b]. All the experiments, numerical and symbolical, are done with the library GENOM3CK-Symbolic numeric techniques for GENus cOMputation of Complex algebraiC Curves using Knot theory. We recall that GENOM3CK is implemented and included as a library in the free system Axel [Wintz, 2008], written in C++ programming language with the Qt framework.

As evidences for the **convergence for exact data 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

Table 5.3: Summary of computed invariants with the GENOM3CK library. Part I

Equation	ϵ	**Box**	**ϵ-Link**	**ϵ-Invariants**
$x^2 - y^2$	1.0	$[-4, 4, -6, 6, -6, 6]$	Hopf link	$\Delta_\epsilon = 1$ $\delta_\epsilon = 1,\ \mu_\epsilon = 1,\ g_\epsilon = -1$
$x^2 - y^3$	1.0	$[-4, 4, -6, 6, -6, 6]$	Trefoil knot	$\Delta_\epsilon = t_1^2 - t_1 + 1$ $\delta_\epsilon = 1,\ \mu_\epsilon = 2,\ g_\epsilon = 0$
$x^2 - y^4$	1.0	$[-4, 4, -6, 6, -6, 6]$	2-knots links	$\Delta_\epsilon^1 = \Delta_\epsilon^2 = t_1 t_2 + 1$ $\delta_\epsilon^1 = \delta_\epsilon^2 = 2,\ \mu_\epsilon^1 = \mu_\epsilon^2 = 3,\ g_\epsilon = -1$
$x^2 - y^5$	1.0	$[-4, 4, -6, 6, -6, 6]$	1-knot link 1-knot link	$\Delta_\epsilon^1 = t_1^4 - t_1^3 + t_1^2 - t_1 + 1$ $\delta_\epsilon^1 = 2,\ \mu_\epsilon^1 = 4,\ g_\epsilon = 0$ $\Delta_\epsilon^2 = t_1^8 - t_1^7 + t_1^5 - t_1^4 + t_1^3 - t_1 + 1$ $\delta_\epsilon^2 = 4,\ \mu_\epsilon^2 = 8$
$x^3 - y^2$	1.0	$[-4, 4, -6, 6, -6, 6]$	Trefoil knot	$\Delta_\epsilon = t_1^2 - t_1 + 1$ $\delta_\epsilon = 1,\ \mu_\epsilon = 2,\ g_\epsilon = 0$
$x^3 - y^3$	1.0	$[-4, 4, -6, 6, -6, 6]$	3-knots link	$\Delta_\epsilon = -t_1 t_2 t_3 + 1$ $\delta_\epsilon = 3,\ \mu_\epsilon = 4,\ g_\epsilon = -2$
$x^3 + y^4$	1.0	$[-4, 4, -6, 6, -6, 6]$	1-knot link	$\Delta_\epsilon = t_1^6 - t_1^5 + t_1^3 - t_1 + 1$ $\delta_\epsilon = 3,\ \mu_\epsilon = 6,\ g_\epsilon = 0$
$x^3 - y^5$	1.0	$[-4, 4, -6, 6, -6, 6]$	1-knot link 1-knot link	$\Delta_\epsilon^1 = t_1^8 - t_1^7 + t_1^5 - t_1^4 + t_0^3 - t_0 + 1$ $\delta_\epsilon^1 = 4,\ \mu_\epsilon^1 = 8,\ g_\epsilon = 0$ $\Delta_\epsilon^2 = t_1^4 - t_1^3 + t_1^2 - t_1 + 1$ $\delta_\epsilon^2 = 2,\ \mu_\epsilon^2 = 4$
$x^3 + y^6$	1.0	$[-6, 6, -6, 6, -6, 6]$	3-knots link 3-knots link	$\Delta_\epsilon^1 = -t_1^3 t_2^3 t_3^3 - t_1^2 t_2^2 t_3^2 + t_1 t_2 t_3 + 1$ $\delta_\epsilon^1 = 6,\ \mu_\epsilon^1 = 10,\ g_\epsilon = -2$ $\Delta_\epsilon^2 = -t_1^3 t_2^3 t_3^3 - t_1^2 t_2^2 t_3^2 + t_1 t_2 t_3 + 1$ $\delta_\epsilon^2 = 6,\ \mu_\epsilon^2 = 10$

Table 5.4: Summary of computed invariants with the GENOM3CK library. Part II

Equation	ϵ	**Box**	**ϵ-Link**	**ϵ-Invariants**
$x^2 \cdot y + y^4$	0.25 0.5	$[-4,4,-6,6,-6,6]$ $[-6,6,-6,6,-6,6]$	2-knots links	$\Delta_\epsilon^1 = \Delta_\epsilon^2 = t_1^3 t_2 + 1$ $\delta_\epsilon^1 = \delta_\epsilon^2 = 3$ $\mu_\epsilon^1 = \mu_\epsilon^2 = 5\ g_\epsilon = -3$
$1.02x^2 \cdot y + 1.12y^4$	0.25	$[-4,4,-6,6,-6,6]$	2-knots links	$\Delta_\epsilon^1 = \Delta_\epsilon^2 = t_1^3 t_2 + 1$ $\delta_\epsilon^1 = \delta_\epsilon^2 = 3$ $\mu_\epsilon^1 = \mu_\epsilon^2 = 5\ g_\epsilon = -3$
$x^2 - y^2 - y^3$	0.5	$[-4,4,-6,6,-6,6]$	Hopf link	$\Delta_\epsilon = 1$ $\delta_\epsilon = 1,\ \mu_\epsilon = 1,\ g_\epsilon = 0$
$x^4 + x^2 \cdot y + y^5$	0.5	$[-4,4,-6,6,-6,6]$	3-knots link	$\Delta_\epsilon = -t_1^2 t_2^2 t_3 + 1$ $\delta_\epsilon = 4,\ \mu_\epsilon = 6,\ g_\epsilon = 2$
$x^2 + x^4 + y^5$	0.5	$[-6,6,-6,6,-6,6]$	1-knot link	$\Delta_\epsilon = t_1^4 - t_1^3 + t_1^2 - t_1 + 1$ $\delta_\epsilon = 2,\ \mu_\epsilon = 4,\ g_\epsilon = 4$
$-x^3 - x \cdot y + y^2$	0.14	$[-4,4,-6,6,-6,6]$	Hopf link	$\Delta_\epsilon = 1$ $\delta_\epsilon = 1,\ \mu_\epsilon = 1,\ g_\epsilon = 0$
$-x^3 - 1.875x \cdot y + y^2$	0.25	$[-4,4,-6,6,-6,6]$ $[-6,6,-8,8,-8,8]$	Hopf link	$\Delta_\epsilon = 1$ $\delta_\epsilon = 1,\ \mu_\epsilon = 1,\ g_\epsilon = 0$

Table 5.5: Summary of computed invariants with the GENOM3CK library. Part III

Equation	ϵ	**Box**	**ϵ-Link**	**ϵ-Invariants**
$9 - 2x + x^2 - 12y + 6y^2 - y^3$	1.0	$[-4,4,-6,6,-6,6]$	Trefoil knot	$\Delta_\epsilon = t_1^2 - t_1 + 1$ $\delta_\epsilon = 1,\ \mu_\epsilon = 2,\ g_\epsilon = 0$
$3x - 3x^2 + x^3 - 3y + 3y^2 - y^3$	1.0	$[-4,4,-6,6,-6,6]$	3-knots link	$\Delta_\epsilon = -t_1 t_2 t_3 + 1$ $\delta_\epsilon = 3,\ \mu_\epsilon = 4,\ g_\epsilon = -2$
$37 - 60x + 37x^2 - 10x^3 + x^4 + y - y^2 - y^3$	0.25	$[-4,4,-6,6,-6,6]$	Hopf links	$\Delta_\epsilon^1 = \Delta_\epsilon^2 = 1$ $\delta_\epsilon^1 = \delta_\epsilon^2 = 1\ \mu_\epsilon^1 = \mu_\epsilon^2 = 1$ $g_\epsilon = 1$
$x^4 + 2x^2y^2 + y^4 + 3x^2y - y^3$	0.5	$[-6,6,-8,8,-8,8]$	3-knots link	$\Delta_\epsilon = 1 - t_1 t_2 t_3$ $\delta_\epsilon = 3,\ \mu_\epsilon = 4,\ g_\epsilon = 0$
$2x^4 - 3x^2y + y^2 - 2y^3 + y^4$	0.5	$[-6,6,-8,8,-8,8]$	2-knots link Unknot	$\Delta_\epsilon^1 = t_1 t_2 + 1$ $\delta_\epsilon^1 = 2,\ \mu_\epsilon^1 = 3,\ g_\epsilon = 0$ $\Delta_\epsilon^2 = 1$ $\delta_\epsilon^2 = 1,\ \mu_\epsilon^2 = 1$
$x^4 + x^2y^2 - 2x^2y - xy^2 + y^2$	1	$[-4,4,-6,6,-6,6]$	1-knot link	$\Delta_\epsilon = t_1^4 - t_1^3 + t_1^2 - t_1 + 1$ $\delta_\epsilon = 1,\ \mu_\epsilon = 4,\ g_\epsilon = 2$

Table 5.6: Convergence of $-x^3 - xy + y^2$ with exact coefficients

Equation	ϵ	**ϵ-Link**	**ϵ-Alexander, ϵ-delta-invariant, ϵ-genus**
$-x^3-xy+y^2$	1.00	Trefoil knot	$\Delta_\epsilon(t_1) = t_1^2 - t_1 + 1$ $\delta_\epsilon = 1$ $g_\epsilon = 0$
$-x^3-xy+y^2$	0.5	Trefoil knot	$\Delta_\epsilon(t_1) = t_1^2 - t_1 + 1$ $\delta_\epsilon = 1$ $g_\epsilon = 0$
$-x^3-xy+y^2$	0.25	Hopf link	$\Delta_\epsilon(t_1, t_2) = 1$ $\delta_\epsilon = 1$ $g_\epsilon = 0$
$-x^3-xy+y^2$	0.14	Hopf link	$\Delta_\epsilon(t_1, t_2) = 1$ $\delta_\epsilon = 1$ $g_\epsilon = 0$

Table 5.7: Convergence of $-x^3 - xy + y^2 - 0.01$ with inexact coefficients

Equation	ϵ	**ϵ-Link**	**ϵ-Alexander, ϵ-delta-invariant, ϵ-genus**
$-x^3-xy+y^2-0.01$	1.00	Trefoil knot	$\Delta_\epsilon(t_1) = t_1^2 - t_1 + 1$ $\delta_\epsilon = 1$ $g_\epsilon = 0$
$-x^3-xy+y^2-0.01$	0.5	Hopf link	$\Delta_\epsilon(t_1, t_2) = 1$ $\delta_\epsilon = 1$ $g_\epsilon = 0$
$-x^3-xy+y^2-0.01$	0.25	Hopf link	$\Delta_\epsilon(t_1, t_2) = 1$ $\delta_\epsilon = 1$ $g_\epsilon = 0$
$-x^3-xy+y^2-0.01$	0.22	Hopf link	$\Delta_\epsilon(t_1, t_2) = 1$ $\delta_\epsilon = 1$ $g_\epsilon = 0$

numerical singularities of the input curve defined by $f(x, y)$, the ϵ-link of each singularity, the ϵ-Alexander polynomial of each ϵ-link, the ϵ-delta-invariant of each singularity, and the ϵ-genus of the curve. The computation of the ϵ-Alexander polynomial, ϵ-delta-invariant and the ϵ-genus depends on the computation of the ϵ-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 23. We consider $f(x, y) = x^2 - xy - y^3$. We notice that $x^2 - xy = x(x - y)$ thus $f(x, y)$ has a vertical tangent $x = 0$ in $\mathbb{C}^2$. 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 $f(x, y) = -x^3 - xy + y^2$, and thus we consider this polynomial as the input of the problem. We use Arnold's results [Arnold et al., 1985] concerning the analysis of curve singularities and we deduce that the algebraic link of the singularity $(0, 0)$ of the polynomial $-x^3-xy+y^2$ is the same as the algebraic link of the singularity $(0, 0)$ of the polynomial $-xy + y^2$ 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 in Table 5.6 that the approximate solution converges to the exact solution as ϵ tends to 0.

We can consider the input polynomial with both exact and inexact coefficients, such as $f(x, y) = -x^3 - xy + y^2 - 0.01$. We observe in Table 5.7 that the approximate solution converges to the exact solution when ϵ tends to 0. This is an evidence for the convergence for noisy data property from Chapter 4.

Example 24. 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 in Table 5.8 that the approximate solution converges to the exact solution as ϵ tends to 0.

Table 5.8: Convergence of $x^2 - y^2 - y^3$ with exact coefficients

Equation	ϵ	**ϵ-Link**	**ϵ-Alexander, ϵ-delta-invariant, ϵ-genus**		
$x^2 - y^2 - y^3$	1.00	1 singularity curve	–	–	–
$x^2 - y^2 - y^3$	0.7	Hopf link	$\Delta_\epsilon(t_1, t_2) = 1$	$\delta_\epsilon = 1$	$g_\epsilon = 0$
$x^2 - y^2 - y^3$	0.5	Hopf link	$\Delta_\epsilon(t_1, t_2) = 1$	$\delta_\epsilon = 1$	$g_\epsilon = 0$
$x^2 - y^2 - y^3$	0.19	Hopf link	$\Delta_\epsilon(t_1, t_2) = 1$	$\delta_\epsilon = 1$	$g_\epsilon = 0$

Table 5.9: Continuity for perturbations of type I of $-x^3 - xy + y^2$

Perturbations I	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	$\Delta_\epsilon(t_1) = t_1^2 - t_1 + 1$ $\delta_\epsilon = 1$ $g_\epsilon = 0$
$-x^3-xy+y^2-10^{-e}$	0.25	$\{10^{-2}, ..., 10^{-10}\}$	Hopf link	$\Delta_\epsilon(t_1, t_2) = 1$ $\delta_\epsilon = 1$ $g_\epsilon = 0$

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 approximate 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 approximate solution.

Example 25. For the exact polynomial $p(x, y) = -x^3 - xy + y^2$, we consider perturbations of type I of the form $p_\sigma(x, y) = -x^3 - xy + y^2 - \sigma$, with $\sigma \in \{10^{-2}, ..., 10^{-10}\}$. We notice in Table 5.9 that for perturbations of type I of $-x^3 - xy + y^2$ we obtain constant approximate solution.

Table 5.10: Continuity for perturbations of type II of $-x^3 - xy + y^2$

Perturbations II	ϵ	$\sigma = 10^{-e}, e \in \mathbb{N}^*$	**ϵ-Link**	**ϵ-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	$\Delta_\epsilon(t_1,t_2)=1$ $\delta_\epsilon = 1\ g_\epsilon = 0$
$-(1+10^{-e})x^3 - (1+2\cdot 10^{-e})xy+$ $+(1+10^{-e})y^2$	0.14	$\{10^{-1}, ...10^{-10}\}$	Hopf link	$\Delta_\epsilon(t_1,t_2)=1$ $\delta_\epsilon = 1\ g_\epsilon = 0$

Table 5.11: Continuity for perturbations of type I of $x^2 - y^2 - y^3$

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

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. In Table 5.10 we notice that for perturbations of type II of $-x^3 - xy + y^2$ we obtain constant approximate solution.

Example 26. 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 $\sigma \in \{10^{-1}, ..., 10^{-10}\}$. In Table 5.11 that for perturbations of type I of $x^2 - y^2 - y^3$ we obtain constant approximate solution.
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. We notice in Table 5.12 that for perturbations of type II of $x^2 - y^2 - y^3$ we obtain constant approximate solution.

Table 5.12: Continuity for perturbations of type II of $x^2 - y^2 - y^3$

Perturbations II	ϵ	$\sigma = 10^{-e}, e \in \mathbb{N}^*$	**ϵ-Link**	**ϵ-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	$\Delta_\epsilon(t_1,t_2)=1$ $\delta_\epsilon = 1\ g_\epsilon = 0$
$(1+10^{-e})x^2 - (1+3\cdot 10^{-e})y^2 -$ $-(1+4\cdot 10^{-e})y^3$	0.14	$\{10^{-1}, ...10^{-10}\}$	Hopf link	$\Delta_\epsilon(t_1,t_2)=1$ $\delta_\epsilon = 1\ g_\epsilon = 0$

Chapter 6

Conclusions and Future Work

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

Isaac Newton

In this thesis, we approach the algebraic problem of computing topological invariants of a plane complex algebraic curve defined by a squarefree polynomial with both exact (i.e. integer numbers or real numbers) and inexact data (i.e numerical values). For the inexact values we associate a positive real number called noise, which measures the error level in the coefficients. We deal with an ill-posed problem in the sense that tiny changes in the input data of the problem lead to dramatic changes in the output solution. For dealing with the ill-posedness of the considered algebraic problem, we adopt a regularization method. This method computes approximate solutions to the ill-posed problem that are stable under small changes of the input.

We design symbolic-numeric algorithms for computing approximate topological invariants of a plane complex algebraic curve. The designed symbolic-numeric algorithms take as input parameter a positive real number, which is called a regularization parameter. These symbolic-numeric algorithms compute approximate topological invariants for a plane complex algebraic curve in the sense that the computed topological invariants depend on the input regularization parameter. The symbolic-numeric algorithms compute the following approximate topological invariants for a plane complex algebraic curve:

- the set of numerical singularities of the input curve in the projective real plane;
- the approximate link of each numerical singularity of the input curve. We compute the approximate link of each singularity as the stereographic projection of the intersection of the input curve with a small sphere centered in the singularity. The radius of this sphere is represented by the input regularization parameter. The approximate link is a smooth and closed space algebraic curve, given as the intersection of two algebraic surfaces. We compute each approximate link as a 3-dimensional graph data structure, i.e. a set of points in the 3-dimensional space together with their Euclidean coordinates and a set of edges connecting them. We call this 3-dimensional graph data structure the topology of the approximate link. In addition, for each approximate link we compute different properties from knot theory: the genus, the unknotting number, the determinant, the number of knot components, the sequence of linking

numbers between all the knot components of the approximate link. Moreover, for each approximate link, we decide whether it is colorable or not;

- the approximate Alexander polynomial of each approximate link. For computing the approximate Alexander polynomial, we design several computational geometry algorithms such as an adapted version of the Bentley-Ottmann algorithm for computing all the intersection points among the edges of the projection of a 3-dimensional graph data structure. We also use combinatorial objects from knot theory such as the diagram of the approximate link, which is a special type of projection of the approximate link in the 2-dimensional Euclidean plane;
- the approximate delta-invariant of each singularity;
- the approximate Milnor number of each singularity;
- the approximate genus of the input curve;
- the approximate Euler characteristic of the Riemann surface attached to the resolution of singularities of the input plane complex algebraic curve.

We describe the designed symbolic-numeric algorithms using principles from regularization theory. We show that the designed symbolic-numeric algorithms with the regularization parameter compute approximate solutions to the considered algebraic problem, approximate solutions that satisfy the following property (called convergence for noisy data property): as the noise level decreases to zero and as the regularization parameter is chosen according to a certain rule (called parameter choice rule), the approximate solutions computed with the designed symbolic-numeric algorithms tend to the exact solutions of the considered algebraic problem. Instead of computing exact solutions to the considered problem, we compute approximate solutions, which satisfy the convergence for noisy data property. By computing approximate topological invariants that satisfy the convergence for noisy data property, we estimate the topological invariants of a plane complex algebraic curve. Basically, our regularization method consists of the set of designed symbolic-numeric algorithms satisfying the convergence for noisy data property and of the parameter choice rule, which is a function in the noise level.

We completely automatize the designed symbolic-numeric algorithms for computing topological invariants of a plane complex algebraic curve in the new software package called GENOM3CK [Hodorog et al., 2010a] (GENus cOMputation of plane Complex algebraiC Curves using Knot theory). GENOM3CK is an open source library, which is written in the free computer algebra system Mathemagix [van der Hoeven et al., 2002] and in the free algebraic geometric modeler Axel [Wintz et al., 2006]. The library combines graphical, numerical and symbolical capabilities into one package. We perform several test experiments with the library GENOM3CK. A first class of test experiments shows the computation of the approximate invariants of a plane complex algebraic curve. A second class of test experiments validate the convergence for noisy data property of the designed symbolic-numeric algorithms. For symbolic input data, the library computes certified and exact results. For numeric input data, we formalize a framework for interpreting the results of these algorithms in the theory of approximate algebraic computation by using regularization principles. Thus, the library provides certified results for both symbolic and numeric input data, due to the efficient combination between the symbolic and numeric algorithms.

The designed symbolic-numeric algorithms can be used to certify information about the singularities of a plane complex algebraic curve. In addition, we can use homotopy continuation methods from [Sommese and Wampler, 2005] for plane algebraic curves. The homotopy continuation methods (i.e. predictor-corrector methods) can be used to compute

the topology of the approximate link of each singularity of the input plane complex algebraic curve. Moreover, we can use homotopy methods to compute the complex singularities of the input plane complex algebraic curve in the projective complex plane.

Bibliography

[Adams, 2004] Adams, C. C. (2004). *The Knot Book. An Elementary Introduction to the Mathematical Theory of Knots.* American Mathematical Society, Providence, Rhode Island.

[Alberti and Mourrain, 2007] Alberti, L. and Mourrain, B. (2007). Visualization of Implicit Algebraic Curves. In *Proceedings of the 15th Pacific Conference on Computer Graphics and Applications*, pages 303–312. IEEE Computer Society, Washington, DC.

[Alexander, 1928] Alexander, J. W. (1928). Topological Invariant of Knots and Links. *Transactions of the American Mathematical Society*, 30:275–306.

[Allison and Eckel, 2004] Allison, C. and Eckel, B. (2004). *Thinking in C++, Volume 2: Practical Programming.* Prentice Hall Inc., New Jersey, US.

[Anderson et al., 1999] Anderson, E., Bai, Z., Bischof, C., Blackford, S., Demmel, J., Dongarra, J., Croz, J. D., Greenbaum, A., Hammarling, S., McKenney, A., and Sorensen, D. (1999). *LAPACK Users' Guide.* Society for Industrial and Applied Mathematics, third edition.

[Arnold et al., 1998] Arnold, V., Goryunov, V., Lyashko, O., and Vasil'ev, V. (1998). *Singularity Theory. Volume I.* Springer Verlag Berlin Heidelberg.

[Arnold, 2004] Arnold, V. I. (2004). *Catastrophe Theory. Third Edition.* Springer Verlag Berlin Heidelberg.

[Arnold et al., 1985] Arnold, V. I., Varchenko, A. N., and Gusein-Zade, S. M. (1985). *Singularities of Differentiable Maps: Volume 1.* Birkhäuser, Boston.

[Banahan et al., 1991] Banahan, M., Brady, D., and Doran, M. (1991). *The C Book. Second Edition.* Addison-Wesley.

[Bates et al., 2006] Bates, D., Hauenstein, J., Sommese, A., and Wampler, C. (2006). Bertini: Software for numerical algebraic geometry. Software for solving polynomial systems. http://www.nd.edu/ sommese/bertini/.

[Bates et al., 2011] Bates, D. J., Peterson, C., Sommese, A. J., and Wampler, C. W. (2011). Numerical computation of the genus of an irreducible curve within an algebraic set. *Journal of Pure and Applied Algebra*, 215:1844–1851.

[Beman and Smith, 2007] Beman, W. W. and Smith, D. E. (2007). Famous problems of elementary geometry: The duplication of the cube, the trisection of an angle, the quadrature of the circle. An Authorized Translation of F. Klein's Vorträge über ausgewahlte Fragen der Elementargeometrie, Ausgearbeitet von F. Tägert, 1897.

[Berg et al., 2008] Berg, M., Krefeld, M., Overmars, M., and Schwarzkopf, O. (2008). *Computational Geometry: Algorithms and Applications. Second edition.* Springer, Berlin.

[Bieri and Schmidt, 1991] Bieri, H. and Schmidt, P. M. (1991). An on-line algorithm for constructing sweep planes in regular position. In Bieri, H. and Noltemeier, H., editors, *Proceedings of the International Workshop on Computational Geometry*, volume 553, pages 27–35. Springer Berlin Heidelberg.

[Blanchette and Summerfield, 2008] Blanchette, J. and Summerfield, M. (2008). *C++ GUI Programming with Qt 4. Second Edition.* Prentice Hall US.

[Bochnak et al., 1998] Bochnak, J., Coste, M., and Roy, M.-F. (1998). *Real Algebraic Geometry. Second Edition in English.* Ergebnisse der Math. Springer Verlag, Germany.

[Bosma et al., 1997] Bosma, W., Cannon, J., and Playoust, C. (1997). The Magma algebra system. I. The user language. *Journal of Symbolic Computation*, 24(3-4):235–265. Computational algebra and number theory (London, 1993).

[Brauner, 1928] Brauner, K. (1928). Zur Geometrie der Funktionen zweier komplexer Veränderlichen:II-IV. *Abh. Math. Sem. Hamburg*, 6:1–55.

[Brieskorn and Knorrer, 1986] Brieskorn, E. and Knorrer, H. (1986). *Plane Algebraic Curves.* Birkhäuser, Berlin.

[Buchberger and Winkler, 1998] Buchberger, B. and Winkler, F. (1998). *Gröbner Bases and Applications.* London Mathematical Society, Cambridge University Press.

[Burde and Zieschang, 1985] Burde, G. and Zieschang, H. (1985). *Knots.* Walter de Gruyter.

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

[CoCoATeam, 1996] CoCoATeam (1996). CoCoA: A system for doing computations in commutative algebra. http://cocoa.dima.unige.it/s.

[Collins-Sussman et al., 2004] Collins-Sussman, B., Fitzpatrick, B. W., and Pilato, C. M. (2004). *Version Control with Subversion.* O'Reilly Media, Inc., California, US.

[Commons, 2000] Commons, C. (2000). CMake - Cross platform make. A cross platform, open source build system. http://www.cmake.org/.

[Corless et al., 2003] Corless, R. M., Kaltofen, E., and Watt, S. M. (2003). Hybrid methods. In Grabmeier, J., Kaltofen, E., and Weispfenning, V., editors, *Computer Algebra Handbook*, pages 112–125. Springer Verlag, Heidelberg, Germany.

[Corporation, 2008] Corporation, N. (2008). Qt Cross Platform Application and UI Framework. http://qt.nokia.com/.

[Cozzarelli and Wasserman, 1986] Cozzarelli, N. R. and Wasserman, S. A. (1986). Biochemical topology: Applications to DNA recombination and replication. In *Science*, volume 232, pages 951–960.

[Crick and Watson, 1953] Crick, F. H. and Watson, J. D. (1953). A structure for deoxyribose nucleic acids. In *Nature*, volume 171, pages 737–738.

[Crowell, 1959] Crowell, R. H. (1959). Genus of alternating link types. *Ann. Math.*, 69:258–275.

[Crowell and Fox, 1963] Crowell, R. H. and Fox, R. H. (1963). *Intoduction to Knot Theory*. Springer Verlag, New York.

[Diestel, 2005] Diestel, R. (2005). *Graph Theory. Graduate Texts in Mathematics*. Springer Verlag, Heidelberg.

[DoCarmo, 1976] DoCarmo, M. (1976). *Differentiable Geometry of Curves and Surfaces*. Prentice Hall.

[E W. Weisstein, 1999] E W. Weisstein, W. R. (1999). Wolfram mathworld: The web's most extensive mathematics resource. http://mathworld.wolfram.com/.

[Eckel, 2000] Eckel, B. (2000). *Thinking in C++, Volume 1: Introduction to Standard C++. Second Edition*. Prentice Hall Inc., New Jersey, US.

[Eisenbud and Neumann, 1985] Eisenbud, D. and Neumann, W. (1985). Three dimensional link theory and invariants of plane curves singularities. In *Annals of Math. Studies*, volume Study 110. Princeton University Press.

[Engl et al., 1996] Engl, H. W., Hanke, M., and Neubauer, A. (1996). *Regularization of Inverse Problems*. Kluwer Academic Publishers Group.

[Fischer, 2001] Fischer, G. (2001). *Plane Algebraic Curves. Student Mathematical Library, Volume 15*. American Mathematical Society, US.

[Foundation, 2004] Foundation, B. (2004). Blender - the free open source 3d content creation suite. http://www.blender.org/.

[Foundation, 2000] Foundation, F. S. (2000). GMP - the GNU multiple precision arithmetic library. http://gmplib.org/.

[Fousse et al., 2007] Fousse, L., Hanrot, G., Lefeévre, V., Pélissier, P., and Zimmermann, P. (2007). MPFR: A multiple-precision binary floating-point library with correct rounding. *ACM Transactions Mathematical Software*, 33.

[Fulton, 1989] Fulton, W. (1989). *Algebraic curves: An Introduction to Algebraic Geometry*. Addison-Wesley, Redwood City California.

[Gaal, 1998] Gaal, L. (1998). *Classical Galois Theory, With Examples (5th Edition)*. American Mathematical Society.

[Geddes et al., 2008] Geddes, K., Labahn, G., and Monagan, M. (2008). *Maple 12 Introductory Programming Guide*. Maplesoft.

[Gilmore, 1981] Gilmore, R. (1981). *Catastrophe Theory for Scientists and Engineers*. John Wiley and Sons, Inc.

[Greuel and Pfister, 2002] Greuel, G. M. and Pfister, G. (2002). *A Singular Introduction to Commutative Algebra*. Springer Verlag, Berlin Heidelberg.

[Guha, 2011] Guha, S. (2011). *Computer Graphics Through OpenGL: From Theory to Experiments*. Chapman and Hall/CRC, Taylor and Francis Group, US.

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

[Haché, 1994] Haché, G. (1994). Example of axiom package paff. http://axiom-wiki.newsynthesis.org/PAFF.

[Hanrot et al., 2005] Hanrot, G., Lefèvre, V., Pélissier, P., Thveny, P., and Zimmermann, P. (2005). MPFR - the GNU MPFR library. http://www.mpfr.org/.

[Harris and Quenell, 1999] Harris, S. and Quenell, G. (1999). Knot labellings and knots without labelings. *The Mathematical Intelligencer*, 21:51–57.

[Hauser, 2000] Hauser, H. (2000). Resolution of singularities 1860-1999. In Hauser, H., Lipman, J., Oort, F., and Quirós, A., editors, *Resolution of singularities: A Research Textbook in Tribute to Oscar Zariski*, pages 5–36. Birkhäuser.

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

[Hodorog et al., 2010a] Hodorog, M., Mourrain, B., and Schicho, J. (2010a). GENOM3CK - A Library for Genus Computation of Plane Complex Algebraic Curves Using Knot Theory. In *ACM SIGSAM Communications in Computer Algebra*, volume 44, pages 198–200. Association for Computing Machinery, Special Interest Group on Symbolic and Algebraic Manipulation.

[Hodorog et al., 2010b] Hodorog, M., Mourrain, B., and Schicho, J. (2010b). A Symbolic-Numeric Algorithm for Computing the Alexander Polynomial of a Plane Curve Singularity. In Ida, T., Negru, V., Jebelean, T., Petcu, D., Watt, S., and Zaharie, D., editors, *Proceedings of the 12th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing*, pages 21–28. IEEE Computer Society, Los Alamitos.

[Hodorog et al., 2011] Hodorog, M., Mourrain, B., and Schicho, J. (2011). An adapted version of the Bentley-Ottmann algorithm for invariants of plane curve singularities. In Murgante, B., Gervasi, O., Iglesias, A., Taniar, D., and Apduhan, B. O., editors, *Proceedings of 11th International Conference on Computational Science and Its Applications, Session: Computational Geometry and Applications*, pages 121–131. Lecture Notes in Computer Science 6784, Springer.

[Hodorog and Schicho,] Hodorog, M. and Schicho, J. A regularization approach for estimating the type of a plane curve singularity. Journal of Theoretical Computer Science, Submitted.

[Hodorog and Schicho, 2010a] Hodorog, M. and Schicho, J. (2010a). Computational geometry and combinatorial algorithms for the genus computation problem. Technical Report DK Report 2010-07, Johannes Kepler University, Linz.

[Hodorog and Schicho, 2010b] Hodorog, M. and Schicho, J. (2010b). A symbolic-numeric algorithm for genus computation. Technical Report DK Report 2010-06, Johannes Kepler University, Linz.

[Hodorog and Schicho, 2011a] Hodorog, M. and Schicho, J. (2011a). A regularization method for computing approximate invariants of plane curves singularities. In et al., M. M. M., editor, *Proceedings of the 4th International Workshop on Symbolic-Numeric Computation*. Association for Computing Machinery. To appear.

[Hodorog and Schicho, 2011b] Hodorog, M. and Schicho, J. (2011b). A symbolic-numeric algorithm for genus computation. In Langer, U. and Paule, P., editors, *Numerical and Symbolic Scientific Computing: Progress and Prospects*. Springer Wien. To appear.

[Hoeij, 2000] Hoeij, M. (2000). The algcurves (algebraic curves) package in Maple. http://www.math.fsu.edu/ hoeij/compalg/algcurves.html.

[Holzer and Labs, 2008] Holzer, S. and Labs, O. (2008). SURFEX 0.90. Technical report, University of Mainz, University of Saarbrücken. `www.surfex.AlgebraicSurface.net`.

[Jänich, 1984] Jänich, K. (1984). *Topology*. Springer-Verlag New York.

[Jänich, 2000] Jänich, K. (2000). *Vector Analysis*. Springer-Verlag New York.

[Jenks and Sutor, 1992] Jenks, R. D. and Sutor, R. S. (1992). *Axiom, the Scientific Computation System*. Springer Verlag.

[Kaltofen et al., 2007] Kaltofen, E., Yang, Z., and Zhi, L. (2007). Structured low rank approximation of a Sylvester matrix. In Wang, D. and Zhi, L., editors, *Symbolic-Numeric Computation*, Texts and Monographs in Symbolic Computation, pages 69–83. Birkhäuser Verlag, Basel, Switzerland.

[Kauffman, 1991] Kauffman, L. H. (1991). *Knots and Physics*. World Scientific Singapore.

[Kendig, 1977] Kendig, K. (1977). *Elementary Algebraic Geometry*. Springer Verlag, New York Inc.

[Kirwan, 1992] Kirwan, F. (1992). *Complex Algebraic Curves*. Cambridge University Press.

[Kronheimer and Mrowka, 1993] Kronheimer, P. and Mrowka, T. (1993). Gauge theory for embedded surfaces: I. *Topology*, 32:773–826.

[Kuehne and Sullivan, 2008] Kuehne, R. J. and Sullivan, J. D. (2008). *OpenGL Programming on Mac OS X. Architecture, Performance, and Integration*. Addison-Wesley US.

[Lang, 2002] Lang, S. (2002). *Algebra, Graduate Texts in Mathematics, 211, Revised third edition*. Springer Verlag.

[Lee, 2000] Lee, J. M. (2000). *Introduction to Topological Manifolds*. Springer Verlag, New York.

[Li et al., 2004] Li, C., Pion, S., and Yap, C. (2004). Recent progress in exact geometric computation. *Journal of Logic and Algebraic Programming*, 64(1):85–111. Special issue on"Practical Development of Exact Real Number Computation".

[Liang et al., 2008] Liang, C., Mourrain, B., and Pavone, J. (2008). Subdivision methods for the topology of 2d and 3d implicit curves. In Jüttler, B. and Piene, R., editors, *Geometric Modeling and Algebraic Geometry*, pages 199–214. Springer, Berlin Heidelberg.

[Livingston, 1993] Livingston, C. (1993). *Knot Theory*. Mathematical Association of America.

[Marker, 2002] Marker, D. (2002). *Model Theory: An Introduction*. Graduate Texts in Mathematics. Springer New York, United States of America.

[Martin and Hoffman, 2003] Martin, K. and Hoffman, B. (2003). *Mastering CMake: A Cross-Platform Build System*. Kitware Inc., New York, US.

[Milnor, 1968] Milnor, J. (1968). *Singular Points of Complex Hypersurfaces*. Princeton University Press and the University of Tokyo Press, New Jersey.

[Mnuk and Winkler, 1996] Mnuk, M. and Winkler, F. (1996). CASA - A system for computer aided constructive algebraic geometry. In *Proceedings International Symposium on Design and Implementation of Symbolic Computation Systems*, pages 297–307.

[Mourrain and Pavone, 2009] Mourrain, B. and Pavone, J. (2009). Subdivision methods for solving polynomial equations. *Journal of Symbolic Computation*, 44:292–306.

[Mourrain et al., 2008] Mourrain, B., Pavone, J. P., Trebuchet, P., Tsigaridas, E. P., and Wintz, J. (2008). Synaps: A library for dedicated applications in symbolic numeric computing. In Arnold, D. N., Santosa, F., Stillman, M., Verschelde, J., and Takayama, N., editors, *Software for Algebraic Geometry*, volume 148 of *The IMA Volumes in Mathematics and its Applications*, pages 81–109. Springer New York.

[Mourrain and Trébuchet, 2005] Mourrain, B. and Trébuchet, P. (2005). Generalized normal forms and polynomial system solving. In *Proceedings of the 2005 International symposium on Symbolic and Algebraic Computation*, pages 253 – 260. ACM Association for Computing Machinery, New York, NY, US.

[Mumkres, 2000] Mumkres, J. (2000). *Topology. Second edition*. Prentice Hall, Inc. Upper Saddle River, New Jersey.

[Murasugi, 1958] Murasugi, K. (1958). On the genus of alternating knots i, ii. *J. Mathematical Soc. Japan*, 10:94–105.

[Murasugi, 1996] Murasugi, K. (1996). *Knot Theory and Its Applications*. Birkhäuser Boston.

[Network, 2002] Network, O. H. (2002). OpenGL (Open Graphics Library) programming guide. http://glprogramming.com/red/chapter01.html.

[Neumann, 2003] Neumann, W. (2003). Topology of hypersurface singularities. In Berndt, R. and Riemenschneider, O., editors, *Eric Kähler - Mathematische Werke, Mathematical Works*, pages 727–736. Walter de Gruyter Berlin, New York.

[Neuwirth, 1963] Neuwirth, L. (1963). On stallings fibrations. *Proc. Amer. Math. Soc.*, 14:380–381.

[Pérez-Díaz et al., 2010] Pérez-Díaz, S., Sendra, J. R., Rueda, S. L., and Sendra, J. (2010). Approximate parametrization of plane algebraic curves by linear systems of curves. *Computer Aided Geometric Design*, 27(2):212–231.

[Project, 2001] Project, F. (2001). The Fink package management system official homepage. http://www.finkproject.org/index.php.

[Project, 2002] Project, M. (2002). The MacPorts package management system official homepage. http://www.macports.org/.

[Rolfsen, 1976] Rolfsen, D. (1976). *Knots and Links*. Publish or Perish, Inc. Houston, Texas.

[Seifert, 1934] Seifert, H. (1934). über das Geschlecht von Knoten. *Mathematische Annalen*, 110:571–592.

[Semwal, 2002] Semwal, S. K. (2002). OpenGL (Open Graphics Library) tutorial. http://www.cs.uccs.edu/ semwal/indexGLTutorial.html.

[Sendra and Alcazar, 2005] Sendra, J. R. and Alcazar, J. G. (2005). Computation of the topology of real algebraic space curves. *Journal of Symbolic Computation*, 39:719–744.

[Sendra and Winkler, 1991] Sendra, J. R. and Winkler, F. (1991). Symbolic parametrization of curves. *Journal of Symbolic Computation*, 12:607–632.

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

[Sendra et al., 2008] Sendra, J. R., Winkler, F., and Pérez-Díaz, S. (2008). *Rational Algebraic Curves. A Computer Algebra Approach.* Springer Verlag, Berlin Heidelberg, Germany.

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

[Sommese and Wampler, 2005] Sommese, A. J. and Wampler, C. W. (2005). *Numerical Solution of Polynomial Systems Arising in Engineering and Science.* World Scientific, Singapore.

[Stetter, 2004] Stetter, H. J. (2004). *Numerical Polynomial Algebra.* Society for Industrial and Applied Mathematics, Philadelphia.

[CGAL Project Members, 1997] CGAL Project Members (1997). CGAL, Computational Geometry Algorithms Library. http://www.cgal.org.

[Thelin, 2007] Thelin, J. (2007). *Foundations of Qt Development.* Springer Verlag New York.

[Tikhonov and Arsenin, 1977] Tikhonov, A. N. and Arsenin, V. A. (1977). *Solution of Ill-posed Problems.* V. H. Winston & Sons, Washington, D.C.

[Tráng, 1973] Tráng, L. D. (1973). Topologie des singularités des hypersurfaces complexes. *Astérisque*, 7-8:171–182.

[University of Wales, 2004] University of Wales, B. (2004). Knots exhibition. Centre for the Popularisation of Mathematics, http://www.popmath.org.uk/exhib/knotexhib.html.

[van der Hoeven, 2004] van der Hoeven, J. (2004). GNU TeXmacs. In *ACM SIGSAM Bulletin*, volume 38, pages 24–25. Association for Computing Machinery, New York, US.

[van der Hoeven et al., 2002] van der Hoeven, J., Lecerf, G., and Mourrain, B. (2002). Mathemagix open source computer algebra system. http://www.mathemagix.org/www/main/index.en.html.

[Walker, 1978] Walker, R. J. (1978). *Algebraic Curves.* Springer-Verlag, New York, United States of America.

[Winkler, 1996] Winkler, F. (1996). *Polynomial Algorithms in Computer Algebra.* Springer Verlag, Wien, New York.

[Wintz, 2008] Wintz, J. (2008). *Algebraic Methods for Geometric Modeling.* PhD thesis, University of Nice, Sophia-Antipolis.

[Wintz et al., 2006] Wintz, J., Chau, S., Alberti, L., and Mourrain, B. (2006). Axel Algebraic Geometric Modeler. http://axel.inria.fr/.

[Wolfram, 2000] Wolfram, S. (2000). *The Mathematica Book*. Wolfram Research Inc.

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

[Zeng, 2003] Zeng, Z. (2003). A method computing multiple roots of inexact polynomials. In *Proceedings of ISSAC '03 International Symposium of Symbolic and Algebraic Computation (Philadelphia, Pennsylvania, August 3-6, 2003)*, pages 266–272. ACM Press, New York.

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

[Zeng, 2009a] Zeng, Z. (2009a). The approximate irreducible factorization of a univariate polynomial. revisited. In *Proceedings of the 2009 International Symposium on Symbolic and Algebraic Computation*, pages 367–374. ACM New York.

[Zeng, 2009b] Zeng, Z. (2009b). Regularization and matrix computation in numerical polynomial algebra. In Robbiano, L. and Abbot, J., editors, *Approximate Commutative Algebra*, pages 125–162. Springer Verlag, Wien.

[Zeng and Dayton, 2004] Zeng, Z. and Dayton, B. H. (2004). The approximate GCD of inexact polynomials. part ii: A multivariate algorithm. In *Proceedings of the 2004 international symposium on Symbolic and algebraic computation*, pages 320–327. ACM New York.

List of Notations

$\mathbb{R}$	Field of real numbers
$\mathbb{C}$	Field of complex numbers
i	Imaginary unit with $i = \sqrt{-1}$
K	Algebraically closed field of characteristic zero
$\mathbb{R}^2$	2-dimensional Euclidean plane
$\mathbb{R}^3$	3-dimensional Euclidean space
$\mathbb{R}^4$	4-dimensional Euclidean space
$\mathbb{R}^n$	n-dimensional Euclidean space with $n \in \mathbb{N} \setminus \{0\}$
$\mathbb{C}^2$	2-dimensional complex plane
$\mathbb{R}[z, w]$	Ring of polynomials in the indeterminates z, w with coefficients in $\mathbb{R}$
$\mathbb{C}[z, w]$	Ring of polynomials in the indeterminates z, w with coefficients in $\mathbb{C}$
$K[z, w]$	Ring of polynomials in the indeterminates z, w with coefficients in K
$\mathbb{A}^2(\mathbb{C})$	Affine complex plane
$\mathbb{P}^2(\mathbb{C})$	Projective complex plane
$\mathcal{C}, \mathcal{D}$	Affine plane complex algebraic curves
$\tilde{\mathcal{C}}, \tilde{\mathcal{D}}$	Projective plane complex algebraic curves
$p(z, w)$	Bivariate complex polynomial of degree m with $m \in \mathbb{N} \setminus \{0\}$
$p(z, w, u)$	Bivariate complex homogeneous polynomial of degre $m \in \mathbb{N} \setminus \{0\}$
$p_z := \frac{\partial p(z,w)}{\partial z}$	First order partial derivative with respect to z of $p(z, w)$
$p_w := \frac{\partial p(z,w)}{\partial w}$	First order partial derivative with respect to w of $p(z, w)$
$p_{zz} := \frac{\partial p^2(z,w)}{\partial z \partial z}$	Second order partial derivative with respect to z, z of $p(z, w)$
$p_{zw} := \frac{\partial p^2(z,w)}{\partial z \partial w}$	Second order partial derivative with respect to z, w of $p(z, w)$
$p_{ww} := \frac{\partial p^2(z,w)}{\partial w \partial w}$	Second order partial derivative with respect to w, w of $p(z, w)$

$p_{wz} := \frac{\partial p^2(z,w)}{\partial w \partial z}$	Second order partial derivative with respect to w, z of $p(z,w)$
$p_{z^{r-1}} := \frac{\partial p^{r-1}(z,w)}{\underbrace{\partial z ... \partial z}_{(r-1) \text{ times}}}$	Partial derivative of order $(r-1)$ of $p(z,w)$
p_{z^r}	Partial derivative of order r of $p(z,w)$
$\binom{r}{k}$	Binomial coefficient with r, k natural numbers including 0
$n!$	Factorial function of n with $n \in \mathbb{N}$
$C^\infty(\mathbb{R}^n)$	Class of infinitely differentiable (or smooth) functions over $\mathbb{R}^n$
$(\nabla f)(D)$	Gradient of the function f at the point D
$Hf(D)$	Hessian matrix of the function f at the point D
$\circ$	Composition of functions
$\sqrt{\cdot}$	Squareroot function
f^{-1}	Inverse of the functions f
$\cdot$	Dot product
$\|\| \cdot \|\|$	Euclidean norm
$d(\cdot, \cdot)$	Euclidean metric (or distance)
cos^{-1}	Inverse of the cosine function
$B_r(x)$	Open ball of radius r around x
$\overline{B_r(x)}$	Closed ball of radius r around x
sup	Supremum of a set
$\vec{0}_{n+1}$	Origin vector in $\mathbb{R}^{n+1}$
$\vec{0}_n$	Origin vector in $\mathbb{R}^n$
$S_1^n(\vec{0}_{n+1})$	Sphere in $\mathbb{R}^{n+1}$ of radius 1 around $\vec{0}_{n+1}$
$B_1^n(\vec{0}_n)$	Open ball in $\mathbb{R}^n$ of radius 1 around $\vec{0}_n$
$\overline{B_1^n(\vec{0}_n)}$	Closed ball in $\mathbb{R}^n$ of radius 1 around $\vec{0}_n$
$\mathbb{T}^n$	n-dimensional torus
id_X	Identity function on the set X
$\aleph_0$	Cardinality of the natural numbers denoted with aleph-0
$\aleph_c$	Cardinality of the real numbers denoted with aleph-c
max	Maximum of a set
S^1	1-dimensional sphere (or circle)
$[p, q]$	Segment line joining p and q
$\cup$	Union of a collection of sets
$\cap$	Intersection of a collection of sets
K	Knot in $\mathbb{R}^3$
L	Link in $\mathbb{R}^3$

$K_1 \# K_2$	Connected sum of the knots $K_1,\ K_2$
$c(K)$	Crossing number of the knot K
$u(K)$	Unknotting number of the knot K
R	Arbitrary commutative ring with nonzero multiplicative identity 1
$\mathcal{M}_{m\times n}(R)$	Set of $m \times n$ matrices over R
S_ϵ	Sphere of radius ϵ centered in the origin $(0, 0)$
π	Stereographic projection of the sphere S_ϵ from $\mathbb{R}^4$ to $\mathbb{R}^3$
$\lvert \cdot \rvert$	Absolute value of a complex number
$L(Q)$	Link of the plane curve singularity Q
$\Delta(Q)$	Alexander polynomial of the plane curve singularity Q
$\delta(Q)$	Delta-invariant of the plane curve singularity Q
$\mu(Q)$	Milnor-number of the plane curve singularity Q
$genus(\tilde{C})$	Genus of the projective plane complex algebraic curve $\tilde{C}$
$\chi(\mathcal{S})$	Euler characteristic of the compact surface $\mathcal{S}$
$L_\epsilon(Q)$	ϵ-Link of the plane curve singularity Q
$\Delta_\epsilon(Q)$	ϵ-Alexander polynomial of the plane curve singularity Q
$\delta_\epsilon(Q)$	ϵ-Delta-invariant of the plane curve singularity Q
$\mu(Q)$	ϵ-Milnor-number of the plane curve singularity Q
$genus_\epsilon(\tilde{C})$	ϵ-Genus of the projective plane complex algebraic curve $\tilde{C}$
$\chi_\epsilon(\mathcal{S})$	ϵ-Euler characteristic of the compact surface $\mathcal{S}$
$Re(z)$	Real part of the complex number z
$Im(z)$	Imaginary part of the complex number z
$\mathbb{Z}_+$	Set of positive integer numbers including 0
$\mathbb{R}_{>0}$	Set of positive real numbers except 0
$\mathcal{M}_{m\times n}(\mathbb{Z})$	Set of $m \times n$ matrices defined over $\mathbb{Z}$
$\mathcal{M}(m, n\mathbb{Z})$	Set of $m \times n$ matrices defined over $\mathbb{Z}$
$\mathcal{M}_{m\times n}(\mathbb{R})$	Set of $m \times n$ matrices defined over $\mathbb{R}$
$\mathcal{M}(m, n, \mathbb{R})$	Set of $m \times n$ matrices defined over $\mathbb{R}$
$\mathcal{M}_{m\times n}(\mathbb{C})$	Set of $m \times n$ matrices defined over $\mathbb{C}$
$\mathcal{M}(m, n, \mathbb{C})$	Set of $m \times n$ matrices defined over $\mathbb{C}$
$\sim$	Equivalence relation
$\simeq$	Isomorphism
$\cong$	Homeomorphism
$\approx$	Homotopy

List of Figures

List of Tables

Index

Curriculum Vitae

Contact Information

E-mail:	madalina.hodorog@oeaw.ac.at
www:	http://people.ricam.oeaw.ac.at/m.hodorog/

Personal Data

Date of Birth:	7^{th} of January 1983.
Place of Birth:	Deva, Romania.
Citizenship:	Romanian.

Education

2008 – present	Ph.D. Student in the "Doctoral Program: Computational Mathematics" (DK) Johannes Kepler University (JKU), Linz, Austria *Thesis Title:* Symbolic-numeric algorithms for plane algebraic curves *Advisor:* Prof. Dr. Josef Schicho
2007 – 2008	Ph.D. Student at the Research Institute for Symbolic Computation (RISC) Johannes Kepler University, Linz, Austria
2005 – 2007	Master Studies in Computer Science West University of Timişoara, Romania Master Diploma in Computer Science in July 2007 *Thesis Title:* A Case Study in Systematic Theory Exploration: Natural Numbers. *Advisor:* Prof. Dr. Tudor Jebelean and Dr. Adrian Crăciun
2001 – 2005	Bachelor Studies in Mathematics and Computer Science West University of Timişoara, Romania Bachelor Diploma in Mathematics and Computer Science in July 2005 *Thesis Title:* Numerical Methods for Solving Nonlinear Systems of Partial Differential Equations-Parallel Calculus. *Advisor:* Prof. Dr. Dana Petcu
1997 – 2001	National College "Decebal", Deva, Romania Baccalaureate Diploma in June 2001

Academic Work Experience

2008 – present	Research assistant in the Symbolic Computation Group Johann Radon Institute for Computational and Applied Mathematics Linz, Austria
2007 – 2008	Junior researcher Research Institute for Symbolic Computation Hagenberg, Austria
2005 – 2007	Junior researcher at the Research Institute e-Austria West University of Timişoara, Romania

Publications

Refereed Conference Papers

- M. Hodorog, J. Schicho. *A Regularization Method for Computing Approximate Invariants of Plane Curves Singularities.* Accepted in: M. M. Maza et al., editors. Proc. of the 4th International Workshop on Symbolic-Numeric Computation (SNC 2011), ACM (Association for Computing Machinery), San Jose, California, June 7-9, 2011, To appear.

- M. Hodorog, B. Mourrain, J. Schicho. *An Adapted Version of the Bentley-Ottmann Algorithm for Invariants of Plane Curve Singularities.* In: B. Murgante et.al, editors. Proc. of the 11th International Conference on Computational Science and Its Applications (ICCSA 2011), Part III, Session: Computational Geometry and Applications, Lecture Notes in Computer Science 6784, pp. 121-131, Springer, 2011.

- M. Hodorog, B. Mourrain, J. Schicho. *A Symbolic-Numeric Algorithm for Computing the Alexander Polynomial of a Plane Curve Singularity.* In: Proc. of the 12th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC 2010), T. Ida, V. Negru, T. Jebelean, D. Petcu, S. Watt, D. Zaharie (eds.), pp. 21-28, September 23-26 2010. Department of Computer Science, West University of Timişoara, Romania, ISBN: 978-0-7695-4324-6.

- A. Crăciun, M. Hodorog. *Decompositions of Natural Numbers: From A Case Study in Mathematical Theory Exploration.* In: Proceedings of the 9th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC 2007), D. Petcu, V. Negru, D. Zaharie and T. Jebelean (ed.), pp. 1-8, September 26-29 2007. West University of Timişoara, Romania, ISBN:0-7695-3078-8.

- M. Hodorog, A. Crăciun. *Scheme-Based Systematic Exploration of Natural Numbers.* In: Proceedings of the 8th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC 2006), D. Petcu, V. Negru, D. Zaharie, T. Jebelean (ed.), pp. 23-34. September 26-29 2006. Department of Computer Science, West University of Timişoara, Romania, ISBN:0-7695-2740-X.

Contributions in Collections

- M. Hodorog, B. Mourrain, J. Schicho. *GENOM3CK - A Library for Genus Computation of Plane Complex Algebraic Curves using Knot Theory.* In: ACM SIGSAM Communications in Computer Algebra, vol. 44, No. 4, Issue 174, pp. 198-200,

December 2010, ISSN:1932-2240. Software Presentations at the 35th International Symposium on Symbolic and Algebraic Computation (ISSAC 2010), July 25-28 2010, Munich, Germany.

Book Chapters

- M. Hodorog, J. Schicho. *A Symbolic-Numeric Algorithm for Genus Computation.* In: Numerical and Symbolic Scientific Computing: Progress and Prospects, U. Langer and P. Paule (ed.), 30 pp., Springer Wien, 2011, to appear.

Technical Reports

- M. Hodorog, J. Schicho, *A Symbolic-Numeric Algorithm for Genus Computation.* DK Report 2010-06, 31 pp., Johannes Kepler University, Linz-Austria.
- M. Hodorog, J. Schicho, *Computational Geometry and Combinatorial Algorithms for the Genus Computation Problem.* DK Report 2010-07, 30 pp., Johannes Kepler University, Linz-Austria.
- M. Hodorog, A. Crăciun. *A Case Study in Systematic Theory Exploration: Natural Numbers.* Technical report no. 07-18 in RISC Report Series, 38 pp., University of Linz, Austria. October 2007. RISC, University of Linz, Austria.

Extended Abstracts

- M. Hodorog, B. Mourrain, J. Schicho. *Topology Analysis of Complex Curves Singularities Using Knot Theory*, June 24-30, 2010. At: 7th International Conference on Curves and Surfaces, Avignon-France.
- M. Hodorog, B. Mourrain, J. Schicho. *The Genus Computation Problem: Symbolic-Numeric Solutions and Beyond*, March 15-19, 2010. At: Second SAGA Winter Workshop, Auron-France.
- A. Crăciun, M. Hodorog. *The Quotient-Remainder Theorem for Natural Numbers: Discovery by Lazy Thinking.* University of Pécs, Hungary, June 21, 2007. At: First Central and Eastern European Conference on Computer Algebra and Dynamic Geometry Systems in Mathematics Education (CADGME).

Submitted

- M. Hodorog, J. Schicho. *A Regularization Approach for Estimating the Type of a Plane Curve Singularity.* Submitted to the Journal of Theoretical Computer Science. Special Issue on Symbolic and Numeric Computation.

Conferences and Scientific Visits

- *M. Hodorog*, B. Mourrain, J. Schicho. *Contributed talk: An Adapted Version of the Bentley-Ottmann Algorithm for Invariants of Plane Curve Singularities.* At: 11th International Conference on Computational Science and Its Applications (ICCSA 2011), Session: Computational Geometry and Applications, June 20-23, 2011, University of Cantabria, Santander, Spain.
- *M. Hodorog*, B. Mourrain, J. Schicho. *Contributed talk: A Regularization Method for Computing Approximate Invariants of Plane Curves Singularities.* At: 4th International Workshop on Symbolic-Numeric Computation (SNC 2011), June 7-9, 2011, San Jose, California, US.

- *M. Hodorog*, B. Mourrain, J. Schicho. *Contributed talk: A Symbolic-Numeric Algorithm for Computing the Alexander Polynomial of a Plane Curve Singularity.* At: 12th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC 2010), September 23-26 2010, Timişoara, Romania.
- *M. Hodorog*, B. Mourrain, J. Schicho. *Contributed talk: GENOM3CK - A Library for Genus Computation of Plane Complex Algebraic Curves using Knot Theory.* Software Presentation at the 35th International Symposium on Symbolic and Algebraic Computation (ISSAC 2010), July 25-28 2010, Munich, Germany.
- *M. Hodorog*, B. Mourrain, J. Schicho. *Contributed talk: Topology Analysis of Complex Curves Singularities Using Knot Theory.* At: 7th International Conference on Curves and Surfaces, June 24-30 2010, Avignon, France.
- *M. Hodorog*, B. Mourrain, J. Schicho. *Contributed talk: The Genus Computation Problem: Symbolic-Numeric Solutions and Beyond.* At: Second Winter Workshop of the SAGA (Shapes, Geometry and Algebra) Project, March 15-19 2010, Auron, France.
- *M. Hodorog*, A. Crăciun, T. Jebelean. *Contributed talk: Systematic Exploration of Mathematical Theories.* July 27-30, 2008. At: Applications of Computer Algebra (ACA 2008), Session Symbolic Computation and Deduction in System Design and Verification.
- *M. Hodorog*, A. Crăciun. *Contributed talk: Scheme-Based Systematic Exploration of Natural Numbers.* At: 8th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC 2006), September 2006. Department of Computer Science, West University of Timişoara, Romania.
- Participating at the Special Semester on Gröebner Bases, Linz-Hagenberg-Austria, March 2006.
- Research visit at INRIA Sophia-Antipolis, Sophia-Antipolis, France, 2009, 2011.
- Research visit at Colorado State University, Fort Collins, US, 2011.